Technische Universität München

Fakultät für Mathematik

Lehrstuhl für Angewandte Numerische Analysis

Algorithms for Robust and Fast Sparse Recovery

New Approaches Towards the Noise Folding Problem and the Big Data Challenge

Steffen Peter

Vollständiger Abdruck der von der Fakultät für Mathematik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr. Michael Ulbrich
Prüfer der Dissertation:	1.	Univ.-Prof. Dr. Massimo Fornasier
	2.	Univ.-Prof. Dr. Holger Rauhut Rheinisch-Westfälische Technische Hochschule Aachen
	3.	Univ.-Prof. Dr. Xiaoxiang Zhu (schriftliche Beurteilung)

Die Dissertation wurde am 19.05.2016 bei der Technischen Universität München eingereicht und durch die Fakultät für Mathematik am 21.09.2016 angenommen.

Abstract

We analyze and numerically validate novel algorithms for sparse recovery in mathematical signal processing. Our focus is on enhancing both robustness and efficiency with respect to state-of-the-art. Regarding robustness, we propose non-convex formulations of sparse recovery problems, featuring enhanced signal identification properties if the original signal is affected by noise prior to measurements. We address improving efficiency by introducing and analyzing an iteratively re-weighted least squares method, exploiting fast matrix-vector multiplications within a conjugate gradient inner iteration. For large-scale problems we study an enhanced subspace correction method towards parallelization.

Zusammenfassung

Wir analysieren und validieren numerisch neue Algorithmen für Sparse Recovery in mathematischer Signalverarbeitung. Unser Fokus liegt auf der Verbesserung von Robustheit und Effizienz bezüglich des State of the Art. Hinsichtlich der Robustheit schlagen wir nicht-konvexe Formulierungen von Sparse Recovery Problemen vor, welche verbesserte Signalidentifizierungseigenschaften aufweisen, wenn das ursprüngliche Signal durch Rauschen vor der Messung gestört ist. Wir behandeln die verbesserte Effizienz durch das Einführen und die Analyse einer iterativ-neugewichtete kleinste Quadrate Methode, indem wir schnelle Matrix-Vektor Multiplikationen in einer internen Konjugierte Gradienten Iteration ausnutzen. Für großskalierte Probleme untersuchen wir eine verbesserte Unterraum-Korrektur Methode auf Parallelisierungsmöglichkeiten.

Contents

Chapter 1

Introduction

What do the image acquisition techniques used by the German optical Earth remote sensing satellite EnMAP, the rating and marketing system of the film streaming company Netflix, and a composition of Mozart have in common? At first sight, it is not much, but a deep look into the matter reveals that all of them are describable mathematically by means of a concise, or more precisely *sparse*, digital/numerical representation, which allows eventually for a relatively simple elaboration. Let us clarify what we mean with a concise description for each of these mentioned examples. When we listen to the music of Mozart, we perceive it as being "pervasive", filling our ears, continuously for several minutes. Despite the beauty and the involvement Mozart's music can raise, it can relatively simply be represented by its score, as an—admittedly well-thought and marvelous—sequence of a finite number of notes, only partially simultaneously played. The video streaming company Netflix got rather well-known at its early development within the scientific community because of its by now famous contest, the so-called Netflix prize [146], for the best algorithm able to accurately predict client's ratings of movies from ones previously watched. As a matter of fact movies are categorized by genre (Drama, Comedy, Sci-Fi etc.) also because large groups of people may identify their taste with one or few more of them. There are people who may mostly like Sci-Fi, but less other genres, and they would pick first a film to watch from this genre than from others. And tastes of people can often be very similar, so much that they can be additionally grouped according to certain inter-genre selections of films. Hence, the preferences of people can be simplified according to their belonging to certain groups, whose number is actually way smaller than the number of films or the entire cohort of the clients. Once an algorithm is able to establish semi-automatically on the basis of previous preferences the partitioning of the clients into certain preference groups, then the prediction of whether one would like a certain movie gets immediately simplified. This succinct representation of the entire cohort of clients into preference groups is another instance of the mathematical concept of sparsity. Eventually the satellite EnMap, which is supposed to be launched in the year 2018, will acquire high quality hyperspectral measurements from the Earth's surface. The image data, which is produced during this mission, will be of extremely high spectral resolution, but, due to technical restrictions, this is at the expense of

the spatial resolution of the image, which is bound to be lower. However, analyzing a typical image of the surface of the Earth (for instance in urban or landscape areas), one can quickly realize that it can be succinctly described as a sparse collection of particular objects, like roofs, trees, river tracks, etc. Hence, more than identifying high spatial resolution pixels, one could enhance the low spatial resolution of the image, by simply identifying its few composing structures, as we describe in details in Section 1.1.2.

Although by now we recognize that lots of different kind of phenomena can be succinctly represented in terms of few descriptors (as we just exemplified with the cases above) the conceptual development of the abstract notion of sparsity followed certain relevant milestones, which we describe concisely with the following aphorism: the scientific community said, "Let there be *compression* [for images]"; and there was JPEG [132, 60]. The scientific community saw that the *compression* was good, and used the *sparsity* as a prior for solving inverse problems.

The economization to sustain the same results with the minimal investment of resources (such as room, energy, effort or cost) is part of the common experience. In the early days of telecommunication, where message transmission was still expensive, people compressed their messages in a few words on telegrams, or restricted themselves to the important information when utilizing callboxes. Still the messages were received and understood. However, only in the middle of the previous century, Claude E. Shannon introduced a formalization of lossless and lossy compression [165, 166]. Among others it led to the image compression standard JPEG (1980's) and JPEG-2000, where fundamental tools from harmonic analysis, such as the cosine and wavelet transformation respectively, were cleverly exploited.

After the success of the JPEG compression standard, as well as other compression techniques which were applied to other signals such as sounds or specialized technical data, we became aware that most of the acquired data is actually of less importance. Thus, it is reasonable to ask, whether we actually have to acquire all the data, whose large parts can eventually be thrown away, and whether devices can be designed, which directly measure and sketch only the relevant information of the signal without the need of acquiring it all.

In the classical sampling theory, signals are modeled as band-limited functions (i.e., with compactly supported Fourier transform) and can be reconstructed from equidistant samples acquired at the *Shannon-Nyquist sampling rate* [166]. This is actually equivalent to the unique solvability of a linear operator equation, which, in finite dimensions can be simply described as the recovery of a vector $x \in \mathbb{R}^N$ from measurements $y = \Phi x \in \mathbb{R}^m$ acquired by means of a linear sensing process $\Phi \in \mathbb{R}^{m \times N}$, modeling the sampling operator. As we know, if $m \geq N$ and the matrix Φ is of full rank, then the problem is in fact uniquely solvable. However, a directly compressed acquisition of the signal would demand $m \ll N$ instead, violating somehow the classical

understanding of signal acquisition as established by Shannon or the fundamental theorem of linear algebra in our finite dimensional model. In this case, the number of competitor solutions of the problem is infinite. However, the assumption that the signal to be reconstructed is *compressible* may actually help to nail the right one. In this finite dimensional model compressibility means that x can be well-approximated by a sparse vector, i.e., a vector with few nonzero coordinates.

The groundbreaking results of the seminal papers [26, 29, 28, 30, 57, 11] showed that this is actually possible, provided suitable linear measurements Φ and enough compressibility of x, which can be recovered by a relatively simple convex optimization, i.e., the minimization of the vector ℓ_1-norm over the set of feasible competitors. The fundamental principle is that Φ needs to be injective on sparse vectors or, a bit more precisely, that its kernel is "well-separated" from the set of sparse vectors (see Figure 1.1 for an illustration). This gap is often implied by certain properties of the matrix, for instance its quasi-isometrical embedding of sparse vectors in lower dimension, the so-called *Restricted Isometry Property* (RIP), implying also the *Null Space Property* (NSP), i.e., the kernel of the matrix does not contain compressible vectors. One of the fundamental challenges of the theory of *compressed sensing* is then the design of measurements Φ, whose kernel is well-separated from sparse vectors with maximal number k of nonzero entries, which is known to be of order $m/(\log(N/m)+1)$. The best "constructions" currently available rely on a certain level of randomness in the definition of Φ, and no fully deterministic construction is yet known to allow the same optimal sparsity level.

Compressed sensing problems can be considered as inverse problems since one wishes to infer (usually uniquely and stably) the original signal from an undersampled collection of (indirect) measurements. However, the applications of compressed sensing are usually addressing the *engineering design* of a sensing process which mimics the random measurements Φ. For instance, it is by now understood that one can modify MRI machines in a way of randomizing their sensing process and producing high-resolution images with less samples [130, 131], or in *multiple-input-multiple-output* (MIMO) radar one works with random measurements by means of the emission of random probes by several transmitters over some time-period (see, e.g., [61]).

In more general inverse problems and differently from compressed sensing, the design of the measurement is often not completely freely chosen and it is constrained by the physical realization of certain sensors and the physical processes involved in the measurements. Also the measurements are usually nonlinear, but a linearization is often used as a simplification, which works well in many concrete situations. A typical example of an inverse problem is the inversion of the Radon transform used in Computerized Tomography (CT). Although the measurements in these cases may not fulfill the separation of the kernel from sparse vectors, the sparsity prior nevertheless induces implicitly a restriction of the infinite dimensional inverse problem to a rather low dimensional space. As a matter of fact many inverse problems turn out of be

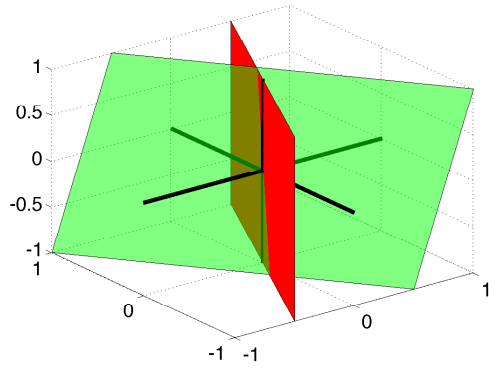


Figure 1.1: For a low-dimensional example with a matrix $\Phi \in \mathbb{R}^{1\times 3}$, we illustrate in the plot the separation of sparse vectors and the kernel of Φ, which can be represented geometrically as a plane. The black lines represent the set of sparse vectors with maximum one non-zero entry. The matrix Φ whose kernel is represented by the red plane is not suitable for compressed sensing since the set of sparse vectors intersects with the kernel. A kernel which is well-separated from the set of sparse vectors is represented by the green plane since both sets only intersect in zero. The respective matrix then is suitable for compressed sensing techniques.

well-posed as soon as one considers their restrictions on finite dimensional subspaces. A relevant well-known example of such a recovered well-posedness in finite dimensions is the Electrical Impedence Tomography (EIT), also known in the mathematical community as *Calderón's inverse conductivity problem*. Here one wants to determine the conductivity of an object, provided measurements in form of Neumann boundary samples. This inverse problem is in general not well-posed. If we however assume that the conductivity is representable as a linear combination of finitely many known profiles, suddenly we obtain a Lipschitz-continuous dependence between the measurements and the conductivity solution, and thus a well-posed problem, see, e.g., [3, Theorem A]. Hence, using sparsity priors for stabilizing an inverse problem is a very powerful method also in very sophisticated (highly nonlinear) situations as in EIT. The scientific community has been aware of the power of the sparsity prior in inverse problems since the 1960's, where ℓ_1-norm minimization was used in the context of sparse frequency estimation [126] (followed by [58]), and in the 1970's for seismic tomography in geophysical exploration [176, 161]. However, only in the 1990's it has been fully understood that the sparsity prior had the potential of significantly outperforming traditional regularizations such as, e.g., ℓ_2-norm minimization of Tychonov type [56, 177]. For instance, J.-L. Starck was one of the first, who used the sparsity prior for significantly improving astronomy imaging [170, 171].

Hence it became very intriguing, as especially it appears in the recent electrical engineering literature on signal processing (see e.g., [203, 202]), to explore the power of the sparsity prior to enhance the recovery of undersampled signals in various contexts, beyond the compressed sensing framework. This development has been additionally boosted by the successful realization of effective algorithms for *dictionary learning*, e.g., [163, 162]. The principle is that classes of statistically related signals may eventually share common features which can be synthesized in a collection of signals, the dictionary, by which all the others can be described by a sparse linear combination. As an example, we already mentioned above that hyperspectral images of Earth surfaces are actually combinable by few fundamental feature images such as roofs, meadows, etc. Hence, we can without concern define this research direction as *sparse recovery*, where one has no control on the measurement process, but has to recover anyway a sparse signal.

In order to successfully establish this field within the applied sciences and industrial research, a robust and efficient computation of sparse signals is of utmost interest. We want to clarify this statement by means of the following detailed reasoning:

- **Robustness:** We already mentioned above that guarantees for the exact and unique recovery of sparse vectors exist as soon as the kernel of the involved matrix Φ and the set of sparse vectors is well-separated (compare Figure 1.1). However, even in such a well-posed setting, real-life-applications from natural sciences and engineering usually are affected by disturbances/perturbations, which concern either the modeling, i.e., signals are not always exactly sparse but most likely compressible, the linearity of the measurement acquisition is only an approximation of the actual likely nonlinear process, or the acquired measurements are corrupted by noise by the design of the involved sensors and environment. Depending on their severity, such perturbations may provoke the failure of the sparse recovery, for instance the wrong identification of the location of the largest components of the signals in absolute value (the essential *support* of the signal). But robustness is an inevitable premise for the responsible application of sparse recovery since some of the already above mentioned examples are critical in the sense that, e.g., human lives may depend on the quality of its solution. For example the reliable identification of small tumors in the brain via MRI allows an early scheduling of a precise surgery and can prevent metastases.

- **Efficiency:** There is by now a large number of algorithms and software for the effective computation of the unknown sparse vector.[1] General purpose optimization methods, such as interior point methods, can be used. However,

[1] Although surely not complete, in [34, 62] one can find a well-kept collection of software and algorithms related to sparse recovery.

fast and robust most specialized methods, taking into account the expected sparsity/compressibility of the solution, have been developed. A popular and simple method is for instance *iterative hard thresholding* [20], which iteratively performs a gradient step, and a projection of the new iterate onto the set of sparse vectors. However, for most of these algorithms their effective scalability with respect to the dimensionality of the problem has not been fully explored as well as their efficiency for Φ not fulfilling the RIP or the NSP is still an open issue. In particular in big data and real-time applications, the computational time is a crucial feature of an algorithm and decisive for its practicalness.

Naturally it would be ideal to produce methods whose robustness and efficiency are guaranteed at the same time, but also providing solutions to enhance one or the other property already is a challenging task.

This thesis is a self-contained compendium of our research work, collected in the research papers [155, 7, 140, 82, 97], the book chapter [81] and the so far unpublished research work of Sections 2.4.1.2, 3.3, and 4.2. In this work, we contribute to the field of sparse recovery by proposing novel methods for enhancing both robustness and efficiency with respect to the state-of-the-art.

Our results can be conceptually summarized into these following two directions:

- **Robust recovery techniques when strong noise on the signal is present [Chapter 3]:** We noted above, that perturbations in applications can have different reasons. We want to focus on the case where the original signal is affected by noise prior to the measurements, essentially destroying its compressibility. This situation, although expected to happen often in real-life applications, was so far only rarely addressed in the literature, in contrast to noise affecting the measurements. The reason of seriously considering the signal noise has been highlighted in [6, 51] where the so called *noise folding* phenomenon was demonstrated: the variance of the noise on the signal prior to measurement gets amplified by the measurements by a factor of $\mathcal{O}(N/m)$, i.e., inversely proportionally with respect to the number of measurements, but also scaling linearly with the dimension of the signal. We provide some theoretical indications that standard methods in sparse recovery are failing even just to detect the relevant entries of the original signal. The reason is their lack of selectivity, i.e., they do not distinguish between signal and noise, which have often instead different statistical properties. For instance, a signal is sparse/compressible, while Gaussian noise is uniformly distributed over the entries of the vector. In view of this observation, we propose two approaches for a more robust recovery:

  1. We design non-convex and non-smooth objective functions, which allows to select large signal components and damp those components which are

attributed to noise. We consider first the optimization of a selective least p-powers functional subject to affine constraints. This method is designed to apply a selective choice, and outperforms standard methods in terms of robustness. Unfortunately, our implementation of it does not scale well with the dimensionality of the problem. Secondly, also to enhance the scalability with respect to the dimension, we revisited the well-known iterative hard thresholding (compare Section 2.4.3.2), by additionally considering a postprocessing correction, performed by a suitable convex program.

2. We minimize multi-penalty functionals, i.e., the summation of a fidelity term, which ensures that the measurement data is met, and at least one possibly non-convex penalty term for the signal and the noise component respectively. By such an approach we are able to take the particular statistics of signal and noise into account, which helps to separate both parts through differently designed penalties. A feature of this approach is its universality, which means that it is not only applicable to the case where the signal is sparse and the noise is Gaussian, but can also potentially be applied to signal-noise combinations with different statistical properties.

Eventually, we present extensive numerical tests, where we compare both approaches with respect to the state-of-the-art methods, regarding their ability of correctly identifying the support of a signal.

- **Accelerated sparse recovery methods based on either efficient matrix-vector multiplications, or distribution techniques (parallelization) [Chapter 4]:** The need for accelerated sparse recovery methods is intrinsically motivated by one of the field's main drivers—the challenge of big data, i.e., the fact that one would like to solve real-world problems with huge dimensions, which are owed to the constantly growing acquisition of data in our quotidian life. We propose two conceptually different methods:

  1. We combine an iteratively re-weighted least squares algorithm, as introduced in Section 2.4.1, with a conjugate gradient acceleration for the approximate solution of the incorporated linear systems (instead of using exact methods such as Gaussian elimination). It is particularly suited to compute solutions of sparse recovery problems, in which the measurement process can be represented by a matrix Φ which allows efficient matrix-vector multiplications (e.g., the fast Fourier transform (FFT)). Although, already used in practice, we thoroughly analyze this algorithm and propose several speed-up techniques in order to make this second-order method competitive with first-order methods—in particular for problems of large size—as we show in the respective numerical results.

  2. In the case that we are not able to use efficient matrix-vector multiplications,

but have to deal with extremely large matrices, without a particular structure, we are likely forced to split the problem. Besides reviewing some recent approaches for the parallel treatment of sparse recovery problems, we resume a domain decomposition method, which was conceptually proposed in an early phase of the sparse recovery "hype". It basically splits the large problem into many smaller ones (domain/subspace decomposition), which can be solved more efficiently. Surprisingly enough, this relatively simple and natural approach has not been fully explored so far, and it was limited to a relatively conceptual level. We reconsider it and tune it to scale for realistic dimensionality of real-life applications and we show the improvement over the state-of-the-art in respective high-performance computing tests.

Additionally, we provide in Chapter 2 a synthetic overview on the fundamentals and most important aspects of sparse recovery. We focus on the theoretical foundations which are required within the thesis. In particular, we recall the most common and intuitive algorithms for sparse recovery. We conclude the thesis in Chapter 5 with the main conclusions of our research and the identification of possible starting points for further investigations and open research questions.

In the remainder of this introduction, we want to present more in detail some applications of sparse recovery and compressed sensing in Section 1.1, in order to underline the relevance and diversity of the topic. Then, we clarify the basic notations, which are used all over the document, in Section 1.2.

1.1 Applications of Sparse Recovery—A Tour from Underwater to Far Galaxies

In order to highlight the importance and influence of the concept of sparsity, we want to take a tour from underwater to far galaxies and briefly present three examples for sparse recovery applications.

1.1.1 Underwater Acoustics

After the tsunami of 2004 in the Indian Ocean, which caused over 230,000 deaths[2], the need was raised for early warning systems for possible causes like underwater earthquakes, volcanic eruptions, etc. Those systems incorporate sensor networks in the ocean which constantly sense possible irregularities in the ocean's underwater acoustics. Underwater acoustic channels can be sparsely represented by a number of distinct

[2]Source: `http://www.spiegel.de/panorama/gesellschaft/tsunami-2004-in-suedost-asien-die-grosse-flut-a-1006392.html`

paths, each characterized by a triplet of delay, Doppler rate, and path attenuation. In [14], the authors reduce the discretized measurement process of those paths towards an dependency of only a delay vector x, from a Fourier-type sensing matrix Φ, i.e., $y = \Phi x + e$, with an additional noise vector e. Since only a few delays are non-zero, x is expected to be sparse and by compressed sensing techniques, one is able to reduce the size of the measurements y, i.e., in practice, less sensors or a lower sampling rate.

1.1.2 Sparse Fusion of Hyperspectral and Multispectral Imagery

We leave the ocean and proceed towards the mainland, which is observed from the space by a diversity of artificial satellites, which have very different purposes. The Environmental Mapping and Analysis Program (EnMAP) is a future German hyperspectral satellite mission. It aims at monitoring and characterising the Earth's environment on a global scale [52, 175]. Among others, the satellite is equipped with a hyperspectral sensor, which produces *hyperspectral data*, i.e., a bunch of images (channels) of the same spatial scene in different spectral ranges. It is the opposite of a greyscale image, which has only one single channel, and is also called *panchromatic image*. In between, one is talking about the so-called *multispectral image* if the number of spectral channels is low (2–10), e.g., the standard RBG image, which has three channels: red, blue, and green. Hyperspectral data is used for instance for the identification of the particular composition (water, vegetation, sand, etc.) of the Earth's surface. The term "hyperspectral" implicitly means, that the resolution of the spectrum is high, since each channel only presents a small portion of the spectrum. Due to physical restrictions in the sensor design, a high spectral resolution comes at the expense of a degradation in spatial resolution. Thus, by deteriorating the spectral resolution, one is able to obtain a better spatial resolution, e.g., in *multispectral data* that has less spectral channels. Thus, from hyperspectral data one is able to distinguish the different materials of a particular region, but one can only roughly say something about the spatial distribution of those materials, while in multispectral data it is the other way round. An example of a hyperspectral low resolution image and multispectral high resolution image is given in Figure 1.2. It was proposed in [101, 98], to fuse both data in order to have the best of both worlds—high spatial and high spectral resolution. The methodology is based on a sparse representation of the data and was already proposed for the *pansharpening problem* in [202], where multispectral data was sharpened through a panchromatic image. We want to describe the idea behind the pansharpening problem, i.e., the fusion of a multispectral and a panchromatic image, for the sake of the conceptual understanding. Then the sharpening of a hyperspectral image by a multispectral image is only a generalization of this approach, but containing more technical details. We refer to [100, 101, 98, 96, 95, 97, 99] for further reading.

The essential assumption is that the scene of interest can be composed of several basic elements like roofs, meadows, lanes, etc. Thus, technically, we obtain such

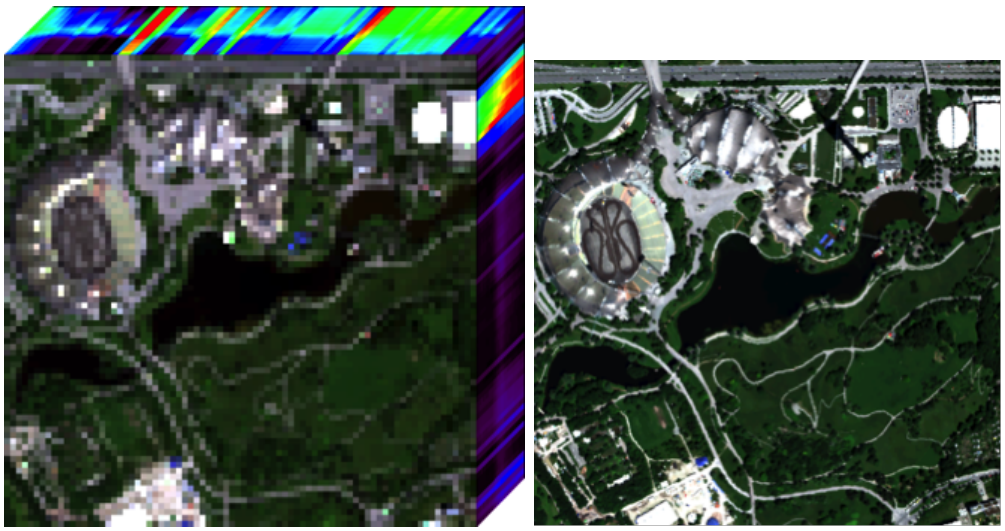


Figure 1.2: Left: Hyperspectral image of low spatial resolution. Right: Multispectral image of high spatial resolution.

elements (or *atoms*) from a set of equally sized patches (subimages) of the original scene, which we call dictionary. A high (spatial) resolution dictionary D_{high} is obtained by choosing patches from the panchromatic image. In practice this means that we choose N different patches of the panchromatic image and store those atoms vectorized as $d_1, \dots, d_N \in \mathbb{R}^{m_h}$, where m_h is the number of pixels of such a subimage. Then, the dictionary matrix $D_{\text{high}} \in \mathbb{R}^{m_h \times N}$ is composed of the column vectors $d_1, \dots, d_N$. The atoms can be low-pass filtered and downsampled in order to obtain a low (spatial) resolution dictionary $D_{\text{low}} \in \mathbb{R}^{m_l \times N}$, where $m_l < m_h$. By our assumption that the scene is composed of those atoms, any (vectorized) patch $Y_{\text{low}} \in \mathbb{R}^{m_l}$ of any channel of the multispectral image can be represented sparsely by atoms of D_{low}. Thus, we determine a sparse solution $X \in \mathbb{R}^N$ of the system $Y_{\text{low}} = D_{\text{low}} X$. The hope is that one obtains from $Y_{\text{high}} = D_{\text{high}} X$ a patch with enhanced spatial resolution. Repeating this procedure for each patch of the low spatial resolution multispectral image, we obtain a high spatial resolution multispectral image.

One can further extend this approach by a *joint-sparse* representation [205, 206]: A patch of the multispectral image is represented by the same part of the image in different channels. If one would perform the sparse recovery in each channel independently, one may get very different sparsity patterns. However, since in all channels the same object is pictured (indeed in different spectra), one may assume that the sparse representation in each channel is approximately the same. Thus, one has to impose joint-sparse penalties on the problem formulation (compare Section 2.3). In Section 4.2.6, we get

back to this application in more detail and provide solutions for an efficient computation.

1.1.3 Pulsating Stars

Eventually, we travel from the low Earth orbit of EnMAP further to the far away stars. Asteroseismologists, who study the oscillations of variable pulsating stars as seismic waves, are particularly interested in the characteristic pattern of the pulsations of a star since it allows the researchers to gain information about its internal layers. We want to follow the presentation in [168, 64] in order to show how sparse recovery can be used for the identification of the star's pulsation.

On the surface of a star there are regions that are expanding and contracting. The expanding part is cooling down, and the contracting part heating up. It is a constant conversion of kinetic into thermal energy and vice versa, and leads to radial and non-radial oscillations, which condense to variabilities in light intensity, observable by human made devices such as the KEPLER space telescope.

Let us consider the wavelength distribution $u(\nu, t)$ on the star's surface (the part that is observable from the Earth), depending on the polar angle $\nu \in [-\pi/2, \pi/2]$, and the time t. From m sensors, we measure this distribution in different spectra (multi- or hyperspectral sensor), where each sensor has a proper sensibility function $w_i(u(\nu, t))$ which is only non-zero in the sensible wavelength range. Stars are too far away in order to resolve the measurements on the level of the polar angle ν, so that only the measurement of the integral

$$y_i(t) = \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} w_i(u(\nu, t))u(\nu, t)\partial\nu$$

is possible.

Within a simple model, asteroseismologists assume that $y_i(t)$ can be well approximated from 2π-periodic functions. As an element of the space of passband filtered trigonometric functions (the N-dimensional approximation of the infinite dimensional space $L^2(-\pi, \pi)$, with N being even), it can be represented by the orthonormal basis of the trigonometric polynomials of maximal degree $N/2$, thus

$$y_i(t) = \sum_{j=-\frac{N}{2}}^{\frac{N}{2}-1} x_j e^{2\pi i j t}.$$

In the community it is accepted that we can assume only some of the frequencies $j \in [-N/2, N/2]$ being active. Like this, we found a sparse representation of the star's oscillations, and we obtain the measurement process $y(t) = \Phi(t)x$, with $(\Phi(t))_{i,j} = e^{2\pi i j t}$ for each instance of time t.

It turns out that the above model can be crucially enhanced since in general one is not able to measure $y_i(t)$ in such a pure linear form which is due to the fact that the rotation axis of the star is inclined and the additional effect of *limb darkening*[3] exists. Instead of a linear model, a quasi-linear model $y(t) = \Phi(t,x)x$ can be established, where $\Phi(t,x)$ is a small perturbation of $\Phi(t)$ depending on x. This modeling leads to a new research field called *quasi-linear compressed sensing*, but will not be addressed in this thesis. For further reading, we refer to the references, mentioned above.

1.2 Notation

Define by $\mathbb{N}$ the set of positive natural numbers, and $\mathbb{N}_0 = \{0\} \cup \mathbb{N}$. The real and integer numbers are denoted by $\mathbb{R}$ and $\mathbb{Z}$ respectively. We denote by

$$\|z\|_{\ell_q} := \left(\sum_{j=1}^{d} \|z_j\|^q \right)^{\frac{1}{q}}, \quad 0 < q < \infty, \quad \|z\|_{\ell_\infty} := \max_{j=1,\dots,d} |z_j|,$$

the standard vector q-(quasi)-norm in $\mathbb{R}^d$, $d \in \mathbb{N}$.

The most general spaces that are used in this thesis are defined as follows: For some countable index set Λ, we denote by $\ell_p(\Lambda)$, $1 \le p \le \infty$, the Banach space of real[4] p-summable (vector) sequences, i.e., $u = (u_i)_{i\in\Lambda}$, $u_i \in \mathbb{R}^d$, $\infty > d \in \mathbb{N}$, and we define the (quasi)-norms

$$\|u\|_{\ell_{p,q}(\Lambda)} := \left(\sum_{i\in\Lambda} \|u_i\|_{\ell_q}^p \right)^{\frac{1}{p}}, \quad 0 < p < \infty, \quad 0 < q \le \infty$$

and $\|u\|_{\ell_{\infty,q}(\Lambda)} := \sup_{i\in\Lambda} \|u_i\|_{\ell_q}$. For simplicity of notation, we define

$$\|u\|_{\ell_p(\Lambda)} := \|u\|_{\ell_{p,2}(\Lambda)}, \quad 0 < p \le \infty. \tag{1.1}$$

For the particular case of $p = q = 2$, we equip $\ell_2(\Lambda)$ with a scalar product, which is given for $u, v \in \ell_2(\Lambda)$ by

$$\langle u, v \rangle := \sum_{i\in\Lambda} \langle u_i, v_i \rangle_{\ell_2},$$

where $\langle \cdot, \cdot \rangle_{\ell_2}$ is the standard scalar product of vectors. We indicate the support of an element $u \in \ell_2(\Lambda)$ by

$$\operatorname{supp}(u) := \{i \in \Lambda \,|\, \|u_i\|_{\ell_2} \neq 0\}.$$

[3] The effect of limb darkening describes the weaker radiation power at the limb regions of the star, where less mass is contributing to the light intensity which is observable from the Earth.

[4] For simplicity, we restrict ourselves to real-valued vector spaces, although most of the theory is also valid for complex-valued vector spaces, as one can verify by cross-reading in the respective literature.

Note that this definition means in particular, $i \in \operatorname{supp}(u)$ if at least one entry of the vector u_i is non-zero. We will need the general spaces, as defined above for arbitrary d, only in the Section 4.2. In the remainder of the thesis, we set $d = 1$, which also motivated the simplification (1.1). Furthermore, a finite dimensional setting is considered in Section 3.1 and 4.1. In this case, $\Lambda = \{1, \dots, N\}$, for $N \in \mathbb{N}$, and $\ell_p(\Lambda) = \mathbb{R}^N$, and we use the short notation $\|\cdot\|_{\ell_p}$ instead of $\|\cdot\|_{\ell_p(\Lambda)}$.

The operator $\#$ specifies the cardinality of a finite set. In particular we define the ℓ_0-"norm" for $u \in \ell_2(\Lambda)$ by

$$\|u\|_{\ell_0} := \|u\|_{\ell_0(\Lambda)} := \#\operatorname{supp}(u).$$

Notice that this is actually not a norm, since it is not homogeneous. However, this term was coined in the field of sparse recovery and compressed sensing.

We will usually consider linear operators $T\colon \ell_2(\Lambda) \to \mathcal{H}$, where $\mathcal{H}$ is a Hilbert space. In particular, we consider in most cases the particular space $\mathcal{H} = \mathbb{R}^{m\times d}$ (and thus $\mathcal{H} = \mathbb{R}^m$ in large parts of the document). Let T^* denote its adjoint operator. In finite dimensions these operators can be represented as matrices Φ. Since we work with real values, the adjoint Φ^* is equal to the transpose of the matrix Φ. For simplicity of notation, we denote $\|T\| := \|T\|_{\ell_2(\Lambda)\to\mathcal{H}}$ the operator norm, which coincides with the spectral norm in the finite dimensional setting. Furthermore, the definitions

$$\begin{aligned} \mathcal{F}_T(y) &:= \{u \in \ell_2(\Lambda) \,|\, Tu = y\}, \\ \mathcal{N}_T &:= \{u \in \ell_2(\Lambda) \,|\, Tu = 0\}, \end{aligned} \tag{1.2}$$

are abbreviations for the solution set of the operator equation $Tu = y$ (or respectively defined for a matrix Φ and $u \in \mathbb{R}^N$) for the measurement vector $y \in \mathcal{H}$, and the *null space* (kernel) of T respectively.

For the index set $\tilde{\Lambda} \subset \Lambda$, we denote the complement by $\tilde{\Lambda}^c := \Lambda \backslash \tilde{\Lambda}$, where in general it should be clear from the context which is the index set of reference Λ. We define the restriction of $u \in \ell_2(\Lambda)$ to the index set $\tilde{\Lambda}$ component-wise by

$$\left(u_{\tilde{\Lambda}}\right)_i := \begin{cases} u_i, & i \in \operatorname{supp}(\tilde{\Lambda}) \\ 0, & i \notin \operatorname{supp}(\tilde{\Lambda}) \end{cases}.$$

For $r \in \mathbb{R}$, we denote the (Gaussian) floor and ceiling function by $\lfloor r \rfloor$ and $\lceil r \rceil$ respectively. For integers $a, b \in \mathbb{Z}$ the remainder of the Euclidean division is denoted by "$a \bmod b$".

Chapter 2

Fundamentals of Sparse Recovery

We want to translate the concept of sparse recovery, which we colloquially described in the introduction of this thesis, into a proper mathematical language. In the following, we present a selected collection of fundamental results in the field, which are essential for the comprehension of the research that is presented in Chapters 3 and 4. More extended tutorials on the matter are given, e.g., in [11, 32, 77, 75, 84, 81, 68].

To the greatest extent, we focus on the setting of finite dimensional spaces. Where it is necessary, we further extend the explanations towards an infinite dimensional setting, for the sake of a self-contained presentation of the thesis.

2.1 A Linear Acquisition Model for Sparse Recovery

The unknown vector $x \in \mathbb{R}^N$, which is referred to as *the signal*, is sampled by a linear *encoder* or *encoding matrix* $\Phi \in \mathbb{R}^{m \times N}$. The simplest possible model is

$$y = \Phi x, \tag{2.1}$$

where the *measurement* $y \in \mathbb{R}^m$ is obtained by a simple linear acquisition not affected by any disturbance. We also refer to it as the *noiseless* model. In the entire thesis, we assume that Φ has full rank, i.e., $\operatorname{rank}(\Phi) = \min\{m, N\}$. From standard linear algebra, it is known that $m = N$ is required to obtain a unique solution to the linear system (2.1). A sparse recovery problem is characterized by the particular requirement $m \ll N$. In this case, there are infinitely many solutions to (2.1). The key idea is to identify among them the *sparse* solutions, which are mathematically modeled in the following section.

2.1.1 Sparse and Compressible Signals

Let us describe the mathematical concept of sparse signals by the following definition.

Definition 2.1 (k-sparse vector)
Let $k \in \mathbb{N}$, $k \leq N$. We call the vector $x \in \mathbb{R}^N$ *k-sparse* if

$$x \in \Sigma_k := \left\{ z \in \mathbb{R}^N | \# \operatorname{supp}(z) = \|z\|_{\ell_0} \leq k \right\}.$$

Sparse signals are a rather ideal construct. In applications, signals are often not exactly sparse but at least compressible. We refer to [132] for more details. We define compressibility in terms of the *best k-term approximation error* with respect to the ℓ_p-norm.

Definition 2.2 (Best k-term approximation)
Let x be an arbitrary vector in $\mathbb{R}^N$. We denote the *best k-term approximation* of x by

$$x_{[k]} := \underset{z\in\Sigma_k}{\arg\min} \|x - z\|_{\ell_p}, \qquad p \in \mathbb{R},\ 1 \le p < \infty,$$

and the respective *best k-term approximation error* of x by

$$\sigma_k(x)_{\ell_p} := \min_{z\in\Sigma_k} \|x - z\|_{\ell_p} = \left\|x - x_{[k]}\right\|_{\ell_p}.$$

Remark 2.3
The best k-term approximation error is the minimal distance of x to a k-sparse vector. Informally, vectors having a relatively small best k-term approximation error are considered to be *compressible*.

Remark 2.4
If we define the *nonincreasing rearrangement* of x by

$$r(x) = (|x_{i_1}|, \dots, |x_{i_N}|)^T, \text{ and } |x_{i_j}| \ge |x_{i_{j+1}}|,\ j = 1, \dots, N-1,$$

then

$$\sigma_k(x)_{\ell_p} = \left(\sum_{j=k+1}^{N} r_j(x)^p\right)^{\frac{1}{p}}, \quad 1 \le p < \infty.$$

Thus, we can alternatively describe the best k-term approximation error by

$$\sigma_k(x)_{\ell_p} = \left(\sum_{j\in\Lambda^c} |x_j|^p\right)^{\frac{1}{p}},$$

where $\Lambda := \operatorname{supp}(x_{[k]})$, and Λ^c is its complement in $\{1, \dots, N\}$.

The quotient k/N describes the *level of sparsity*. If it is small, we talk of a *high/strong sparsity (level)*, and if it is big, we talk of a *low sparsity (level)*. However, there is no particular ratio as, e.g., $k/N < 1/2$, which would restrict the usage of the term "sparsity". As soon as a vector contains at least one single vanishing element, i.e., $k \le N - 1$, one may call it "sparse". Nevertheless the goal in applications is in general the identification of problems with a high sparsity level, i.e., $k/N \ll 1$, since it turns out to be theoretically advantageous, as we mention at the end of Section 2.1.4.

2.1.2 A Simple Decoder

After clarifying the definition of sparse signals, we ask whether one can (uniquely) robustly identify sparse solutions of (2.1), by means of an efficient nonlinear *decoder* $\Delta\colon \mathbb{R}^m \to \mathbb{R}^N$. Colloquially, solving a sparse recovery problem is the identification of the "simplest" description of the data y by a linear combination of the columns of Φ with a minimum of non-zero coefficients. As a matter of fact, the most intuitive decoder is the optimization problem

$$\Delta_0(y) := \operatorname*{arg\,min}_{z \in \mathcal{F}_\Phi(y)} \|z\|_{\ell_0}, \tag{2.2}$$

where we recall the definition of $\mathcal{F}_\Phi(y)$ in (1.2). This decoder is called the *ℓ_0-norm minimization* or *ℓ_0-minimization*, although $\|\cdot\|_{\ell_0}$ is actually not a norm (compare Section 1.2). This problem is known to be *NP-hard*[1] [133, 138]. As long as the solution is expected to be very sparse, a brute-force combinatorial approach may be efficient. However, the field of sparse recovery is motivated by big data problems, and the solution of (2.2) becomes very quickly computationally intractable when the dimension of the problem gets large. Therefore, we search for a proper relaxation of the problem which can lead to tractable algorithms.

In order to see a relaxation of $\|x\|_{\ell_0}$, we define

$$|t|_0 := \begin{cases} 1 & t \neq 0 \\ 0 & t = 0 \end{cases},$$

so that we obtain the representation $\|x\|_{\ell_0} = \sum_{i=1}^{N} |x_i|_0$. In Figure 2.1, we plot $|\cdot|_0$ together with $|\cdot|^p$ for $p \in \{1/3, 1/2, 1\}$ in the interval $[-1, 1]$. Obviously, for $0 < p \leq 1$, the function $|\cdot|^p$ is a continuous relaxation of $|\cdot|_0$, and it is even convex for $p = 1$. Thus, the ℓ_p-norm can be considered as an approximation of the ℓ_0-norm, and instead of solving the discontinuous and non-convex problem (2.2), we relax the problem by the *ℓ_p-norm minimization* problem

$$\Delta_p(y) := \operatorname*{arg\,min}_{z \in \mathcal{F}_\Phi(y)} \|z\|_{\ell_p}^p, \tag{2.3}$$

$0 < p \leq 1$. By solving the relaxed problem, we hope that $\Delta_p(y) \approx \Delta_0(y)$. The non-convex problem for $p < 1$ is still hard to solve, and one has to pay attention to

[1] "NP" is the abbreviation for non-deterministic polynomial-time and indicates a class of problems for which the verification of their solution has a computational cost which is polynomial in the size of the input. However presently it is not known whether such problems can be solved with a polynomial complexity algorithm. This issue is the first in the list of the *Millennium Prize Problems* of the Clay Mathematics Institute.

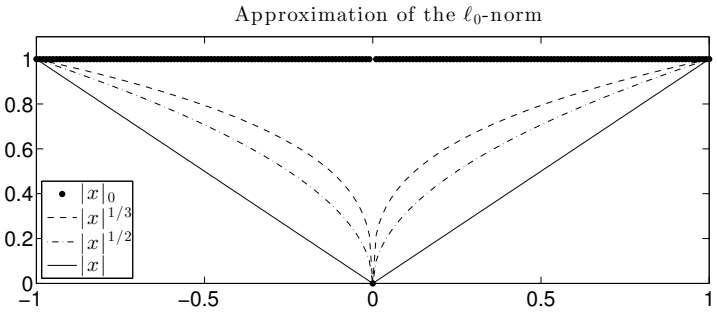


Figure 2.1: Plot of $|\cdot|_0$ in comparison to $|\cdot|^p$ for $p \in \{1/3, 1/2, 1\}$ in the interval $[-1, 1]$.

local minimizers, which makes this approach relatively unpopular. In contrast, for the particular case of $p = 1$, this problem becomes the well-known ℓ_1-*norm minimization* (ℓ_1-*minimization*) problem

$$\Delta_1(y) := \underset{z \in \mathcal{F}_\Phi(y)}{\arg\min} \|z\|_{\ell_1}. \tag{2.4}$$

It is probably the most studied case, due to its convexity, which allows to derive that indeed $\Delta_1(y) = \Delta_0(y)$ if the solutions are sparse enough, and the encoder Φ is fulfilling some spectral conditions (compare Lemma 2.12). In the following, we specify those spectral conditions of the encoder Φ, before commenting in detail on properties of the decoder Δ_p.

2.1.3 Encoder Properties

The following property of the encoder Φ turns out to be crucial.

Definition 2.5 (Null Space Property (NSP))
A matrix $\Phi \in \mathbb{R}^{m \times N}$ has the *Null Space Property* (NSP) of order k for positive constant $0 < \gamma_k < 1$, and fixed $0 < p \leq 1$, if

$$\|z|_\Lambda\|_{\ell_p}^p \leq \gamma_k \, \|z|_{\Lambda^c}\|_{\ell_p}^p \,,$$

for all $z \in \mathcal{N}_\Phi$ (see (1.2)), and all $\Lambda \subset \{1, \ldots, N\}$ such that $\#\Lambda \leq k$. We abbreviate this property with the writing (k, γ_k)-*NSP*. If we refer to it without explicitly mentioning the parameter p, we mean that we use it with $p = 1$.

The Null Space Property states that the kernel of the encoding matrix Φ contains no vectors where some entries have a significantly larger magnitude with respect to the others. In particular, no compressible vector is contained in the kernel. This is a

natural requirement since otherwise no decoder would be able to robustly distinguish a sparse vector from zero.

The NSP has the following stability result as consequence.

Lemma 2.6 ([48, Lemma 7.6], [84, Theorem 4.14])
Assume that $\Phi \in \mathbb{R}^{m\times N}$ satisfies the (k,γ_k)-NSP for $0 < p \leq 1$. Then for any vectors $z, z' \in \mathcal{F}_\Phi(y)$ it holds

$$\|z' - z\|_{\ell_p}^p \leq \frac{1+\gamma_k}{1-\gamma_k}\left(\|z'\|_{\ell_p}^p - \|z\|_{\ell_p}^p + 2\sigma_k(z)_{\ell_p}\right).$$

Unfortunately, the NSP is hard to verify in practice. Therefore one can introduce another property which is called the *Restricted Isometry Property*.

Definition 2.7 (Restricted Isometry Property (RIP))
A matrix $\Phi \in \mathbb{R}^{m\times N}$ has the *Restricted Isometry Property* (RIP) of order k with constant $0 < \delta_k < 1$ if

$$(1-\delta_k)\|z\|_{\ell_2} \leq \|\Phi z\|_{\ell_2} \leq (1+\delta_k)\|z\|_{\ell_2},$$

for all $z \in \Sigma_k$.[2] We refer to this property by the short writing (k,δ_k)-*RIP*.

According to [84, Theorem 6.8], the RIP constant of order k, δ_k, is bounded below by $c\sqrt{k/m}$, with constant c. The RIP implies the NSP, and is connected to it as follows.

Lemma 2.8
Let $k, h \in \mathbb{N}$ and $K = k + h$. Assume that $\Phi \in \mathbb{R}^{m\times N}$ has (K,δ_K)-RIP. Then Φ has (k,γ_k)-NSP, where

$$\gamma_k := \sqrt{\frac{k}{h}}\frac{1+\delta_K}{1-\delta_K}.$$

The proof of this lemma can be found, for instance, in [77]. The following result, which is also connecting the RIP and NSP, is used further below in Section 3.1.2. It originates from [50], where also a proof is given.

Lemma 2.9
Let $k \in \mathbb{N}$, and assume that $\Phi \in \mathbb{R}^{m\times N}$ has $(2k,\delta_{2k})$-RIP. Then Φ has $(2k,\gamma_{2k})$-NSP, where

$$\gamma_{2k} := \frac{\sqrt{2}\delta_{2k}}{1-(1+\sqrt{2})\delta_{2k}}.$$

[2] Very often, one can also find the definition $(1-\delta_k)\|z\|_{\ell_2}^2 \leq \|\Phi z\|_{\ell_2}^2 \leq (1+\delta_k)\|z\|_{\ell_2}^2$ in the literature. Thus one, has to be careful with constants during cross-reading.

Being a spectral concentration property, the Restricted Isometry Property is particularly suited to be verified with high probability by certain random matrices; we mention some instances of such classes in the next section.

The RIP implies that, for a any subset $\tilde{\Lambda} \subset \{1, \dots, N\}$ with $\#\tilde{\Lambda} \leq k$, the matrix Φ is injective on the subspace $\operatorname{span}\{e_i, i \in \tilde{\Lambda}\}$. This property can be generalized towards an infinite dimensional setting.

Definition 2.10 (Finite Basis Injectivity Property (FBI) [23, Definition 2.2]) Let $T\colon \ell_2(\Lambda) \to \mathcal{H}$ be an operator, mapping into a Hilbert space $\mathcal{H}$. Then T has the *Finite Basis Injectivity* (FBI) property if for all finite subsets $\tilde{\Lambda} \subset \Lambda$, the operator, restricted to $\operatorname{span}\{e_i | \, i \in \tilde{\Lambda}\}$, is injective, i.e., for all $u, v \in \ell_2(\Lambda)$ with $Tu = Tv$ and $u_i = v_i = 0$, $i \in \tilde{\Lambda}^c$, it follows that $u = v$.

This property was defined and used in order to show the linear convergence of iterative soft thresholding (see Section 2.4.3.1) [22]. It turns out to be useful for our convergence analysis in Section 3.2. Note that in contrast to the RIP, the FBI property does not make an assumption on involved constants.

2.1.4 Instance Optimality of Decoders

Being aware of the fundamental encoder properties that we explained above, we return to the ℓ_p-minimization decoder (2.3), and state a property of the decoder Δ_p, which is called *instance optimality* and is a direct consequence of Lemma 2.6.

Lemma 2.11 ([84, Theorem 4.12], [173])
Let $\Phi \in \mathbb{R}^{m \times N}$ have the (k, γ_k)-NSP for $0 < p \leq 1$. Then the decoder Δ_p performs

$$\|z - \Delta_p(y)\|_{\ell_p}^p \leq C \sigma_k(z)_{\ell_p}, \tag{2.5}$$

for all $z \in \mathcal{F}_\Phi(y)$ and the constant $C := \frac{2(1+\gamma_k)}{1-\gamma_k}$.

The instance optimality implies in particular that by the decoder Δ_p, we are able to recover a k-sparse signal x exactly since in this case $\sigma_k(x)_{\ell_p} = 0$.

Another consequence of Lemma 2.6 is the following result.

Lemma 2.12 ([48, Lemma 4.3])
Assume that Φ has the (k, γ_k)-NSP for $0 < p \leq 1$. Suppose that $\mathcal{F}_\Phi(y)$ contains a k-sparse vector x^*. Then this vector is the unique ℓ_p-minimizer in $\mathcal{F}_\Phi(y)$. Moreover we have for all $z \in \mathcal{F}_\Phi(y)$ that

$$\|z - x^*\|_{\ell_p}^p \leq 2 \frac{1 + \gamma_k}{1 - \gamma_k} \sigma_k(z)_{\ell_p}.$$

So far we showed instance optimality for the Δ_p decoder and an encoder which fulfills the NSP for some k, but we did not clarify whether there exists such an encoder. Gaussian random matrices satisfy the RIP of order k with high probability if

$$m \geq Ck \log(N/k). \tag{2.6}$$

Also structured random matrices, i.a., random partial Fourier and discrete cosine matrices, and partial random circulant matrices satisfy the RIP with high probability provided that $m \geq Ck \log^4(N)$ [30, 84, 118, 159, 160]. Since the RIP implies the NSP (see, e.g., Lemma 2.8 and 2.9), we have the existence of encoder/decoder pairs that satisfy the instance optimality. Within this thesis we use as prototypical cases mainly such stochastic encoders. By the concept of Gelfand-widths (see also [88, 90, 116]), it was shown in [12, 41, 57] that the bound on the number of measurements (2.6) is optimal, in the sense that an encoder/decoder pair (Φ, Δ) satisfying the instance optimality cannot exist for m below this bound.

2.1.5 Non-Standard Bases

Up to this point, we assumed x to be sparse with respect to the canonical basis of $\mathbb{R}^N$, and being directly measured by means of the linear encoder Φ. However, in practice very often, a basis transformation $\Psi \in \mathbb{R}^{N \times N}$ might be necessary to find a proper sparse representation $\tilde{x}$ of the vector x, i.e., $x = \Psi\tilde{x}$. By defining $\tilde{\Phi} := \Phi\Psi$, we obtain the linear measurement process

$$y = \tilde{\Phi}\tilde{x},$$

which is again of the type (2.1) with $\tilde{\Phi}$ having full rank and $\tilde{x}$ being sparse. Such aspects are important for applications of sparse recovery, and they are in particular a modeling issue, but not relevant in this thesis. Since we focus on algorithmic solutions for sparse recovery problems, we assume without loss of generality that the solutions of interest are sparse in the canonical basis.

2.2 Noise Models

So far, we were only considering the noiseless acquisition model (2.1). Unfortunately, it only serves as a rough description of real physical processes, which are usually corrupted by disturbances on the measurement data and the signal. In the following, we introduce more sophisticated models and sparse recovery decoders that take into account such perturbations.

2.2.1 Measurement Noise and Model Error

In applications of the industry or scientific experiments the measurement data is in general affected by disturbances that originate from the environment or the device

itself. Also a simplified description of a measurement process, for instance from a linearization, can lead to a model error. Both effects are typically modeled by

$$y = \Phi x + e, \tag{2.7}$$

where an additional deterministic or random noise vector $e \in \mathbb{R}^m$ corrupts the linear measurements. Regarding this modification of the noiseless model (2.1), an enhanced stability property of the ℓ_1-minimization decoder Δ_1 is for instance established in [28]. In the following we state this result, preceded by an auxiliary lemma.

Lemma 2.13
For any x, e we denote $x^{[k]} := x - x_{[k]}$, and $\tilde{e} := \Phi x^{[k]} + e$, where $x_{[k]}$ is the best k-term approximation to x. Let

$$y = \Phi x + e = \Phi x_{[k]} + \Phi x^{[k]} + e = \Phi x_{[k]} + \tilde{e}.$$

If Φ has the (k, δ_k)-RIP, then the norm of the error $\tilde{e}$ can be bounded by

$$\|\tilde{e}\|_{\ell_2} \leq (1 + \delta_k) \left(\sigma_k(x)_{\ell_2} + \frac{\sigma_k(x)_{\ell_1}}{k^{1/2}} \right) + \|e\|_{\ell_2}.$$

Proof. By definition we have $\tilde{e} = \Phi x^{[k]} + e$. To compute the norm of the error term, we simply apply the triangle inequality and use [142, Proposition 3.5], which states that

$$\|\Phi z\|_{\ell_2} \leq (1 + \delta_k) \left(\|z\|_{\ell_2} + \frac{\|z\|_{\ell_1}}{k^{1/2}} \right),$$

for an arbitrary vector $z \in \mathbb{R}^m$, and in particular for $z = x^{[k]}$. □

Theorem 2.14
Let $\Phi \in \mathbb{R}^{m \times N}$ satisfy the $(2k, \delta_{2k})$-RIP with constant $\delta_{2k} > 0$ sufficiently small. Assume further that $y = \Phi x + e$ where e is a measurement error, and that $x^* = \Delta_1(y)$ is k-sparse. Then the decoder Δ_1 has the further enhanced stability property

$$\|x - \Delta_1(y)\|_{\ell_2} \leq C \left(\sigma_k(x)_{\ell_2} + \frac{\sigma_k(x)_{\ell_1}}{k^{1/2}} + \|e\|_{\ell_2} \right). \tag{2.8}$$

Proof. Let us consider the following estimate, which follows by an application of the lower bound of the RIP and Lemma 2.13:

$$\begin{aligned} \|x - \Delta_1(y)\|_{\ell_2} &\leq \|x^{[k]}\|_{\ell_2} + \|x_{[k]} - x^*\|_{\ell_2} \leq \sigma_k(x)_{\ell_2} + \frac{1}{1 - \delta_{2k}} \|\Phi x_{[k]} - \Phi x^*\|_{\ell_2} \\ &= \sigma_k(x)_{\ell_2} + \frac{1}{1 - \delta_{2k}} \|\Phi x_{[k]} - y\|_{\ell_2} = \sigma_k(x)_{\ell_2} + \frac{1}{1 - \delta_{2k}} \|\tilde{e}\|_{\ell_2} \\ &\leq C \left(\sigma_k(x)_{\ell_2} + \frac{\sigma_k(x)_{\ell_1}}{k^{1/2}} + \|e\|_{\ell_2} \right). \end{aligned}$$

□

There is a vast literature where problem (2.7) is considered, and alternative decoders to ℓ_1-minimization were proposed. The most intuitive approach to cover noisy measurements is the *quadratically constrained ℓ_1-minimization* or *Basis Pursuit Denoising* (BPDN) problem

$$\Delta_{\mathrm{DN}}(y) := \underset{\|\Phi z - y\|_{\ell_2} \leq \delta}{\arg\min} \|z\|_{\ell_1}, \tag{2.9}$$

where one has to tune the parameter $\delta \in \mathbb{R}$, such that $0 < \delta \approx \|\Phi x - y\|_{\ell_2} = \|e\|_{\ell_2}$. Notice that the notation δ without any subindex denotes an inequality bound parameter, while δ_k, δ_{2k}, etc. represent RIP constants. Let us mention that in the literature the standard ℓ_1-minimization decoding process Δ_1 is also referred to as *Basis Pursuit* (BP), and it is the special case of (2.9) for $\delta = 0$. Another approach is the so called *Least Absolute Shrinkage and Selection Operator* (LASSO) [177],

$$\Delta_{\mathrm{LA}}(y) := \underset{\|z\|_{\ell_1} \leq \epsilon}{\arg\min} \|\Phi z - y\|_{\ell_2}, \tag{2.10}$$

for a positive parameter $\epsilon \in \mathbb{R}$. It can be seen as a convexification of the more intuitive problem formulation

$$\underset{\|z\|_{\ell_0} \leq k}{\arg\min} \|\Phi z - y\|_{\ell_2}, \tag{2.11}$$

where the sparsity of the signal to be recovered is restricted by the positive parameter $k \in \mathbb{N}$. It is also common to include the constraint by a regularization term in the objective function. A popular formulation is the *ℓ_1-regularized least squares problem*

$$\Delta_\lambda(y) := \underset{z \in \mathbb{R}^N}{\arg\min} \left(\mathcal{J}_\lambda(z) := \lambda \|z\|_{\ell_1} + \|\Phi z - y\|_{\ell_2}^2 \right), \tag{2.12}$$

which again can be considered as a convexification of the *ℓ_0-regularized least squares problem*

$$\Delta_{0,\lambda}(y) := \underset{z \in \mathbb{R}^N}{\arg\min} \left(\mathcal{J}_0(z) := \lambda \|z\|_{\ell_0} + \|\Phi z - y\|_{\ell_2}^2 \right), \tag{2.13}$$

where the regularization parameter $\lambda > 0$ controls the balance of the fidelity and penalty term. In the literature one can also find problem (2.12) referred to as LASSO or BPDN. The reason is that problems (2.9), (2.10), and (2.12) are equivalent in the sense that, for a given parameter δ, a solution to (2.9) is either zero or a minimizer of (2.12) for some $\lambda > 0$, and a solution of (2.10) for any $\epsilon \geq 0$ is also a minimizer for (2.12) for some $\lambda \geq 0$ [72].

Those three decoders are used with preference since they are convex problems and therefore relatively easy to solve. But also non-convex approaches are considered, as, e.g., the ℓ_p-minimization (2.3) or the *ℓ_p-regularized least squares problem*

$$\Delta_{p,\lambda}(y) := \underset{z \in \mathbb{R}^N}{\arg\min} \left(\mathcal{J}_p(z) := \lambda \|z\|_{\ell_p}^p + \|\Phi z - y\|_{\ell_2}^2 \right), \tag{2.14}$$

$0 < p < 1$. In non-convex approaches, one has to deal with local minimizers, which may lower the reliability of those methods. However, as we also show in Chapter 3, in practice the use of non-convex decoders may result in a better reconstruction of the original vector x, when strong noise is present (in particular strong signal noise as explained in the next section). Furthermore, a performance increase with respect to the rate of convergence was reported in algorithmic schemes based on non-convex optimization problems, e.g., in iteratively re-weighted least squares algorithms for ℓ_p-minimization ($p < 1$); compare Section 2.4.1. The reason behind this enhanced robustness and performance improvement is roughly that such non-convex methods can be a much better approximation of the ℓ_0-penalty with respect to the ℓ_1-penalty (compare Figure 2.1).

In this thesis we also consider the generalization of (2.12) towards an infinite dimensional setting. Suppose that we dispose of measurement data y, given as an element of a Hilbert space $\mathcal{H}$, and let $T\colon \ell_1(\Lambda) \to \mathcal{H}$ be a bounded linear operator. Then we want to reconstruct a potentially sparse signal $u \in \ell_1(\Lambda)$ from y by the decoder

$$\Delta_\lambda(y) := \operatorname*{arg\,min}_{u \in \ell_1(\Lambda)} \left(\mathcal{J}_\lambda(u) := \|Tu - y\|_{\mathcal{H}}^2 + \lambda \|u\|_{\ell_1(\Lambda)} \right). \tag{2.15}$$

Note that we refer by the notation $\mathcal{J}_\lambda$ and Δ_λ to the problem and functional in both (2.12) and (2.15). It will be clear from the context whether we mean the specific finite dimensional version or the infinite dimensional one.

For any of the above presented decoders, the recovery result depends in general on the choice of the parameter ϵ, δ, k, or λ respectively. A concrete example for the choice of δ is given in [33, 28]. In this paper the authors propose to set $\delta = \sigma\sqrt{m + 2\sqrt{2m}}$, where σ is the standard deviation of a white noise vector $e \in \mathbb{R}^m$. A parameter can take different "optimal" values for different purposes, e.g., in our numerical investigation in Section 3.3, we observed that for BPDN there is a difference between targeting an exact recovery of the support of a signal and targeting a good ℓ_2-approximation error of the amplitudes of the original and recovered signal. When data is available where the ground truth is known, it is also common practice to determine optimal parameters by the training of the algorithm (e.g., [205]).

2.2.2 First Order Optimality Conditions of the ℓ_1-regularized Least Squares Functional

In the following, we derive the first order optimality conditions for the functional $\mathcal{J}_\lambda$, which turns out to be a helpful tool in some of the analysis parts of this thesis. We directly present it for its infinite dimensional formulation (2.15), since we will also utilize it later on. The minimizer of $\mathcal{J}_\lambda$ can be characterized using the *subdifferential*

[66], which is defined for a general convex function $F\colon \mathcal{K} \to \mathbb{R}$ at a point u in the Banach space $\mathcal{K}$ by

$$\partial F(u) = \{v \in \mathcal{K}^*, F(z) - F(u) \geq \langle v, z - u\rangle \text{ for all } z \in \mathcal{K}\}, \tag{2.16}$$

with $\mathcal{K}^*$ being the dual space of $\mathcal{K}$. Clearly, u is a minimizer of F if and only if $0 \in \partial F(u)$. For $\mathcal{K} = \ell_1(\Lambda)$, the subdifferential of $\mathcal{J}_\lambda$ is given by

$$\partial \mathcal{J}_\lambda(u) = 2T^*(Tu - y) + \lambda \partial \|\cdot\|_{\ell_1(\Lambda)}(u).$$

We remind that the notation "$\|\cdot\|_{\ell_1(\Lambda)}$" is actually a simplified notation for the $\ell_{1,2}(\Lambda)$-norm, which is defined by $\|u\|_{\ell_{1,2}(\Lambda)} := \sum_{i\in\Lambda} \|u_i\|_{\ell_2}$ with $\|\cdot\|_{\ell_2}$ being the standard vector norm in $\mathbb{R}^d$ (see Section 1.2). Thus, the subdifferential of the $\ell_1(\Lambda)$-norm is given by

$$\partial \|\cdot\|_{\ell_1(\Lambda)}(u) = \{v \in \mathcal{K}^* | \, v_i \in \partial \|\cdot\|_{\ell_2}(u_i), i \in \Lambda\},$$

with the subdifferential of the ℓ_2-norm in $z \in \mathbb{R}^d$ being

$$\partial \|\cdot\|_{\ell_2}(z) = \begin{cases} \left\{\frac{z}{\|z\|_{\ell_2}}\right\} & \text{if } z \neq 0, \\ \{v | \|v\|_{\ell_2} \leq 1\} & \text{if } z = 0. \end{cases}$$

Then the inclusion $0 \in \partial \mathcal{J}_\lambda(u)$ is equivalent to

$$\begin{aligned} -2(T^*(Tu - y))_i &= \lambda \frac{u_i}{\|u_i\|_{\ell_2}} && \text{if } u_i \neq 0, \\ 2\|(T^*(Tu - y))_i\|_{\ell_2} &\leq \lambda && \text{if } u_i = 0, \end{aligned} \tag{2.17}$$

for all $i \in \Lambda$. The conditions (2.17) are referred to as the *first order optimality condtions for the ℓ_1-regularized least squares functional*.

If we only consider the finite dimensional problem with the decoder (2.12), and additionally set $d = 1$, the conditions (2.17) reduce to

$$\begin{aligned} -2(\Phi^*(\Phi x - y))_i &= \lambda \operatorname{sign}(x_i) && \text{if } x_i \neq 0, \\ 2|(\Phi^*(\Phi x - y))_i| &\leq \lambda && \text{if } x_i = 0, \end{aligned} \quad i = 1, \ldots, N, \tag{2.18}$$

for $x \in \mathbb{R}^N$.

The following properties, which are standard results in regularization theory (see, e.g., [69]), can be derived from (2.17).

Lemma 2.15
Denote u_λ a solution to problem (2.15) for $\lambda > 0$, then $u_0 := \lim_{\lambda \to 0} u_\lambda$ is a solution of the ℓ_1-minimization problem (see (2.4) in finite dimension)

$$\arg\min_{u \in \mathcal{F}_T(y)} \|u\|_{\ell_1(\Lambda)}. \tag{2.19}$$

Lemma 2.16
If $\lambda > 2\|T^* y\|_{\ell_\infty(\Lambda)}$, then 0 is the only solution to problem (2.15) (with $\mathcal{H} = \ell_2(\Lambda)$).

A proof of Lemma 2.15 and 2.16 can be found in Appendix A. Lemma 2.15 means that the solutions to problem (2.15) converge to a limit that is characterized by the sparsest representation which exactly fits the data. Lemma 2.16 states that the solution to the optimization problem (2.15) is constant for all $\lambda \in \left]2\|T^* y\|_{\ell_\infty(\Lambda)}, \infty\right[$. It implicates an upper bound for the choice of λ.

2.2.3 Signal Noise and Noise Folding

In Section 2.2.1, we recalled a model which takes corrupted measurements into account, and recalled several methods to solve it. However, in practice it is very uncommon to have the signal x detected by a certain device, totally free from some external noise. In this case it is reasonable to consider the more realistic model

$$y = \Phi(x + n) + e, \tag{2.20}$$

instead of (2.7), where $x \in \mathbb{R}^N$ is the noiseless signal, and $n \in \mathbb{R}^N$ is the noise on the original signal. Obviously, by defining $\tilde{e} := \Phi n + e$, this model is reduced to $y = \Phi x + \tilde{e}$, and we could again consider the situation in Section 2.2.1. However, assume that n is white noise, i.e., $n \sim (\mathcal{N}(0, \sigma_n))^N$ with standard deviation σ_n[3]. Then, in the recent work [6, 178, 51] it was shown how the measurement process actually causes the *noise-folding phenomenon*. It implies that the variance of the noise on the original signal is amplified by a factor of N/m, additionally contributing to the measurement noise, playing to our disadvantage in the recovery phase. Thus, the signal noise n is a worse enemy towards accurate reconstructions than the measurement noise e. In the following we want to describe those results in more detail. In order to focus on the influence of the signal noise n, we will from now on consider the model

$$y = \Phi(x + n). \tag{2.21}$$

If n is white noise, it means that Φn is in general not white and has covariance $C_{\Phi n} = \sigma_n^2 \Phi\Phi^*$. Multiplying the linear system (2.21) from the left by the matrix

[3]The notation σ_n for the standard deviation shall not be confused with the notation $\sigma_k(x)_{\ell_p}$ for the best k-term approximation error.

$M := \left(\frac{m}{N} C_{\Phi n}\right)^{-\frac{1}{2}}$ transforms it into the system

$$\bar{y} = \bar{\Phi} x + \bar{e},$$

where $\bar{y} := My$, $\bar{\Phi} = M\Phi$, and $\bar{e} := M\Phi n$. We actually performed a *whitening* of Φn, so that $\bar{e} \sim \left(\mathcal{N}(0, \sqrt{\frac{N}{m}}\sigma_n)\right)^m$. If one disposes of σ_n, it is a common procedure (see, e.g., [204]) to enforce such a *prewhitening* of the data, before applying a decoder, such as BP(DN), LASSO, or the ℓ_1-regularized least squares. The resulting methods are called the prewhitened basis pursuit (denoising) (PWBP(DN)),

$$\underset{\|M(\Phi z - y)\|_{\ell_2} \leq \delta}{\arg\min} \|z\|_{\ell_1}, \tag{2.22}$$

the prewhitened LASSO (PWLASSO),

$$\underset{\|z\|_{\ell_1} \leq \epsilon}{\arg\min} \|M(\Phi z - y)\|_{\ell_2},$$

or the prewhitened ℓ_1-regularized least squares minimization,

$$\underset{z \in \mathbb{R}^N}{\arg\min} \lambda \|z\|_{\ell_1} + \|M(\Phi z - y)\|_{\ell_2}^2,$$

with M as defined above. We take into account PWBPDN in the numerical comparisons of Section 3.3.

In the following lemma, which we recall from [6], it is shown that the transformed measurement matrix $\bar{\Phi}$ is of equal statistics as Φ.

Lemma 2.17 ([6, Proposition 1])
Assume that $\kappa := \left\|I - \frac{m}{N}\Phi\Phi^*\right\| < \frac{1}{2}$ *and that* Φ *satisfies the* (k, δ_k)*-RIP. Then* $\bar{\Phi}$ *satisfies the* $(k, \bar{\delta}_k)$*-RIP with constant* $\bar{\delta}_k := \max\{1 - (1 - \delta_k)\sqrt{1 - \kappa_1}, (1 + \delta_k)\sqrt{1 + \kappa_1} - 1\}$*, and* $\kappa_1 := \kappa/(1 - \kappa)$*.*

Remark 2.18
Note that the assumption $\left\|I - \frac{m}{N}\Phi\Phi^*\right\| < \frac{1}{2}$ can be fulfilled with high probability in the standard setting of compressed sensing, e.g., if the entries of Φ are i.i.d. Gaussian (compare [185, Corollary 35, Theorem 39]).

Thus, we conclude that the linear measurement process (2.21), which is corrupted by white signal noise with entries of standard deviation σ_n, can be considered equivalent to a linear measurement process of the form (2.7). This equivalent measurement process involves a matrix whose RIP constant is close to the one of the original process, and a measurement noise vector with entries of standard deviation $\sqrt{\frac{N}{m}}\sigma_n$. In this sense, we talk about the *amplification* of the variance of the signal noise by a factor of N/m.

In the same stochastic context, the so-called Dantzig selector has been analyzed in [31] showing that the recovered signal x^* from the measurement $y = \Phi x + e$, $e \sim (\mathcal{N}(0, \sigma_e))^m$, fulfills with high probability the following nearly-optimal distortion guarantees, under the assumption that Φ satisfies the RIP:

$$\|x - x^*\|_{\ell_2}^2 \leq C^2 \cdot 2 \log N \cdot \left(\sigma_e^2 + \sum_{i=1}^{N} \min\{x_i^2, \sigma_e^2\} \right),$$

which, for a sparse vector x with at most k-nonzero entries and the substitution $\sigma_e^2 = \frac{N}{m}\sigma_n^2$, reduces to the following estimate

$$\|x - x^*\|_{\ell_2}^2 \leq C^2 \cdot 2 \log N \cdot \left((1+k)\sigma_e^2 \right) = C^2 \cdot 2 \log N \cdot \left((1+k)\frac{N}{m}\sigma_n^2 \right). \tag{2.23}$$

Thus, we observe that the squared error between the decoded signal x^* and the original signal x is influenced by the factor N/m.

In [178, 51] the authors describe the noise folding phenomenon, i.e., the exaltation of the signal noise after measurements, following a different reasoning: Assume that there is an oracle[4] that provides us with the support of the sparse signal $\Lambda = \mathrm{supp}(x)$. Then a natural recovery strategy is

$$\underset{\mathrm{supp}(z)=\Lambda}{\arg\min} \ \|\Phi z - y\|_{\ell_2}. \tag{2.24}$$

Theorem 2.19 ([51, Theorem 4.3])
Let x^* be the solution to problem (2.24) (assume Φ to have full rank). Suppose n to be white noise, and Φ to satisfy the (k, δ_k)-RIP, and to have orthogonal rows, each of norm $\sqrt{\frac{N}{m}}$. Then the expected value of an error estimate for x^* is given by

$$\frac{N}{m}(1+\delta_k)^{-2}\mathbb{E}(\|n_\Lambda\|_{\ell_2}^2) \leq \mathbb{E}(\|x - x^*\|_{\ell_2}^2) \leq \frac{N}{m}(1-\delta_k)^{-2}\mathbb{E}(\|n_\Lambda\|_{\ell_2}^2). \tag{2.25}$$

Remark 2.20
The condition that Φ consists of orthogonal rows of equal norm is not restrictive in the setting of compressed sensing since for any arbitrary matrix $\tilde{\Phi}$ which satisfies the RIP, it is always possible to construct a matrix Φ that has the same row space as $\tilde{\Phi}$ and does satisfy this property (compare [51, Lemma 4.1]).

[4] The word "oracle" is used in the respective paper for a predictive method from which we do not know if it exists.

The estimate (2.25) and (2.23) are both leading to the same result that the squared ℓ_2-norm error of the recovered signal with respect to the original signal is proportional to N/m times the variance of the noise. In a noise-free setting the decoder (2.24) exactly recovers x if Φ_Λ is full rank. The actual challenge in the noise-folding regime is to identify a decoder which "simulates" the oracle, i.e., which robustly and reliably determines the support Λ.

In Chapter 3, we pursue the latter observation, and focus on the design and analysis of proper decoders, whose strength is the correct detection of the index support of the original vector x. Once we obtain the support, we implement (2.24) on the identified entries.

2.3 Joint Sparsity

So far, we did not properly motivate the use of $d > 1$ in Section 1.2. Assume that we have given an encoder $\Phi \in \mathbb{R}^m$ and d measurements $y^1, \dots, y^d \in \mathbb{R}^m$. We ask for the d (sparse) solution vectors $x^1, \dots, x^d \in \mathbb{R}^N$, which can be computed independently by one of the decoders, presented in the previous two sections. Following this procedure, we may obtain very different sparsity patterns in the d solution vectors. However, in some applications (and we present one in Section 4.2.6) additional a priori knowledge on the support of $x^1, \dots, x^d$ is given. For instance, *joint sparsity* may be required, i.e., the sparsity pattern of the d solution vectors is supposed to be similar, which means that $\operatorname{supp}(x^1) \approx \operatorname{supp}(x^2) \approx \dots \approx \operatorname{supp}(x^d)$. If we combine the measurement (column) vectors in the variable $y := (y^1, \dots, y^d) \in \mathbb{R}^{m \times d}$, and search for solutions $x := (x^1, \dots, x^d) \in \mathbb{R}^{N \times d}$, joint sparsity can be enforced by the penalty norms $\|\cdot\|_{\ell_{p,2}}$, for $0 \leq p \leq 1$. Therefore, all decoders that we presented in Section 2.1 and 2.2 can be extended to solve the joint-sparsity problem, by replacing the ℓ_p- and ℓ_2-penalties by $\ell_{p,2}$- and $\ell_{2,2}$-penalties respectively. However, we do not have to redefine those problems since this generalization complies with the notation (1.1), which incorporates standard models with $d = 1$, and joint sparsity models with $d > 1$. For instance the formulation $\lambda\|z\|_{\ell_1} + \frac{1}{2}\|\Phi z - y\|_{\ell_2}^2$ is defined for vectors $z \in \mathbb{R}^N$ (standard sparsity) or $z \in \mathbb{R}^{N \times d}$ for $d > 1$ (joint-sparsity).

The concept of joint sparsity can be generalized further, e.g., with different measurement processes, i.e., we consider d measurements $y^1, \dots, y^d \in \mathbb{R}^m$, D solution vectors $x^1, \dots, x^D \in \mathbb{R}^N$, and encoders $\Phi^{i,j} \in \mathbb{R}^{m \times N}$, $i = 1, \dots, D$, $j = 1, \dots, d$, where the measurement acquisition process is defined by $y^j = \sum_{i=1}^D \Phi^{i,j} x^i$, $j = 1, \dots, d$ (see, e.g., [78]). Since we do not use such a generalization, we do not go into further detail. In the literature the joint-sparsity concept is also described by the terms *group sparsity* (e.g., [85]) and *block sparsity* (e.g., [67]). In particular for $d > 1$, the generalized problems (2.12) and (2.15) are referred to as the *group LASSO*. The optimality conditions in (2.17) are also valid for the group LASSO since we already derived them for $d \geq 1$.

2.4 Algorithms for Sparse Recovery

We recalled in Section 2.1 and 2.2 that sparse recovery problems can be approached by convex (involving the ℓ_1-norm) and non-convex (involving the ℓ_p-norm for $0 \leq p < 1$) optimization problems. In order to apply sparse recovery in practice, efficient algorithms are required that either solve those optimization problems exactly, or compute an approximation which fulfills similar recovery guarantees such as instance optimality (2.5).

A first precise approach in order to solve the ℓ_1-minimization problem (2.4) is its transformation into the equivalent linear program

$$\underset{\tilde{x}\in\mathbb{R}^{2N}}{\arg\min} \sum_{j=1}^{2N} \tilde{x}_j \quad \text{subject to} \quad \tilde{x} \geq 0,\ \begin{pmatrix}\Phi & -\Phi\end{pmatrix} \tilde{x} = y, \tag{2.26}$$

assuming a real decoding matrix $\Phi \in \mathbb{R}^{m\times N}$ and data $y \in \mathbb{R}^m$. The solution x^* to (2.4) is obtained from the solution $\tilde{x}^*$ of (2.26) via $x^* = \begin{pmatrix}I & -I\end{pmatrix} \tilde{x}^*$, for I the identity matrix. Thus, in principle any linear programming method may be used for solving (2.4), in particular the simplex method and interior point methods [145].

However, instead of optimizing a vector of length N, we have to solve a larger optimization problem for a vector of the length $2N$ since a transformation of the non-smooth ℓ_1-norm into a smooth objective function was necessary. Indeed most intuitive optimization methods such as gradient descent or Newton methods are tailored to objective functions which are at least differentiable, and cannot be applied directly to problem formulations that involve non-smooth terms such as the ℓ_1-norm. Furthermore, standard software very rarely provides the possibility of a quick and easy tuning, in the sense that only full matrices are accepted instead of fast routines for matrix-vector multiplications as for instance in the case of partial Fourier matrices. Based on those observations, relatively simple alternative methods could be found that are tailored to problem formulations involving non-smooth (and maybe non-convex) objective functions and possible prior information on the sparsity of the solution (e.g., the number of non-zero support entries). In particular the scientific community from mathematics and engineering contributed various specialized algorithms for sparse recovery that are expected to outperform standard methods. Most of those tailored methods are based on different assumptions, which makes it difficult to compare them or to identify "the best" one. At the internet sources [34, 62] one can find a relatively updated and complete collection of relevant algorithms for the field.

In this section, we focus on a detailed explanation of the basic concepts of some of the most popular algorithms. The explanations serve in particular as a fundament for the presented work in Chapter 3 and 4. Specifically, we recall the efficient and easy-to-implement algorithms *iteratively re-weighted least squares* (IRLS), *iteratively re-weighted* ℓ_1 (IRL1), and *iterative thresholding* for the approximate or exact solution

of the convex problems (2.4) (IRLS), (2.12) (IRLS, iterative thresholding) and the non-convex problems (2.13), (2.14) (iterative thresholding, IRL1), (2.2), (2.3) (IRLS).

Without going further, we mention some other quite popular methods. In particular for high sparsity (very few non-zero entries), greedy methods like *orthogonal matching pursuit* (OMP) [179], and *compressive sampling matching pursuit* (CoSaMP) [142], as well as the *homotopy method* or *modified LARS* [59, 63, 150, 151], are suitable. The *Chambolle-Pock primal dual algorithm* [35] can be applied to ℓ_1-minimization, and its performance is not depending on the sparsity of the solution. Another method for solving (2.12) is the *alternating direction method of multipliers* (ADMM) [87, 89, 1, 42, 70]. For further reading, we propose the detailed and widespread overview literature in [84, 75, 81].

2.4.1 Iteratively Re-weighted Least Squares (IRLS)

Iteratively re-weighted least squares (IRLS) is a method for solving minimization problems by transforming them into a sequence of quadratic problems which can be solved by efficient tools of numerical linear algebra. Thus, contrary to classical Newton methods smoothness of the objective function is not required in general. We refer to the recent paper [148] for an updated and rather general view about these methods.

An IRLS algorithm appeared for the first time in the doctoral thesis of Lawson in 1961 [123] in the form of an algorithm for solving uniform approximation problems. It computes a sequence of polynomials that minimize a sequence of weighted L_p-norms. This iterative algorithm is now well-known in classical approximation theory as Lawson's algorithm. In [39] it is proved that this algorithm essentially obeys a linear convergence rate. In the 1970s extensions of Lawson's algorithm for ℓ_p-norm minimization, and in particular ℓ_1-norm minimization, were proposed. Since then IRLS has become a rather popular method also in mathematical statistics for robust linear regression [110]. Perhaps the most comprehensive mathematical analysis of the performance of IRLS for ℓ_p-norm minimization was given in the work of Osborne [149]. After the starting of the development of compressed sensing, several works [36, 38, 37, 48] addressed systematically the analysis of IRLS for ℓ_p-norm minimization (2.3), with $0 < p \leq 1$ (including the ℓ_1-minimization problem (2.4)). In these papers, the asymptotic super-linear convergence of IRLS towards ℓ_p-norm minimization for $p < 1$ has been shown. As an extension of the analysis of the aforementioned papers, IRLS also has been generalized towards low-rank matrix recovery from minimal linear measurements [83].

In the following we recall the fundamental IRLS concept and how it is used to solve sparse recovery problems. In Section 2.4.1.1, we present an IRLS algorithm for the

solution of (2.3), originating from the work [48]. In addition to it, we comment in Section 2.4.1.2 on a numerical issue which appears in this algorithm in an advanced state of propagation and we propose a practical stopping criterion. We further explain in Section 2.4.1.3 how the IRLS scheme is applied to the class of problems of the regularized ℓ_p-minimization (2.14), for $0 < p \leq 1$, which was first proposed in [120, 189, 190].

2.4.1.1 IRLS Method for ℓ_p-minimization

The most important advantage of the IRLS scheme may be its simplicity and intuitive derivation, which shall be outlined here. Regarding the ℓ_p-norm ($0 < p \leq 1$), one can rewrite

$$\|z\|_{\ell_p}^p = \sum_{i=1}^N |z_i|^p = \sum_{i=1}^N |z_i|^{p-2} z_i^2,$$

and thus, intending to solve the problem (2.3), one would hope that

$$\operatorname*{arg\,min}_{z \in \mathcal{F}_\Phi(y)} \|z\|_{\ell_p}^p = \operatorname*{arg\,min}_{z \in \mathcal{F}_\Phi(y)} \sum_{i=1}^N |x_i^*|^{p-2} z_i^2,$$

if x^* is the ℓ_p-minimizer, and $x_i^* \neq 0$, $i = 1, \ldots, N$. At least for $p = 1$ this was shown to be true in [48, Equation (1.4) and footnote 1], and [75, Lemma 3.3]. This well-known linearly constrained quadratic problem can be solved by standard linear algebra, in contrast to the more complicated non-smooth (and maybe non-convex) ℓ_p-norm minimization problem. However, the above observation is obviously unpractical since one does not dispose of the minimizer x^* (which is actually the goal of the computations). Another drawback is that x^* is assumed to have no vanishing coordinates, which stands in contrast with the fact that we are interested in computing a sparse minimizer. Despite those objections, in view of the above observation we motivate the IRLS algorithm for ℓ_p-norm minimization as follows: We assume, that we have a good approximation x^n of the sparse minimizer and define a weight vector w^n, with entries $w_i^n := [|x_i^n|^2 + (\varepsilon^n)^2]^{-\frac{2-p}{2}}$, with a small ε^n in order to regularize vanishing entries in x^n. Then, we can solve the problem

$$\operatorname*{arg\,min}_{z \in \mathcal{F}_\Phi(y)} \sum_{i=1}^N w_i^n z_i^2,$$

in order to obtain a new iterate x^{n+1}. Eventually we update ε^{n+1} depending on x^{n+1} and repeat the iteration as presented above. Letting $\varepsilon^n \to 0$, one hopes for the convergence of the algorithm to a solution of (2.3).

We can alternatively formulate this so concisely described algorithm as an alternating minimization of the following multivariate functional.

Definition 2.21
Given a real number $\varepsilon > 0$, $x \in \mathbb{R}^N$, and a weight vector $w \in \mathbb{R}^N$ with positive entries $w_j > 0$, $j = 1, \dots, N$, we define

$$J_p(x, w, \varepsilon) := \frac{p}{2} \left[\sum_{j=1}^{N} |x_j|^2 w_j + \sum_{j=1}^{N} \left(\varepsilon^2 w_j + \frac{2-p}{p} w_j^{-\frac{p}{2-p}} \right) \right].$$

Furthermore, we denote the *weighted ℓ_2-norm* and the *weighted scalar product* by

$$\|x\|_{\ell_2(w)} := \sum_{j=1}^{N} |x_j|^2 w_j, \quad \langle x, z \rangle_w := \sum_{j=1}^{N} x_j z_j w_j.$$

The notation $\|\cdot\|_{\ell_2(w)}$, which is defined for a positive weight[5] $w \in \mathbb{R}^N$ should not be confused with the definition $\|\cdot\|_{\ell_2(\Lambda)}$ for a countable set Λ, as given in Section 1.2.

We formulate IRLS in Algorithm 1 as defined in [48, Section 7.2], or [84, Chapter 15.3].

Algorithm 1 Iteratively Re-weighted Least Squares (IRLS)

Set $w^0 := (1, \dots, 1)$, $\varepsilon^0 := 1$

1: **while** $\varepsilon^n \neq 0$ **do**
2: $x^{n+1} := \operatorname*{arg\,min}_{x \in \mathcal{F}_\Phi(y)} J_p(x, w^n, \varepsilon^n) = \operatorname*{arg\,min}_{x \in \mathcal{F}_\Phi(y)} \|x\|_{\ell_2(w^n)}$
3: $\varepsilon^{n+1} := \min\left(\varepsilon^n, \frac{r(x^{n+1})_{K+1}}{N}\right)$
4: $w^{n+1} := \operatorname*{arg\,min}_{w>0} J_p(x^{n+1}, w, \varepsilon^{n+1})$,
i.e., $w_j^{n+1} = (|x_j^{n+1}|^2 + (\varepsilon^{n+1})^2)^{-\frac{2-p}{2}}$, $j = 1, \dots, N$
5: **end while**

The parameter $K \in \mathbb{N}$ has to be set bigger than $k = \#\operatorname{supp}(x^*)$. Its role is clarified in more detail in [48] or in Section 4.1.2 (in particular Theorem 4.4). In order to solve the least squares problems appearing in Step 2 of Algorithm 1, the following characterization of their solution turns out to be very useful. Note that the weighted ℓ_2-norm is strictly convex, therefore its minimizer subject to an affine constraint is unique.

Lemma 2.22 ([48, Equation (2.6)], [84, Proposition A.23])
We have $\hat{x} = \operatorname*{arg\,min}_{x \in \mathcal{F}_\Phi(y)} \|x\|_{\ell_2(w)}$ *if and only if* $\hat{x} \in \mathcal{F}_\Phi(y)$ *and*

$$\langle \hat{x}, \eta \rangle_w = 0, \qquad \text{for all} \qquad \eta \in \mathcal{N}_\Phi. \tag{2.27}$$

[5] With "positive", we mean that all entries of the vector are positive, i.e., $w_i > 0$, $i = 1, \dots, N$.

By Lemma 2.22, we are able to derive an explicit representation of the weighted ℓ_2-minimizer $\hat{x} := \operatorname*{arg\,min}_{x \in \mathcal{F}_\Phi(y)} \|x\|_{\ell_2(w)}$. Define $D := \operatorname{diag}\left[(w_j)^{-1}\right]_{j=1}^N$, and denote with $\mathcal{R}(\cdot)$ the range of a linear map. Then we have from (2.27) the equivalent formulation

$$D^{-1}\hat{x} \in \mathcal{R}(\Phi^*),$$

by the fact that $\mathcal{N}_\Phi \perp \mathcal{R}(\Phi^*)$. Therefore, there is a $\xi \in \mathbb{R}^m$ such that $\hat{x} = D\Phi^*\xi$. To compute ξ, we observe that

$$y = \Phi\hat{x} = (\Phi D \Phi^*)\xi,$$

and thus, since Φ has full rank and $\Phi D \Phi^*$ is invertible, we conclude

$$\hat{x} = D\Phi^*\xi = D\Phi^*(\Phi D \Phi^*)^{-1}y.$$

This result is useful in order to provide an explicit representation of Step 2 in Algorithm 1. The minimizer of the least squares problem is explicitly given by the equation

$$x^{n+1} = D_n \Phi^* (\Phi D_n \Phi^*)^{-1} y, \tag{2.28}$$

where we introduced the $N \times N$ diagonal matrix

$$D_n := \operatorname{diag}\left[(w_j^n)^{-1}\right]_{j=1}^N. \tag{2.29}$$

Furthermore, the new weight vector in Step 4 of Algorithm 1 is explicitly given by

$$w_j^{n+1} = (|x_j^{n+1}|^2 + (\varepsilon^{n+1})^2)^{-\frac{2-p}{2}}, \qquad j = 1, \dots, N. \tag{2.30}$$

Taking into consideration that $w_j > 0$, this formula can be derived from the first order optimality condition $\partial J_p(x^{n+1}, w, \varepsilon^{n+1})/\partial w = 0$.

Indeed, in [48, Section 5 and 7], the authors show the convergence of Algorithm 1 by the monotonicity of J_p along its iterations. This allows for showing that the limit vector $x^* := \lim_{n\to\infty} x^n$ is a minimizer of problem (2.4) if $p = 1$ and $\lim_{n\to\infty} \varepsilon^n = 0$. Moreover, a linear rate of convergence can be established in a ball around the sparse solution; In the case of $p < 1$ the rate is even super-linear. For a detailed statement and proof of those results, the interested reader is referred to the respective literature [48, 84, 75]. We omit it here since we present a modified version of IRLS in Section 4.1.2, which actually generalizes the results in the literature.

2.4.1.2 A Practical Comment on the Convergence of IRLS

In this section, we want to comment on the evolution of the difference of successive iterates $\|x^n - x^{n+1}\|_{\ell_2}$ of IRLS, which is not converging to zero for $n \to \infty$ in practice,

although it is proven in theory [48]. This has consequences for the definition of a stopping criterion that is defined by this difference. For simplicity we only consider here the case of $p = 1$.

As mentioned in the previous section, it is possible that the ε^n converge to zero, i.e., $\lim_{n\to\infty} \varepsilon^n = 0$. Since at the same time some components x_i^n may vanish for $n \to \infty$, in practice we will likely get to the point, where we have to divide by a numerical zero in Step 4. In order to avoid this situation happening in practice, one replaces Step 3 by

$$\varepsilon^{n+1} := \max\left(\min\left(\varepsilon^n, \frac{r(x^{n+1})_{K+1}}{N}\right), \varepsilon_{\min}\right), \tag{2.31}$$

for a parameter $\varepsilon_{\min} > 0$, so that $\varepsilon^n \geq \varepsilon^* := \lim_{n\to\infty} \varepsilon^n \geq \varepsilon_{\min}$, for all $n \in \mathbb{N}$. Using this update rule, we run Algorithm 1 for a typical problem with $N = 1500$, $m = 250$, and $K = 45$ for a K-sparse (normalized) signal x and its noiseless measurements $y = \Phi x$ with (normalized) Gaussian measurement matrix Φ, and set $\varepsilon_{\min} = 1\text{E-}10$. In Figure 2.2(a), we plot the history of the ℓ_1-error of the iterates with respect to the ℓ_1-minimizer x^*, i.e., $\|x^n - x^*\|_{\ell_1}$, the history of the ε^n, and the history of the difference of successive iterates, i.e., $\|x^{n+1} - x^n\|_{\ell_2}$ versus the iteration number n. We clearly observe, that the decreasing of the ε^n stops as soon as $\varepsilon_{\min}$ is reached. A few iterations after this limit was reached ($n \geq 150$), the value of the error and the difference of the successive iterates start to *wiggle* around a certain constant value. We conjecture that this behavior is very likely due to the solution of the linear system in (2.28) since the small value of ε^n has as a consequence a bad conditioning of the matrix $\Phi D_n \Phi^*$. The most important observation is, that the difference of successive iterates does not become arbitrarily small, in contrast to the theoretical results in [48], where asymptotic convergence was shown. In practice this is a drawback since this property is often used as a stopping criterion. Thus it may be useful to have some prior knowledge about the general behavior of $\|x^{n+1} - x^n\|_{\ell_2}$ (for large n) in order to make sure that the algorithm can be stopped automatically, e.g., if we could a priori estimate a value $B \in \mathbb{R}$ for which we know that $\|x^{n+1} - x^n\|_{\ell_2} \approx B$, for large n, then it is reasonable to define the stopping rule $B - \epsilon \leq \|x^{n+1} - x^n\|_{\ell_2} \leq B + \epsilon$ for a small value $\epsilon > 0$.

Regarding Figure 2.2(a), also the error with respect to the ℓ_1-minimizer reaches an early saturation in practice. This means that the accuracy of the result of the IRLS algorithm has numerical limitations depending on $\varepsilon_{\min}$ (or even the limit ε^*, which is the same in this example).

In a personal communication with Colas Schretter[6], we figured out how the accuracy of the IRLS result depends on ε^*[7] starting by the following observation from Colas

[6] Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussel

[7] Those results are not published yet, but were discussed in a personal correspondence.

Schretter: He proposed the update rule

$$\varepsilon^{n+1} := \gamma \frac{\sum_{i=1}^{N} |x_i^n|^3}{N^2 \sum_{i=1}^{N} |x_i^n|}, \tag{2.32}$$

which was originally motivated by the weighted variance of x^n. It was created with the intention to dispose of an update-rule which is not depending on K and $\varepsilon_{\min}$, and which scales with $\|x^*\|_{\ell_2}$.[8] Notice that the rule (2.32) does not ensure that the ε^n are decreasing, which is required as an ingredient in the proof of convergence in [48]. Nevertheless, at least in our experiments the algorithm "converged" in practice[9] being equipped with this rule. For the same test problem, as used above, we present the numerical results of IRLS with the update rule (2.32) (setting $\gamma = 1$) in Figure 2.2(b). The history of the ε^n appears smoother than the one in Figure 2.2(a). The "final value"[10] of the approximation error of the IRLS solution with respect to the ℓ_1-minimizer is bigger, and the final value of the difference of successive iterates is smaller than in Figure 2.2(a). We also observe the wiggling in the history plot of the difference of successive iterates. Furthermore, it seems that the wiggling in the approximation error history disappeared. But this is only a display issue; Unfortunately, when zooming in, still a wiggling is observable, although with smaller amplitude.

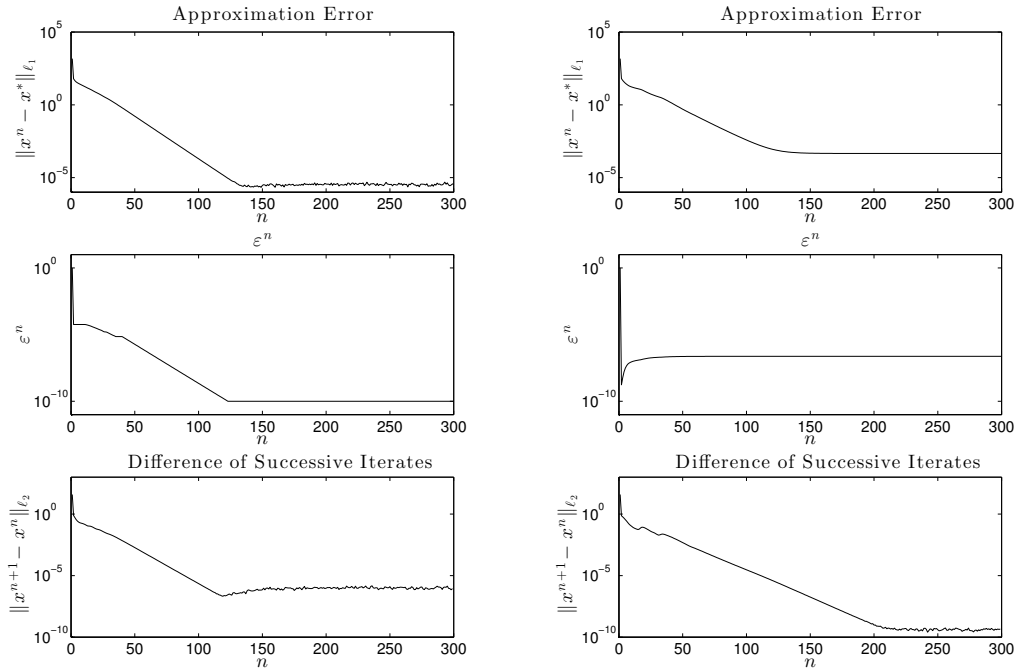
In the latter experiment, the algorithm produces the limit value $\varepsilon^* \approx 2.5\text{E-}7$. Remind that we do not have any influence on this limit value since the update rule does not depend on any parameter. In order to summarize the outcome of both experiments, we observed that for a bigger ε^* the final value on the approximation error increased, the final value on the difference of successive iterates decreased, and also the amplitude of the wiggling slightly decreased.

Although we are not able to eliminate the wiggling by the rule (2.32), the above observations gave us an impression how the lower bound on $\|x^{n+1} - x^n\|_{\ell_2}$ and $\|x^n - x^*\|_{\ell_1}$ is correlated with the value of $\varepsilon^* = \lim_{n\to\infty} \varepsilon^n$. In order to obtain an even clearer view, we ran a test series of Algorithm 1 with the update rule (2.31), using the parameter $\varepsilon_{\min} \in \{10^{-t} |\ t = 3, \dots, 13\}$. For each experiment, we plot in Figure 2.3 by a $\times$-marker the final value of the approximation error, and the final value of the difference of successive iterates versus the parameter ε^*. In all experiments, we obtained $\varepsilon^* = \varepsilon_{\min}$. For comparison, we added by a $+$-marker also the results for a test series of Algorithm 1 with the update rule (2.32) using the parameter $\gamma \in \{10^t |\ t = -3, \dots, 3\}$. Starting the analysis on the right-hand side plot, it seems that the difference of successive iterates

[8] In a "black box" implementation of IRLS such a rule actually releases the user from the daunting task of setting and adapting many parameters.

[9] We mean "convergence" in the sense as we observed it in the experiment before: The difference of the successive iterates starts to wiggle around a certain constant value.

[10] With "final value" we denote the mean of the values at iterations 250–300.



(a) Results using (2.31).

(b) Results using (2.32).

Figure 2.2: History of characteristic quantities (versus the iteration number n) in an IRLS test run with the ε-update rules (2.31) and (2.32) respectively.

is reciprocally decreasing with the value of ε^*. This means that the iterates wiggle less, when a big $\varepsilon_{\min}$ is chosen. However, in the design of the IRLS algorithm the ε^n were introduced as a "small" regularization variable with the purpose to avoid a division by zero. It is intuitive that the approximation error of the result of the algorithm with respect to the ℓ_1-minimizer is getting worse, the more we regularize, i.e., the bigger is ε^*. Thus, we have two concurrent drivers which influence the approximation error. This can be seen in the left-hand side plot of Figure 2.3: With decreasing ε^*, the approximation error is getting smaller until the effect of the difference of successive iterates kicks in, and again deteriorates the approximation. Thus, there is an optimal value for ε^*. Regarding the results with the update rule (2.32), we see that they perfectly get in line with the results of the other test series. Therefore, we conclude that ε^* is the essential quantity that trades between accuracy and robustness (in form of the wiggling) of the IRLS solution.

Focusing on the difference of successive iterates, we claim the hypothesis that there is $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$,

$$\frac{\|x^{n+1} - x^n\|_{\ell_2}}{\|x^n\|_{\ell_2}} \sim \frac{C}{\varepsilon^*}, \tag{2.33}$$

with a constant $0 < C \ll 1$. In our experiments, we measured $C \sim 1\text{E-}16$.

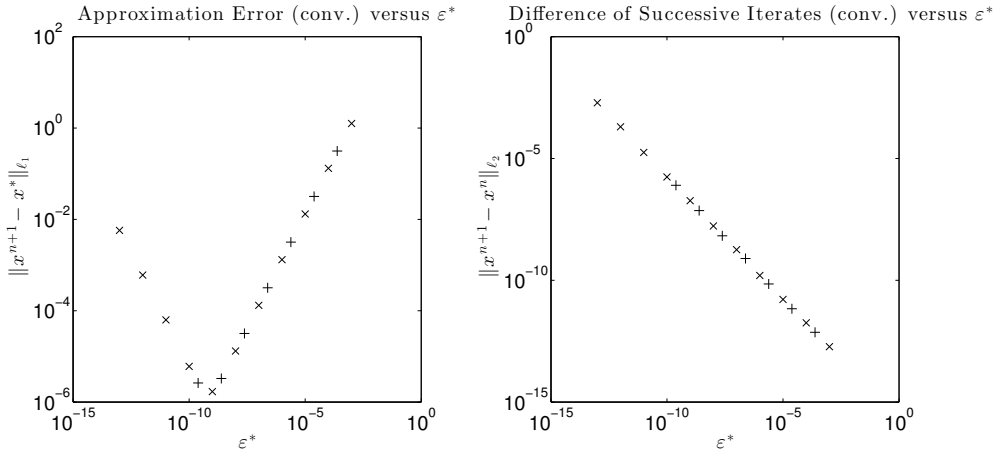


Figure 2.3: Dependency of the final approximation error and the difference of successive iterates (after numerical convergence) on the limit value ε^*.

We leave hypothesis (2.33) as an open problem, and only give a theoretical indication for an upper bound on $\|x^{n+1} - x^n\|_{\ell_2}/\|x^n\|_{\ell_2}$. Therefore, we highlight that x^{n+1} is only an approximation of the exact solution $\hat{x}^{n+1} := D_n\Phi^*(\Phi D_n\Phi^*)^{-1}y$. This is due to the computation in finite precision. We denote the error $\Delta x^{n+1} := x^{n+1} - \hat{x}^{n+1}$. Since $\|x^{n+1} - x^n\|_{\ell_2} \leq \|\hat{x}^{n+1} - x^n\|_{\ell_2} + \|\Delta x^{n+1}\|_{\ell_2}$, and it was shown in [48] that $\|\hat{x}^{n+1} - x^n\|_{\ell_2}$ vanishes for $n \to \infty$ in theory, then we conclude that $\|x^{n+1} - x^n\|_{\ell_2} \sim \|\Delta x^{n+1}\|_{\ell_2}$ for sufficiently large n. Therefore, we focus on giving an upper bound for the error $\|\Delta x^{n+1}\|_{\ell_2}$. The origin of this error can be explained in detail as follows: When solving problem (2.28), we first need to find a solution $\hat{\xi}^{n+1}$ of the (unperturbed) linear system $(\Phi D_n\Phi^*)\hat{\xi}^{n+1} = y$. In finite precision the "=" has to be replaced by a "≈", and we rather solve the perturbed linear system $(\Phi D_n\Phi^*)(\hat{\xi}^{n+1} + \Delta\xi^{n+1}) = y + \Delta y$, where $\|\Delta y\|_{\ell_2} \leq \text{eps}\|y\|_{\ell_2}$, with eps being the relative machine precision.[11] The unknown error vector $\Delta\xi^{n+1}$ is then further propagated into Δx^{n+1}, via $\hat{x}^{n+1} + \Delta x^{n+1} = D_n\Phi^*(\hat{\xi}^{n+1} + \Delta\xi^{n+1})$. In the following theorem, we give an upper bound for $\|\Delta x^{n+1}\|_{\ell_2}$.

[11] One could also add a perturbation to the matrix $\Phi D_n\Phi^*$, but for a simplified analysis, we omit it here. We further see in Theorem 2.23 that the bad conditioning of $\Phi D_n\Phi^*$ is mainly responsible for the error amplification, which is much severe than a small perturbation in the matrix itself.

Theorem 2.23
Assume that the IRLS algorithm produces a sequence of iterates x^n and that n exceeded some value n_0 such that

$$\max_i(|x_i^n|) + \varepsilon^n \leq c_1 \|x^n\|_{\ell_2} \tag{2.34}$$

for some constant $1 \leq c_1 \sim 1$, for all $n \geq n_0$. Let Φ be full rank, and let $\hat{\xi}^{n+1}$ be the solution of the (unperturbed) linear system $(\Phi D_n \Phi^*)\hat{\xi}^{n+1} = y$. Consider furthermore the perturbed linear system $(\Phi D_n \Phi^*)(\hat{\xi}^{n+1} + \Delta\xi^{n+1}) = y + \Delta y$. Then a bound on the error in the solution of the weighted least squares system is given by

$$\|\Delta x^{n+1}\|_{\ell_2} \leq \bar{C}\|x^n\|_{\ell_2} \frac{\|\Delta y\|_{\ell_2}}{\varepsilon^n}, \tag{2.35}$$

where $\bar{C} := c_1 \frac{\|\Phi\|}{(\sigma_{\min}(\Phi))^2}$.

Remark 2.24
Condition (2.34) is not restrictive in practice since we want the limit of the ε^n to be much smaller than the limit of the x^n.

Remark 2.25
The theorem explains, why the constant C in (2.33) is in the order of 1E-16 in our experiments. We computed $\bar{C} \approx 1.5$ for the used Gaussian matrix, and the value $\|\Delta y\|_{\ell_2}$ is expected in the order of machine precision, i.e., 1E-16 for double precision.

Proof. Since we have $(\Phi D_n \Phi^*)\hat{\xi}^{n+1} = y$, and $(\Phi D_n \Phi^*)(\hat{\xi}^{n+1} + \Delta\xi^{n+1}) = y + \Delta y$, we obtain

$$(\Phi D_n \Phi^*)\Delta\xi^{n+1} = \Delta y,$$

and thus the ℓ_2-norm error of the solution of the linear system can be estimated by

$$\|\Delta\xi^{n+1}\|_{\ell_2} \leq \|(\Phi D_n \Phi^*)^{-1}\| \|\Delta y\|_{\ell_2}.$$

According to (2.28) the least squares solution is obtained by

$$x^{n+1} = D_n \Phi^* \hat{\xi}^{n+1},$$

and thus by error propagation, we obtain

$$\begin{aligned} \|\Delta x^{n+1}\|_{\ell_2} &\leq \|D_n \Phi^*\| \|\Delta\xi^{n+1}\|_{\ell_2} \leq \|D_n \Phi^*\| \|(\Phi D_n \Phi^*)^{-1}\| \|\Delta y\|_{\ell_2} \\ &\leq \|D_n\| \|\Phi^*\| \|(\Phi D_n \Phi^*)^{-1}\| \|\Delta y\|_{\ell_2}. \end{aligned} \tag{2.36}$$

From the definition of the singular values, we estimate

$$\begin{aligned} \|D_n\| = \sigma_{\max}(D_n) &= \max_i \left((w_i^n)^{-1}\right) = \max_i \sqrt{(x_i^n)^2 + (\varepsilon^n)^2} \\ &\leq \max_i(|x_i^n|) + \varepsilon^n \leq c_1 \|x^n\|_{\ell_2}, \end{aligned} \tag{2.37}$$

where the last step follows by (2.34). Moreover

$$\begin{aligned}\|(\Phi D_n \Phi^*)^{-1}\| &= \frac{1}{\sigma_{\min}(\Phi D_n \Phi^*)} \leq \frac{1}{(\sigma_{\min}(\Phi))^2\ \sigma_{\min}(D_n)} \\ &= \frac{1}{(\sigma_{\min}(\Phi))^2\ \min_i \sqrt{(x_i^n)^2 + (\varepsilon^n)^2}} \leq \frac{1}{(\sigma_{\min}(\Phi))^2\ \varepsilon^n}.\end{aligned} \tag{2.38}$$

Plugging in (2.37), and (2.38) in (2.36) yields (2.35). □

In view of Theorem 2.23, we conclude this section by a practical hands-on guideline for a stopping criterion for IRLS: Run the algorithm until the values ε^n do not change anymore (or only change within a very small tolerance) for some n.[12] Then we have that $\varepsilon^* \approx \varepsilon^n \approx \varepsilon^{n+1}$ and we iterate the algorithm until

$$\frac{\|x^{n+1} - x^n\|_{\ell_2}}{\|x^n\|_{\ell_2}} \leq \bar{\bar{C}} \frac{\|\Delta y\|_{\ell_2}}{\varepsilon^*},$$

where $\bar{\bar{C}}$ has to be chosen slightly bigger than $\bar{C}$ and $\|\Delta y\|_{\ell_2}$ has to be replaced by a value in the order of the machine precision.

2.4.1.3 IRLS Method for ℓ_p-norm Regularized Least Squares

As we explained in Section 2.2, sometimes it is more appropriate to work with a functional that balances the residual error in the linear system with an ℓ_p-norm penalty, promoting sparsity. We consider the problem

$$\arg\min_{x \in \mathbb{R}^N} \left(F_{p,\lambda}(x) := ||x||_{\ell_p}^p + \frac{1}{2\lambda} ||\Phi x - y||_{\ell_2}^2 \right), \tag{2.39}$$

where $\lambda > 0$. This problem is actually equivalent to problem (2.14), if we replace "λ" by "2λ". We use the formulation (2.39) in order to ease cross-reading. Again, we define an auxiliary functional for the analysis of the algorithm.

Definition 2.26
Given a real number $\varepsilon > 0$, $x \in \mathbb{R}^N$, and a weight vector $w \in \mathbb{R}^N$, $w > 0$, we define

$$J_{p,\lambda}(x, w, \varepsilon) := \frac{p}{2} \sum_{j=1}^{N} \left[|x_j|^2 w_j + \varepsilon^2 w_j + \frac{2-p}{p} w_j^{-\frac{p}{2-p}} \right] + \frac{1}{2\lambda} ||\Phi x - y||_{\ell_2}^2. \tag{2.40}$$

[12] The update rule (2.31) is designed such that $\varepsilon^{n+1} = \varepsilon^n$ if $r(x^{n+1})_{K+1}/N \geq \varepsilon^n$ (in order to ensure that the ε^n decrease), although one may still be far away from the solution. Thus one may exclude such special cases by, e.g., checking the value of ϵ^n for the previous 10 iterations.

Lai, Xu, and Yin in [120], and Daubechies and Voronin in [189, 190] showed independently that computing the optimizer of the problem (2.39) can be approached by an alternating minimization of the functional $J_{p,\lambda}$ with respect to x, w, and ε. The difference between these two works is the definition of the update rule for ε. Both authors show that the algorithm converges. Furthermore, in [189, 190], the authors show that the limit point is a minimizer ($p = 1$), or a critical point ($p < 1$) of (2.39). By the ε-update rule in [120], the authors showed that the algorithm converges, in the case that $\lim_{n\to\infty} \varepsilon^n = \varepsilon > 0$, to a minimizer ($p = 1$), or critical point ($p < 1$) of the smoothed functional

$$||x||^p_{\ell_p,\varepsilon} + \frac{1}{2\lambda}||\Phi x - y||^2_{\ell_2}, \tag{2.41}$$

where $||x||^p_{\ell_p,\varepsilon} := \sum_{j=1}^{N} |x_j^2 + \varepsilon^2|^{\frac{p}{2}}$. The rule that is used by Daubechies and Voronin allows a more elegant analysis of the algorithm and will be more useful for the analysis of a modified IRLS method that we present in Section 4.1.3. Therefore, we define IRLS-λ in Algorithm 2, using the update rule of Daubechies/Voronin.

Algorithm 2 IRLS-λ

1: Set $w^0 := (1, \dots, 1)$, $\varepsilon^0 := 1$, $\alpha \in (0, 1]$, $\phi \in (0, \frac{1}{4-p})$.
2: **while** $\varepsilon^n > 0$ **do**
3: $\quad x^{n+1} := \arg\min_x J_{p,\lambda}(x, w^n, \varepsilon^n)$
4: $\quad \varepsilon^{n+1} := \min\left\{\varepsilon^n, |J_{p,\lambda}(\tilde{x}^{n-1}, w^{n-1}, \varepsilon^{n-1}) - J_{p,\lambda}(\tilde{x}^n, w^n, \varepsilon^n)|^\phi + \alpha^{n+1}\right\}$
5: $\quad w^{n+1} := \arg\min_{w>0} J_{p,\lambda}(x^{n+1}, w, \varepsilon^{n+1})$
6: **end while**

We approach the first step of the algorithm by computing a critical point of $J_{p,\lambda}(\cdot, w, \varepsilon)$ via the first order optimality condition

$$p\left[x_j w_j^n\right]_{j=1}^N + \frac{1}{\lambda}\Phi^*(\Phi x - y) = 0, \tag{2.42}$$

or equivalently

$$\left(\Phi^*\Phi + \operatorname{diag}\left[\lambda p w_j^n\right]_{j=1}^N\right) x = \Phi^* y. \tag{2.43}$$

We denote the solution of this linear system by x^{n+1}. The new weight w^{n+1} is obtained in Step 5, and can be expressed componentwise by

$$w_j^{n+1} = ((x_j^{n+1})^2 + (\varepsilon^{n+1})^2)^{-\frac{2-p}{2}}. \tag{2.44}$$

As in the previous section, we omit a detailed statement of the above mentioned convergence results since we present one for a modified version of Algorithm 2 in Section 4.1.3 that greatly extends them.

2.4.2 Iteratively Re-weighted ℓ_1-minimization (IRL1)

The ℓ_1-norm has the drawback with respect to the ℓ_0-norm, that it is not democratic, i.e., while the ℓ_0-norm penalizes all non-zero entries by one, in the ℓ_1-norm the contribution of each entry depends on its amplitude. An ansatz to overcome this drawback is *iteratively re-weighted ℓ_1-minimization* (IRL1). It was proposed in [33], and analyzed in [141]. It iteratively computes the solution of

$$x^{n+1} = \underset{x \in \mathcal{F}_\Phi(y)}{\arg\min} \sum_{i=1}^{N} w_i^n |x_i|, \quad n = 0, 1, 2, \ldots, \tag{2.45}$$

or

$$x^{n+1} = \underset{\|\Phi x - y\|_{\ell_2} \leq \delta}{\arg\min} \sum_{i=1}^{N} w_i^n |x_i|, \quad n = 0, 1, 2, \ldots, \tag{2.46}$$

while updating the weights according to $w_i^n = (|x_i^n| + a)^{-1}$, $i = 1, \ldots, N$, for a suitably chosen stability parameter $a > 0$ (which is considered fixed and prevents the weights to go to infinity), and a potential noise level $\delta \geq 0$. The weights w_i^n are used in order to promote that all non-zero entries equally contribute to the value of the objective function. Thus, instead of solving the NP-hard ℓ_0-minimization problem, one can solve a series of the convex weighted ℓ_1-minimization problems, while preserving the democratic property of the ℓ_0-norm.

We recall an instance optimality property for the result of the iterative procedure (2.45). We use it in Section 3.1.1 in order to perform a robustness analysis towards the support identification properties of IRL1. We will not further theoretically analyze (2.46), but for the sake of completeness and a broad investigation, we use it in the numerical tests in Section 3.3.

Lemma 2.27 ([141, Theorem 3.2])
Let $\Phi \in \mathbb{R}^{m \times N}$ have the $(2k, \delta_{2k})$-RIP, with $\delta_{2k} < \sqrt{2} - 1$, and assume the smallest nonzero coordinate of $x_{[k]}$ in absolute value larger than the threshold

$$\bar{r} := 9.6 \frac{\sqrt{1 + \delta_{2k}}}{1 - (\sqrt{2} + 1)\delta_{2k}} \left(\sigma_k(x)_{\ell_2} + \frac{\sigma_k(x)_{\ell_1}}{\sqrt{k}} \right). \tag{2.47}$$

Denote the decoder given by the iterative procedure (2.45) by Δ_{1rew}, and the limit by $\Delta_{1rew}(y)$. Then for $x \in \mathcal{F}_\Phi(y)$, the decoder Δ_{1rew} performs

$$\|x - \Delta_{1rew}(y)\|_{\ell_2} \leq 4.8 \frac{\sqrt{1 + \delta_{2k}}}{1 + (\sqrt{2} - 1)\delta_{2k}} \left(\sigma_k(x)_{\ell_2} + \frac{\sigma_k(x)_{\ell_1}}{\sqrt{k}} \right). \tag{2.48}$$

2.4.3 Thresholding Algorithms

Motivated by the ℓ_p-norm regularized problem descriptions (2.12), (2.13), and (2.14), we consider a generalized problem formulation of the type

$$\arg\min_{x\in\mathbb{R}^N} \lambda\phi(x) + \|\Phi x - y\|_{\ell_2}^2, \tag{2.49}$$

where $\phi(x)$ is a separable lower semi-continuous function. Thresholding algorithms can be considered as a subclass of *forward-backward splitting* methods. A basic thresholding algorithm consists of a forward gradient descent (Landweber) step on the function $\|\Phi x - y\|_{\ell_2}^2$, with stepsize $t^{(n)}$, and a backward gradient descent step (also called *proximal step*). All in all, this means, that we iterate

$$x^{(n+1)} = \mathbb{T}_\alpha(x^{(n)} + t^{(n)}\Phi^*(\Phi x^{(n)} - y)),$$

where the proximity operator $\mathbb{T}_\alpha$ is defined as

$$\mathbb{T}_\alpha(x) = \arg\min_{z\in\mathbb{R}^N} \alpha\phi(z) + \|z - x\|_{\ell_2}^2, \tag{2.50}$$

where α may depend on λ and $t^{(n)}$. In the case of $\phi(\cdot) = \|\cdot\|_{\ell_p}^p$, $0 \le p \le 1$, the proximity operator can be reduced for some values of p to a closed form component-wise thresholding operator, which gives the name to those methods. We recall an explicit formulation of those thresholding functions for $p \in \{0, 1\}$ in the following two subsections. A further promising thresholding algorithm with a closed form operator for $p = 1/2$ was proposed and analyzed in [193, 196] (compare also Remark 3.14).

In Section 3.2, we use thresholding iterations in order to solve multi-penalty regularized functionals, such as

$$\arg\min_{x,z\in\mathbb{R}^N} \lambda_p\|x\|_{\ell_p}^p + \lambda_q\|z\|_{\ell_q}^q + \|\Phi(x + z) - y\|_{\ell_2}^2,$$

with $\lambda_p, \lambda_q > 0$ and arbitrary values $0 < p \le 1$, $1 \le q \le \infty$. The algorithm that we present, principally fixes alternatingly either x or z, and solves as an intermediate step again problems of the type (2.49).

2.4.3.1 Iterative Soft Thresholding (ISTA)

In order to solve problem (2.12), which is of the type (2.49), we have to find the proximal operator for $\phi(\cdot) = \|\cdot\|_{\ell_1}$. It was shown that it is actually uniquely given by the *soft thresholding operator* $\mathbb{S}_\lambda(x)$ (see, e.g., [47]), which is defined component-wise by

$$(\mathbb{S}_\lambda(x))_i := \begin{cases} \left(1 - \frac{\lambda}{2\|x_i\|_{\ell_2}}\right) x_i, & \|x_i\|_{\ell_2} > \frac{\lambda}{2}, \\ 0, & \text{otherwise}, \end{cases} \quad i = 1, \dots, N. \tag{2.51}$$

We obtain the well-known *iterative soft thresholding algorithm* (ISTA). Furthermore, it can be equipped by a stepsize $t \leq 1$, as presented in Algorithm 3. Several authors have proposed this algorithm independently in different context [71, 169, 172, 63].

Algorithm 3 Iterative Soft Thresholding Algorithm (ISTA)

Choose $x^{(0)}$, e.g., $x^{(0)} = 0$, and $t \leq 1$.

1: **loop**
2: $\quad x^{(n+1)} = \mathbb{S}_{t\lambda}\left(x^{(n)} + t\Phi^*(y - \Phi x^{(n)})\right)$
3: **end loop**

For the sake of the consistency of this section, we chose a finite dimensional formulation, although ISTA can similarly solve the infinite dimensional problem (2.15). We address this particular situation in Sections 3.2 and 4.2. Strong convergence of this algorithm (which is in particular interesting in the infinite dimensional setting) was proved in [47], under the assumption that $\|\Phi\| < 1$ (actually, convergence can be shown also for $\|\Phi\| < \sqrt{2}$, compare also [43]). This condition is not a restriction, since it can always be met by a suitable rescaling of Φ, λ, and y without changing the actual problem. Moreover, the limit point x^* is a fixed point of the thresholded Landweber iteration

$$x^* = \mathbb{S}_\lambda(x^* + \Phi^*(y - \Phi x^*)).$$

ISTA is suitable for a quick implementation, but converges slowly in general. Therefore it was subject of some acceleration techniques, e.g., by using a decreasing sequence of positive thresholds $\lambda^{(n)}$ [45], or using an adaptive stepsize $t^{(n)}$ instead of the static stepsize t [181]. One of the most popular acceleration techniques is the *fast iterative soft thresholding algorithm* (FISTA) [13], which is based on [144]. We state it in Algorithm 4. It is known for its good worst-case performance and achieves a converge rate of $\mathcal{O}(n^{-2})$ while ISTA was shown to converge only linearly [22].

Algorithm 4 Fast Iterative Soft Thresholding Algorithm (FISTA)

Choose $x^{(0)} = u^{(0)} = 0$, $t \leq 1$, $s^{(0)} = 1$.

1: **loop**
2: $\quad u^{(n+1)} = \mathbb{S}_{t\lambda}\left(x^{(n)} + t\Phi^*(y - \Phi x^{(n)})\right)$
3: $\quad s^{(n+1)} = \frac{1+\sqrt{1+4(s^{(n)})^2}}{2}$
4: $\quad w^{(n)} = 1 + \frac{s^{(n)}-1}{s^{(n+1)}}$
5: $\quad x^{(n+1)} = w^{(n)}u^{(n+1)} + \left(1 - w^{(n)}\right)u^{(n)}$
6: **end loop**

Both, ISTA and FISTA, can be further accelerated by backtracking strategies, as

proposed in [13], where the stepsize $t = t^{(n)}$ is again adaptively chosen. We explain this particular technique more in detail in Section 4.2.1.

2.4.3.2 Iterative Hard Thresholding (IHT)

Let us now consider $\phi(\cdot) = \|\cdot\|_0$, which leads to the *iterative hard thresholding algorithm* (IHT). There are two versions, respectively for the problems (2.13) and (2.11), although a thresholding operator for the solution of problem (2.11) cannot directly be derived from the proximity operator (2.50), but rather from a simple projection on the set of sparse vectors. Let us briefly introduce those two thresholding operators, before going into detail:

- It turns out that in the case of problem (2.11) the thresholding operator $\mathbb{H}_k(z) := z_{[k]}$, which returns the best k-term approximation to z (see Definition 2.2), is appropriate. Note that if x^* is k-sparse, and $\Phi x^* = y$, then x^* is a fixed point of

$$x^* = \mathbb{H}^k(x^* + \Phi^*(y - \Phi x^*)).$$

  Further below, we recall proper analysis results of the algorithm with iteration

$$x^{n+1} = \mathbb{H}^k(x^n + \Phi^*(y - \Phi x^n)).$$

  We show that under the RIP for Φ, it converges to a local minimizer of (2.11), which fulfills the aforementioned fixed point equation, and has stability properties as in (2.8), which are reached in a finite number of iterations.

- In order to solve problem (2.13), the thresholding operator is given by $\mathbb{H}_{\sqrt{\lambda}}(z)$ where

$$(\mathbb{H}_{\sqrt{\lambda}}(z))_i := \begin{cases} z_i & \text{if } |z_i| > \sqrt{\lambda}, \\ 0 & \text{else,} \end{cases}, \ i = 1 \ldots, N.$$

  Again, we present further below a theorem that states the convergence of the algorithm to a fixed point x^* fulfilling

$$x^* = \mathbb{H}_{\sqrt{\lambda}}\left(x^* + \Phi^*(y - \Phi x^*)\right), \tag{2.52}$$

  which is a local minimizer of the functional (2.13).

IHT for the ℓ_0-constrained Problem We first specify in Algorithm 5 the formulation of IHT for problem (2.11), so concisely described in the introductory paragraph. It was shown in [19] that if $\|\Phi\| < 1$ then this algorithm converges to a local minimizer of (2.13). The same authors establish in [20] the following convergence result in the case that Φ satisfies the RIP.

Algorithm 5 IHT-k

1: Set $x^0 := 0$.
2: **loop**
3: $\quad x^{n+1} := \mathbb{H}^k(x^n + \Phi^*(y - \Phi x^n)) = (x^n + \Phi^*(y - \Phi x^n))_{[k]}$
4: **end loop**

Theorem 2.28 ([20, Theorem 5])
Let us assume that $y = \Phi x + e$ is a noisy encoding of x via Φ, where x is k-sparse. If Φ has the $(3k, \delta_{3k})$-RIP with constant $\delta_{3k} < \frac{1}{\sqrt{32}}$, then, at iteration n, Algorithm 5 recovers an approximation x^n satisfying

$$\|x - x^n\|_{\ell_2} \leq 2^{-n}\|x\|_{\ell_2} + 5\|e\|_{\ell_2}.$$

Furthermore, after at most

$$n^* = \left\lceil \log_2 \left(\frac{\|x\|_{\ell_2}}{\|e\|_{\ell_2}} \right) \right\rceil$$

iterations, the algorithm estimates x with accuracy

$$\|x - x^{n^*}\|_{\ell_2} \leq 6\|e\|_{\ell_2}.$$

Moreover a result for arbitrary vectors x is given.

Corollary 2.29 ([20, Theorem 4])
Let us assume that $y = \Phi x + e$ is a noisy encoding of x via Φ, where x is an arbitrary vector. *If Φ has the $(3k, \delta_{3k})$-RIP with constant $\delta_{3k} < \frac{1}{\sqrt{32}}$, then, at iteration n, Algorithm IHT-k will recover an approximation x^n satisfying*

$$\|x - x^n\|_{\ell_2} \leq 2^{-n}\|x\|_{\ell_2} + 6\left(\sigma_k(x)_{\ell_2} + \frac{\sigma_k(x)_{\ell_1}}{\sqrt{k}} + \|e\|_{\ell_2}\right).$$

Furthermore, after at most

$$n^* = \left\lceil \log_2 \left(\frac{\|x\|_{\ell_2}}{\|e\|_{\ell_2}} \right) \right\rceil$$

iterations, the algorithm estimates x with accuracy

$$\|x - x^{n^*}\|_{\ell_2} \leq 7\left(\sigma_k(x)_{\ell_2} + \frac{\sigma_k(x)_{\ell_1}}{\sqrt{k}} + \|e\|_{\ell_2}\right).$$

IHT for the ℓ_0-regularized Problem The introductory rough formulation of an algorithm for the solution of problem (2.13) is specified in Algorithm 6. In the following, we present a respective convergence result of [19], which is used further below in Section 3.1. We also sketch the proof.

Algorithm 6 IHT-λ

Set $x^0 := 0$.
loop
  $z^{n+1} := x^n + \Phi^*(y - \Phi x^n)$
  for $i = 1, \ldots, N$ **do**
    if $z_i^{n+1} > \sqrt{\lambda}$ **then**
      $x_i^{n+1} := z_i^{n+1}$
    else
      $x_i^{n+1} := 0$
    end if
  end for
end loop

Theorem 2.30 ([19, Theorem 3, Lemma 4])
If $\|\Phi\| < 1$, then the sequence $(x^n)_{n\in\mathbb{N}}$ defined by Algorithm 6 converges to a fixed point x^* of Algorithm 6, and thus fulfilling (2.52), which is a local minimum of $\mathcal{J}_0(x)$. If furthermore the set of columns $\{\Phi_i\}_{i=1}^N$ contains a basis for the signal space and $\|\Phi_i\|_{\ell_2} > 0$, $i = 1, \ldots, N$, then a tight bound for the approximation error at the fixed point x^* is

$$\|y - \Phi x^*\|_{\ell_2} \leq \frac{\sqrt{\lambda}}{\beta(\Phi)},$$

where $\beta(\Phi) > 0$ is such that

$$\|\Phi^* z\|_{\ell_\infty} \geq \beta(\Phi)\|z\|_{\ell_2} \tag{2.53}$$

holds for all $z \in \mathbb{R}^m$.

Proof. Define the sets $\Lambda_0(z) := \{i|\ y_i = 0\}$ and $\Lambda_1(z) := \{i|\ |y_i| > \sqrt{\lambda}\}$. The proof of convergence starts by showing that after a finite number of iterations these two sets are fixed. Thus the algorithm then can be considered as a standard Landweber algorithm with guaranteed convergence [122]. In the proof of [19, Theorem 3] a detailed presentation of this argumentation, and the proof that the limit is a fixed point of the algorithm and therefore also a local minimum of $\mathcal{J}_0(x)$, is given. To show the approximation error estimate, we assume that the algorithm converged to a fixed point x^* which then has to fulfill (2.52). Since $\mathbb{H}_{\sqrt{\lambda}}$ is defined component-wise, we conclude that $|\Phi_i^*(y - \Phi x^*)| \leq \sqrt{\lambda}$ if $i \in \Lambda_0(x^*)$, and $\Phi_i^*(y - \Phi x^*) = 0$ if $i \in \Lambda_1(x^*)$. Thus, we have in particular that $\|\Phi_i^*(y - \Phi x^*)\|_{\ell_\infty} \leq \sqrt{\lambda}$. By this observation and the application of condition (2.53) for $z = y - \Phi x^*$, we obtain

$$\beta(\Phi)\|y - \Phi x^*\|_{\ell_2} \leq \|\Phi^*(y - \Phi x^*)\|_{\ell_\infty} \leq \sqrt{\lambda}.$$

Remark 2.31
Although it is the scope of the algorithm to produce a vector with small ℓ_0-norm—and thus a sparse vector—it is important to notice that this algorithm is only computing a local minimizer of the functional $\mathcal{J}_0(x)$, which is not necessarily sparse. In contrast to Algorithm 5 there is no guarantee, that this Algorithm produces a k-sparse vector.

Remark 2.32
In the proof, it is shown that the algorithm can be considered as a standard Landweber algorithm as soon as the sets $\Lambda_0(x^n)$ and $\Lambda_1(x^n)$ are fixed after a finite number of iterations n_0. According to [122], then the algorithm converges linearly as

$$\|x^n - x^*\|_{\ell_2} \leq \|I - \Phi^*_{\Lambda_1(x^{n_0})}\Phi_{\Lambda_1(x^{n_0})}\|^{n-n_0}\|x^{n_0} - x^*\|_{\ell_2}.$$

A brief comparison of the IHT algorithms At first glance Algorithm 5 should be preferred to 6 since it offers a more robust error analysis and a guaranteed error reduction from the very beginning and it is robust to noise, i.e., an estimate of the type (2.8) holds. However its main drawback is that it requires the (precise) knowledge of k, which one might not dispose of in some applications. Therefore in this case one can consider to use Algorithm 6. Nevertheless one has to tune λ. In Section 2.2.3 we present an application of the latter algorithm which turns out to be very robust when one is interested in the exact support identification of an original signal which is corrupted by noise. In this particular application we determine a specific range for λ which is supposed to provide optimal support identification performance.

Chapter 3

Robust Sparse Recovery in the Presence of Strong Signal Noise

The *noise folding problem*, whose importance for the field of sparse recovery was thoroughly discussed in Section 2.2.3, is the main motivation of this chapter. In accordance with the general setting in Section 2.2, we exclusively consider the problem (2.21), i.e.,

$$y = \Phi(\bar{x} + n),$$

here. We use here the notation $\bar{x}$ for a sparse vector since we will use the notation x further below for a sparse vector, corrupted by noise, i.e., the sum of $\bar{x}$ and n. As we explained in the respective section, the measurement vector y can be considered equivalently obtained by a measurement procedure of the form (2.7), i.e., $y = \Phi\bar{x} + e$ (possibly with another measurement matrix Φ of equal statistics), where now the vector e is composed by i.i.d. Gaussian entries with distribution $\mathcal{N}(0, \sigma_e)$ and $\sigma_e^2 = \frac{N}{m}\sigma_n^2$. Therefore, the noise folding phenomenon may significantly reduce in practice the potential advantages of sparse recovery in terms of the trade-off between robustness and efficient compression (here represented by the factor $\frac{N}{m}$), with respect to other more traditional subsampling encoding methods [51]. An approach to control the noise folding, is proposed in [6]. In this case, one may tune the linear measurement process in order to a priori filter the noise. However, this strategy requires to have a precise knowledge of the noise statistics and to design proper filters. Other related work [104, 105, 106] addresses the problem of designing adaptive measurements, called *distilled sensing*, in order to detect and locate the signal within white noise.

In this chapter, we shall follow a *blind-to-statistics* approach, which does not modify the non-adaptive measurements, and, differently from the Dantzig selector analysis in [31] (compare Section 2.2.3), this chapter is restricted to a purely deterministic setting. In Theorem 2.19 it is stated that even if we would have an oracle at disposal that tells us the support of the solution vector still the noise folding phenomenon is present. Thus, in order to not loose even more accuracy due to the wrong detection of the support, the challenge in the noise folding regime is to identify a decoder which "simulates" the oracle, i.e., which robustly and reliably determines the support

$\Lambda = \mathrm{supp}(\bar{x})$ (*support identification*). In fact, for certain applications, such as radar [112], the support recovery can be even more relevant than an accurate estimate of the respective amplitude and sign of the non-zero entries. However, apart from the exact support, we are also interested in the best possible *recovery of those relevant entries*. Provided an exact support identification, a good approximation of the non-zero entries of $\bar{x}$ then can always be found by using the optimization process (2.24). As we will see below, some methods are already defined in such a way that they naturally return a good approximation of the non-zero entries as a byproduct of the method itself. In this scope, the better a recovery method copes with the support identification and estimation of the relevant entries of the vector $\bar{x}$, the more *robust* it is. We likewise say that it has a higher *performance*.

To get an immediate insight into the problems that arise due to the noise folding, we start the investigation with a generic and simple example.

Example 3.1
We recover the decoded signal x^* from a measurement data vector y, which was obtained from the original signal $\bar{x}$ through the measurement process (2.21) with the addition of a noise vector n. As decoder we use the ℓ_1-minimization process (2.4), which is one of the most popular methods for sparse recovery (see Section 2.1). In Figure 3.1, we plot the original signal $\bar{x}$, the noise n, and the recovery result x^* so that we are able to compare it. The immediate observation is that the original signal

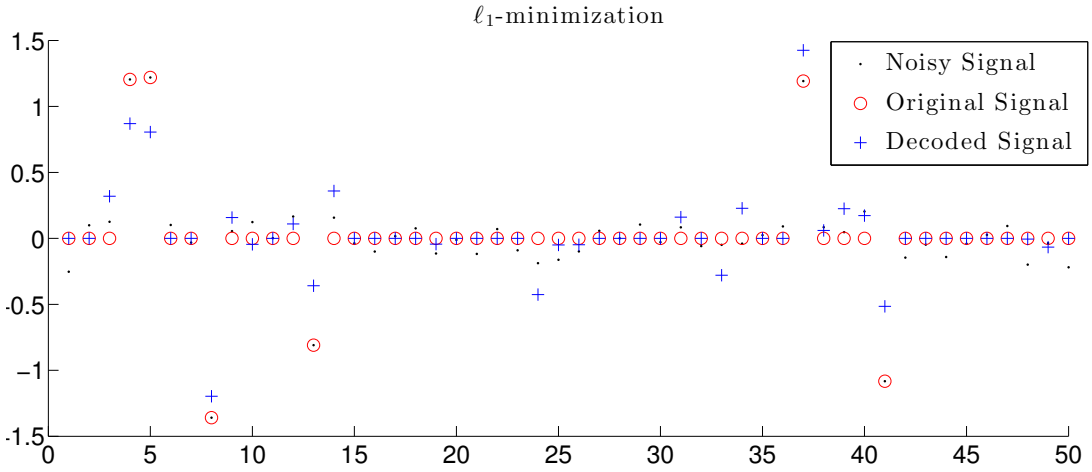


Figure 3.1: Recovery result x^* (+) of the ℓ_1-minimization starting from the measurement of a generic sparse signal $\bar{x}$ (o) in the presence of signal noise n (·).

$\bar{x}$ can clearly be separated from the noise n since all entries of n have an absolute

value below 0.25, and the relevant large signal entries of $\bar{x}$ exceed 0.5 in absolute value. It is rather difficult to make a similar distinction within the decoded signal x^*. For example, the 24-th entry of the decoded vector, which is supposed to belong to the noisy part of the signal, exceeds in absolute value its 13-th entry, although the latter is supposed to belong to the relevant part of the signal. Thus, in potential applications the misinterpretation of results is bound to occur.

The presentation of Example 3.1 gives rise to the need of

1. a proper definition of statistical properties of signal and noise, and
2. new, or modified algorithms which take such statistics into account.

In this chapter, we present two different approaches, in which we draw new decoding strategies that are based on respective statistics of the original signal and noise. We furthermore give theoretical indications why those methods overcome the current state-of-the-art in compressed sensing/sparse recovery in terms of support identification.

In Section 3.1, we define the class of *sparse vectors affected by bounded noise*. Based on this class, we show that the classical ℓ_1-minimization, but also the iteratively re-weighted ℓ_1-minimization (Section 2.4.2), considered one of the most robust in the field, easily fails in the task of an exact support identification. The deep reason of this failure is the lack of selectivity of these algorithms, which are designed to promote not only the sparsity of the recovered signal x^* but also of the recovered noise. We propose a new decoding procedure, combining ℓ_1-minimization as a warm-up step and an additional non-convex method, which enhances the support identification properties of the procedure. The mentioned non-convex method is either the regularized *selective least p-powers* or the well-known iterative hard thresholding (Section 2.4.3.2, IHT-λ). The regularized selective least p-powers allows the principal academic understanding of the use of the statistics of signal and noise, but suffers from a high computational complexity. Instead, using iterative hard thresholding, maintains the computational complexity of state-of-the-art methods. Moreover it is as robust as using regularized selective least p-powers since it is based on the same principle. The content of Section 3.1 was published by the author of this dissertation as leading author in [155], except for the results in Section 3.1.3.1, which are presented in more detail in the dissertation of Marco Artina [8] who is main contributor to the respective results.

In Section 3.2, we generalize the concept of Section 3.1 in the sense that we assume that signal and noise, dependent on their statistical properties, are contained in different (also non-convex) balls. Based on this assumption, we define (non-convex) multi-penalty functionals, which allow a separation of signal and noise. In order to minimize those functionals, we propose an alternating algorithm and show its convergence. The content of Section 3.2 was published by the author of this dissertation in [140], holding the main authorship of the results presented in Subsections 3.2.1 and 3.2.3. The

co-author Valeriya Naumova is leading author of the results that are presented in Subsections 3.2.2.1–3.2.2.3.

We provide in Section 3.3 numerical tests of the methods which are presented in Section 3.1 and 3.2 in a common scope. The section is loosely based on the numerical results in the two above mentioned publications [155, 140]. However, it contains essentially new results, in particular a comparison of both approaches to each other and to the "classical" methods as well as a thoroughly investigated parameter choice for all involved methods.

3.1 Approach 1: Damping Noise-Folding by Non-Convex Methods

We want to embed the involved original signal of Example 3.1 into a specific class of vectors. To this end, let us introduce for $r > \eta > 0$ and $1 \leq k < m$ the class of *sparse vectors affected by bounded noise*,

$$\mathcal{S}^p_{\eta,k,r} := \left\{ x \in \mathbb{R}^N | \#S_r(x) \leq k \text{ and } \sum_{i \in (S_r(x))^c} |x_i|^p \leq \eta^p \right\}, \quad 1 \leq p \leq 2, \tag{3.1}$$

where $S_r(x) := \{i \in \{1, \dots, N\} |\, |x_i| > r\}$ is the index support of the *relevant entries* exceeding in absolute value the threshold r. This class contains all vectors for which at most $1 \leq k < m$ large entries exceed the threshold r in absolute value, while the p-norm of the remaining entries stays below a certain noise level η. We do not specify whether the relevant entries of x are themselves affected by noise. Indeed, as we can at most approximate them anyway with an accuracy, which is never better than the noise level $\eta = \|x_{(S_r(x))^c}\|_{\ell_2}$, see (3.9) below, it is clearly redundant to discuss their exactness or noiseless nature. The definition of the class $\mathcal{S}^p_{\eta,k,r}$ makes redundant the notation $\bar{x} + n$ from the model (2.21) in the introduction. We consider hereafter only the single vector x which contains relevant and noise entries in the sense that $\bar{x} \hat{=} x_{S_r(x)}$ and $n \hat{=} x_{S_r(x)^c}$.

In the remainder of this section, first, we describe the limitations of ℓ_1-minimization when noise on the signal is present, and we perform a very similar analogue analysis for the iteratively re-weighted ℓ_1-minimization (IRL1) based on the results in [141]. Second, we propose the linearly constrained minimization of the regularized selective p-potential functional as an alternative, and show that certain sufficient conditions for recovery indicate significantly better performance than the one provided by ℓ_1-minimization and IRL1. Third, we address the issue of the high computational cost of the regularized selective p-potential optimization and propose, exploiting a similar *selectivity principle*, an alternative method based on iterative hard thresholding.

3.1.1 Support Identification Stability Results in Standard Sparse Recovery

The theory of sparse recovery, as presented in Chapter 2, tells us that we are able to recover by (iteratively re-weighted) ℓ_1-minimization compressible vectors within a certain accuracy, given by (2.5) or (2.48) respectively. If we re-interpret compressible vectors as sparse vectors which are corrupted by noise, we immediately see that the accuracy of the recovered solution is basically driven by the noise level affecting the vector. Nevertheless, neither inequalities (2.5) and (2.48) tell us immediately if the recovered support of the k largest entries of the decoded vector in absolute value is the same as the one of the original signal, nor are we able to identify the large entries exceeding a given threshold in absolute value. In this section, the limitations of ℓ_1-minimization and iteratively re-weighted ℓ_1-minimization are investigated in detail.

Let us informally explain how the class $\mathcal{S}^p_{\eta,k,r}$ can be crucially used to analyze the effects of noise folding in terms of support recovery depending on the parameters η, r, k. Therefore, assume now that $x \in \mathcal{S}^2_{\eta,k,r}$ in expectation (notice that we specified here $p = 2$). By the statistical equivalence of the model (2.21) and (2.7), which was mentioned in the introduction of this chapter and recalled in more detail in Section 2.2.3, we infer that the recovered vector x^* by means of the Dantzig selector will fulfill the following error estimate (compare (2.23)):

$$\begin{aligned} \|x_{S_r(x)} - x^*\|_{\ell_2}^2 &\leq C^2 \cdot 2\log N \cdot \left((1+k)\frac{N}{m}\sigma_n^2\right) \\ &\leq C^2 \cdot 2\log N \cdot \left((1+k)\frac{N}{m}\frac{\eta^2}{N-k}\right), \end{aligned} \tag{3.2}$$

where the last inequality follows by the requirement

$$(N-k)\sigma_n^2 = \mathbb{E}\left(\sum_{i\in(S_r(x))^c} |n_i|^2\right) \leq \eta^2,$$

considering here $S_r(x)$ as previously defined below (3.1). Since we assume that $k \ll N$ and $\frac{k+1}{m} \leq 1$, the right-hand-side of (3.2) can be further bounded from above by

$$C^2 \cdot 2\log N \cdot \left((1+k)\frac{N}{m}\frac{\eta^2}{N-k}\right) \leq C_1^2 \cdot 2\log N \cdot \eta^2.$$

It easily follows (and we will use similar arguments below for different decoding methods) that a sufficient condition for the identification of the relevant entries of x, i.e., $S_r(x) \subset \mathrm{supp}(x^*)$, is

$$C_1^2 \cdot 2\log N \cdot \eta^2 < r^2,$$

or, equivalently,

$$\eta < \frac{r}{C_1\sqrt{2\log N}}.$$

Notice that such a sufficient condition on η actually implies a rather large gap between the significant entries of x and its noise components. Hence, it is of utmost practical interest to understand how small this gap is actually allowed to be, i.e., how large η can be relatively to r, for the most used recovery algorithms in sparse recovery (not only the Dantzig selector) to be able to have both support identification and a good approximation of the relevant entries of x.

For later use, let us denote, for $1 \leq p \leq 2$ and q such that $\frac{1}{p} + \frac{1}{q} = 1$,

$$\kappa_p := \kappa_p(N,k) := \begin{cases} 1, & p = 1, \\ \sqrt[q]{N-k}, & 1 < p \leq 2. \end{cases}$$

The following simple theorem shows how one can estimate the support of the relevant entries of the original signal if we know the support of the ℓ_1-minimizer.

Theorem 3.2
Let $x \in \mathbb{R}^N$ be a noisy signal with k relevant entries and the noise level $\eta \in \mathbb{R}$, $\eta \geq 0$, i.e., for $\Lambda = \operatorname{supp}(x_{[k]})$,

$$\sum_{j \in \Lambda^c} |x_j|^p \leq \eta^p, \tag{3.3}$$

for a fixed $1 \leq p \leq 2$. Consider further an encoder $\Phi \in \mathbb{R}^{m \times N}$ which has the (k, γ_k)-NSP, with $\gamma_k < 1$, the respective measurement vector $y = \Phi x \in \mathbb{R}^m$, and the ℓ_1-minimizer $x^* := \Delta_1(y)$ (see (2.4)). If the i-th component of the original signal x is such that

$$|x_i| > \frac{2(1+\gamma_k)}{1-\gamma_k}\kappa_p\,\eta, \tag{3.4}$$

then $i \in \operatorname{supp}(x^*)$.

Proof. Hölder's inequality applied on the instance optimality property (2.5), and the assumption (3.3) yield the estimate

$$\|x^* - x\|_{\ell_1} \leq \frac{2(1+\gamma_k)}{1-\gamma_k}\sigma_k(x)_{\ell_1} \leq \frac{2(1+\gamma_k)}{1-\gamma_k}\kappa_p\,\eta.$$

We now choose a component $i \in \{1, \ldots, N\}$ such that $|x_i| > \frac{2(1+\gamma_k)}{1-\gamma_k}\kappa_p\,\eta$, and assume $i \notin \operatorname{supp}(x^*)$. This leads to the contradiction:

$$|x_i| = |x_i - x_i^*| \leq \|x - x^*\|_{\ell_1} \leq \frac{2(1+\gamma_k)}{1-\gamma_k}\kappa_p\,\eta < |x_i|.$$

Hence, necessarily $i \in \operatorname{supp}(x^*)$. □

The noise level substantially influences the ability of support identification. Here, the noisy signal should have (as a sufficient condition) the k largest entries in absolute value above

$$r_1 := \frac{2(1+\gamma_k)}{1-\gamma_k}\kappa_p\,\eta,$$

in order to guarantee support identification.

We are able to show a similar support identification result also in the case of the iteratively re-weighted ℓ_1-minimization (see Section 2.4.2, IRL1), as a consequence of the respective instance optimality result in Lemma 2.27.

Theorem 3.3
Let $x \in \mathbb{R}^N$ be a noisy signal with k relevant entries and the noise level $\eta \in \mathbb{R}$, $\eta \geq 0$, i.e., for $\Lambda = \operatorname{supp}(x_{[k]})$,

$$\sum_{j\in\Lambda^c} |x_j|^p \leq \eta^p,$$

for a fixed $1 \leq p \leq 2$. Consider further an encoder $\Phi \in \mathbb{R}^{m\times N}$ which has the $(2k, \delta_{2k})$-RIP, with $\delta_{2k} < \sqrt{2}-1$, the respective measurement vector $y = \Phi x \in \mathbb{R}^m$, and the iteratively re-weighted ℓ_1-minimizer $x^* := \Delta_{1rew}(y)$. If for all $i \in \operatorname{supp}(x_{[k]})$

$$|x_i| > 9.6\frac{\sqrt{1+\delta_{2k}}}{1-(\sqrt{2}+1)\delta_{2k}}\left(1+\frac{\kappa_p}{\sqrt{k}}\right)\eta =: r_{1rew}, \tag{3.5}$$

then $\operatorname{supp}(x_{[k]}) \subset \operatorname{supp}(x^*)$.

Proof. First, notice that

$$\eta \geq \left(\sum_{j\in\Lambda^c} |x_j|^p\right)^{\frac{1}{p}} = \sigma_k(x)_{\ell_p} \geq \sigma_k(x)_{\ell_2},$$

and $\kappa_p\sigma_k(x)_{\ell_p} \geq \sigma_k(x)_{\ell_1}$ by Hölder's inequality. Thus, we have for all $i \in \operatorname{supp}(x_{[k]})$ that

$$\begin{aligned}|x_i| &> 9.6\frac{\sqrt{1+\delta_{2k}}}{1-(\sqrt{2}+1)\delta_{2k}}\left(1+\frac{\kappa_p}{\sqrt{k}}\right)\eta \geq 9.6\frac{\sqrt{1+\delta_{2k}}}{1-(\sqrt{2}+1)\delta_{2k}}\left(\sigma_k(x)_{\ell_p}+\frac{\kappa_p}{\sqrt{k}}\sigma_k(x)_{\ell_p}\right)\\ &\geq 9.6\frac{\sqrt{1+\delta_{2k}}}{1-(\sqrt{2}+1)\delta_{2k}}\left(\sigma_k(x)_{\ell_2}+\frac{\sigma_k(x)_{\ell_1}}{\sqrt{k}}\right).\end{aligned}$$

Consequently, we fulfill the conditions of Lemma 2.27 for which, for all $i \in \operatorname{supp}(x_{[k]})$, $|x_i| > \bar{r}$, as defined in (2.47). Assume now that there is $i \in \operatorname{supp}(x_{[k]})$ and $i \notin \operatorname{supp}(x^*)$.

By Lemma 2.27 we obtain the contradiction

$$|x_i| = |x_i - x_i^*| \leq \|x - x^*\|_{\ell_2} \leq 4.8\frac{\sqrt{1+\delta_{2k}}}{1+(\sqrt{2}-1)\delta_{2k}}\left(\sigma_k(x)_{\ell_2} + \frac{\sigma_k(x)_{\ell_1}}{\sqrt{k}}\right)$$
$$\leq 9.6\frac{\sqrt{1+\delta_{2k}}}{1-(\sqrt{2}+1)\delta_{2k}}\left(\sigma_k(x)_{\ell_2} + \frac{\sigma_k(x)_{\ell_1}}{\sqrt{k}}\right) = \bar{r} < |x_i|.$$

Hence, $i \in \operatorname{supp}(x^*)$. □

Here, the noisy signal should have the k largest entries in absolute value above r_{1rew} in order to guarantee support identification.

3.1.2 Support Identification Stability in the Class of Sparse Vectors Affected by Bounded Noise

In this section, we present results in terms of support discrepancy once we consider two elements of the class $\mathcal{S}^p_{\eta,k,r}$ (defined in (3.1)), having the same measurements. This condition is a basic requirement for the design of a decoder that is supposed to have enhanced support identification properties.

Theorem 3.4
Let $\Phi \in \mathbb{R}^{m\times N}$ have the $(2k,\gamma_{2k})$-NSP, for $\gamma_{2k} < 1$, $1 \leq p \leq 2$, and $x, x' \in \mathcal{S}^p_{\eta,k,r}$ such that $\Phi x = \Phi x'$, and $0 \leq \eta < r$. Then

$$\#(S_r(x)\Delta S_r(x')) \leq \frac{(2\gamma_{2k}\kappa_p\eta)^p}{(r-\eta)^p}. \tag{3.6}$$

(Here we denote by "Δ" the set symmetric difference, not to be confused with the previously introduced symbol of a generic decoder as used in Section 2.1.4. See (3.50) for a detailed definition of the symmetric difference.) If additionally

$$r > \eta(1 + 2\gamma_{2k}\kappa_p) =: r_S, \tag{3.7}$$

then $S_r(x) = S_r(x')$.

Proof. As $\Phi x = \Phi x'$, then $(x - x') \in \mathcal{N}_\Phi$. By the $(2k,\gamma_{2k})$-NSP, Hölder's inequality, and the triangle inequality we have

$$\left\|(x-x')_{S_r(x)\cup S_r(x')}\right\|_{\ell_p} \leq \left\|(x-x')_{S_r(x)\cup S_r(x')}\right\|_{\ell_1} \leq \gamma_{2k}\left\|(x-x')_{(S_r(x)\cup S_r(x'))^c}\right\|_{\ell_1}$$
$$\leq \gamma_{2k}\kappa_p\left\|(x-x')_{(S_r(x)\cup S_r(x'))^c}\right\|_{\ell_p} \leq 2\gamma_{2k}\kappa_p\eta. \tag{3.8}$$

Now we estimate the symmetric difference of the supports of the large entries of x and x' in absolute value as follows: if $i \in S_r(x)\Delta S_r(x')$, then either $|x_i| > r$ and

$|x'_i| \leq \eta$ or $|x_i| \leq \eta$ and $|x'_i| > r$. This implies that $|x'_i - x_i| > (r - \eta)$. Thus we have $\left\|(x - x')_{S_r(x)\Delta S_r(x')}\right\|_{\ell_p}^p \geq (\#(S_r(x)\Delta S_r(x')))\,(r-\eta)^p$. Together with the non-negativity of $\left\|(x - x')_{S_r(x)\cap S_r(x')}\right\|_{\ell_p}$, we obtain the chain of inequalities

$$\begin{aligned}(2\gamma_{2k}\kappa_p\eta)^p &\geq \left\|(x - x')_{S_r(x)\cup S_r(x')}\right\|_{\ell_p}^p \geq \left\|(x - x')_{S_r(x)\cap S_r(x')}\right\|_{\ell_p}^p + \left\|(x - x')_{S_r(x)\Delta S_r(x')}\right\|_{\ell_p}^p \\ &\geq (\#(S_r(x)\Delta S_r(x')))\,(r - \eta)^p,\end{aligned}$$

and therefore we obtain (3.6). Notice now that (3.6) and (3.7) imply $\mathbb{N} \ni \#(S_r(x)\Delta S_r(x')) < 1$ and $S_r(x)\Delta S_r(x') = \emptyset$. □

Remark 3.5
One additional implication of this latter theorem is that we can give a bound on the difference of x and x' restricted to the relevant entries. Indeed, in case of unique identification of the relevant entries, i.e., $\Lambda := S_r(x) = S_r(x')$ we obtain, by the inequality (3.8), that

$$\left\|(x - x')_\Lambda\right\|_{\ell_1} \leq 2\gamma_k\kappa_p\eta. \tag{3.9}$$

Notice that we replaced γ_{2k} by $\gamma_k \leq \gamma_{2k}$, because now $\#\Lambda \leq k$.

Unfortunately, we are not able to provide the *necessity of the gap conditions* (3.4), (3.5), (3.7) for successful support recovery, simply because we lack optimal deterministic error bounds in general: one way of producing a lower bound would be to construct for each algorithm a counterexample, for which a certain gap condition is violated and recovery of support fails. Since most of the algorithms we shall illustrate below are iterative, it is likely extremely difficult to provide such explicit counterexamples. Therefore, we limit ourselves here to discuss the discrepancies of r_1 and r_S and of $r_{1\text{rew}}$ and r_S. We shall see in the numerical experiments that the *sufficient gap conditions* (3.4), (3.5), (3.7) nevertheless provide actual indications of performance of the algorithms.

The gap between the two thresholds r_1, r_S is given by

$$r_1 - r_S = \left(2\left(\frac{1+\gamma_k}{1-\gamma_k} - \gamma_{2k}\right)\kappa_p(N,k) - 1\right)\eta.$$

As $\gamma_{2k} < 1 < \frac{1+\gamma_k}{1-\gamma_k}$ and $\kappa_p(N,k)$ is very large for $N \gg k$ and $p > 1$, this *positive* gap is actually very large, for $N \gg 1$.

The gap between the two thresholds $r_{1\text{rew}}$, r_S is given by

$$r_{1\text{rew}} - r_S = \left(9.6\frac{\sqrt{1+\delta_{2k}}}{1-(\sqrt{2}+1)\delta_{2k}}\left(1 + \frac{\kappa_p}{\sqrt{k}}\right) - (1 + 2\gamma_{2k}\kappa_p)\right)\eta.$$

By Lemma 2.9, a matrix Φ having the $(2k, \delta_{2k})$-RIP has also the $(2k, \gamma_{2k})$-NSP with $\gamma_{2k} = \frac{\sqrt{2}\delta_{2k}}{1-(\sqrt{2}+1)\delta_{2k}}$, which, substituted into the above equation, yields

$$r_{1\mathrm{rew}} - r_S = \frac{\left(\frac{9.6\sqrt{1+\delta_{2k}}}{\sqrt{k}} - 2\sqrt{2}\delta_{2k}\right)\kappa_p + \left[9.6\sqrt{1+\delta_{2k}} - (1-(\sqrt{2}+1)\delta_{2k})\right]}{1-(\sqrt{2}+1)\delta_{2k}}\,\eta.$$

Since $0 < \delta_{2k} < \sqrt{2}-1$, we have $0 < 1-(\sqrt{2}+1)\delta_{2k} < 1$, and therefore the denominator and the right summand in the numerator are positive. The left summand of the numerator is positive and very large as soon as $k < \left(\frac{9.6\sqrt{1+\delta_{2k}}}{2\sqrt{2}\delta_{2k}}\right)^2$. Thus, even in the limiting scenario where $\delta_{2k} \approx \sqrt{2}-1$, we still have $k \leq 94$, which may be considered sufficient for a wide range of applications. A more sophisticated estimate of the above term can actually reveal even less restrictive bounds on k. Thus, in general, since k and δ_{2k} are small, also the left summand is positive. We conclude again that the gap is large for $N \gg k$ and $p > 1$.

Unfortunately, the discrepancies $r_1 - r_S \gg 0$ and $r_{1\mathrm{rew}} - r_S \gg 0$ cannot be amended because in general the ℓ_1-minimization decoder Δ_1 and the iteratively re-weighted ℓ_1-minimization decoder $\Delta_{1\mathrm{rew}}$ do *not* have the property of decoding a vector in the class $\mathcal{S}^p_{\eta,k,r}$, even if the original vector x belongs to it, i.e., in general the implication

$$x \in \mathcal{S}^p_{\eta,k,r} \Rightarrow \{\Delta_1(\Phi x), \Delta_{1\mathrm{rew}}(Ax)\} \ni x^* \in \mathcal{S}^p_{\eta,k,r} \tag{3.10}$$

does *not* hold for these decoders. These ineliminable limitations of Δ_1 and $\Delta_{1\mathrm{rew}}$ can be verified, e.g., in Example 3.1, where (3.10) does not hold for the ℓ_1-minimizer.

3.1.3 Non-convex Methods for Enhanced Support Identification Properties

To overcome the shortcomings of methods based exclusively on ℓ_1-minimizations in

1. damping the noise-folding, and consequently in
2. having a stable support recovery,

in this section, we present the design and the properties of two decoding procedures, with output in $\mathcal{S}^p_{\eta,k,r}$, which consequently allow us to have both these very desirable properties.

3.1.3.1 Properties of the Regularized Selective p-potential Functional (SLP)

Let us first introduce the following functional.

Definition 3.6 (Regularized selective p-potential)
We define the *regularized truncated p-power function* $W_r^{p,\epsilon}\colon \mathbb{R} \to \mathbb{R}_0^+$ by

$$W_r^{p,\epsilon}(t) = \begin{cases} t^p & 0 \le t < r-\epsilon, \\ \pi_p(t) & r-\epsilon \le t \le r+\epsilon, \\ r^p & t > r+\epsilon, \end{cases} \qquad t \ge 0,$$

where $0 < \varepsilon < r$, and $\pi_p(t)$ is the third degree interpolating polynomial

$$\pi_p(t) := A(t-s_2)^3 + B(t-s_2)^2 + C,$$

and $C = \mu_3$, $B = \frac{\mu_1}{s_2-s_1} - \frac{3(\mu_3-\mu_2)}{(s_2-s_1)^2}$, $A = \frac{\mu_1}{3(s_2-s_1)^2} + \frac{2B}{3(s_2-s_1)}$, where $s_1 = (r-\varepsilon)$, $s_2 = (r+\varepsilon)$, $\mu_1 = p(r-\varepsilon)^{p-1}$, $\mu_2 = (r-\varepsilon)^p$, and $\mu_3 = r^p$. Moreover, we set $W_r^{p,\epsilon}(t) = W_r^{p,\epsilon}(-t)$ for $t < 0$. We call the functional $\mathcal{SP}_r^{p,\epsilon}\colon \mathbb{R}^N \to \mathbb{R}_0^+$,

$$\mathcal{SP}_r^{p,\epsilon}(x) = \sum_{j=1}^{N} W_r^{p,\epsilon}(x_j), \quad r > 0, \quad 1 \le p \le 2,$$

the *regularized selective p-potential (SP) functional*.

The graphs of $W_r^{p,0}$ and $W_r^{p,\epsilon}$ are shown in Figure 3.2 for $p = 2$, $r = 1$, and $\epsilon = 0.4$, see [9, 79] for related literature and further details in statistical signal processing.

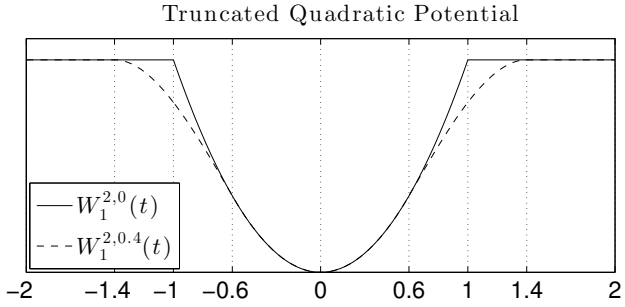


Figure 3.2: Truncated quadratic potential $W_1^{2,0}$ and its regularization $W_1^{2,0.4}$ (dashed).

The deep reason, why we defined the potential $\mathcal{SP}_r^{p,\epsilon}$ is, that it is "selective". If a component is large, it is simply counted, if instead a component is relatively small it is damped and encoded as noise to be filtered. The reasonable hope is that a solution x^* of the problem

$$x^* := \operatorname*{arg\,min}_{z \in \mathcal{F}_\Phi(y)} \mathcal{SP}_r^{p,\epsilon}(z) \tag{3.11}$$

is a member of the class $\mathcal{S}_{\eta,k,r}^p$ and has therefore enhanced support identification properties (compare Theorem 3.4). Indeed, we show this statement in the following

Theorem 3.7, by the even weaker condition (3.13), which only requires a vector x^* whose functional value is less or equal to the functional value of the original signal x. However, (3.11) will make (3.13) automatically true, whichever x is, but at the same time (3.11) is a highly nonconvex problem whose solution is in general NP-hard [4]. The way we will circumvent this drawback is to employ an iterative algorithm, which we call *selective least p-powers (SLP)*, that we present in the remainder of this section.

Theorem 3.7
Let $\Phi \in \mathbb{R}^{m\times N}$ have the $(2k,\gamma_{2k})$-NSP, with $\gamma_{2k}<1$, and $1\le p\le 2$. Furthermore, we assume $x\in\mathcal{S}^p_{\eta,k,r+\epsilon}$, for $\epsilon>0$, $0<\eta<r+\epsilon$, with the property of having the minimal $\#S_{r+\epsilon}(x)$ within $\mathcal{F}_\Phi(y)$, where $y=\Phi x$ is its associated measurement vector, i.e.,

$$\#S_{r+\epsilon}(x)\le \#S_{r+\epsilon}(z) \text{ for all } z\in\mathcal{F}_\Phi(y). \tag{3.12}$$

If x^* is such that

$$\mathcal{SP}^{p,\epsilon}_r(x^*)\le \mathcal{SP}^{p,\epsilon}_r(x), \tag{3.13}$$

and

$$|x^*_i|<r-\epsilon, \tag{3.14}$$

for all $i\in(S_{r+\epsilon}(x^*))^c$, then also $x^*\in\mathcal{S}^p_{\eta,k,r+\epsilon}$, implying noise-folding damping. Moreover, we have the support stability property

$$\#(S_{r+\epsilon}(x)\Delta S_{r+\epsilon}(x^*))\le \frac{(2\gamma_{2k}\kappa_p\eta)^p}{(r+\epsilon-\eta)^p}. \tag{3.15}$$

Proof. Notice that we can equally rewrite the $\mathcal{SP}^{p,\epsilon}_r$ functional as $\mathcal{SP}^{p,\epsilon}_r(z)=r^p\#S_{r+\epsilon}(z)+\sum_{i\in(S_{r+\epsilon}(z))^c}|z_i|^p_\epsilon$, where $|t|^p_\epsilon:=W^{p,\epsilon}_r(t)$ for $|t|\le r+\epsilon$. Here, by construction, we have $|t|^p_\epsilon\le|t|^p$. By the assumptions (3.13) and $x\in\mathcal{S}^p_{\eta,k,r+\epsilon}$, we obtain the estimates

$$\begin{aligned} r^p\#S_{r+\epsilon}(x^*)\le \mathcal{SP}^{p,\epsilon}_r(x^*)\le \mathcal{SP}^{p,\epsilon}_r(x) &= r^p\#S_{r+\epsilon}(x)+\sum_{i\in(S_{r+\epsilon}(x))^c}|x_i|^p_\epsilon \\ &\le r^p\#S_{r+\epsilon}(x)+\sum_{i\in(S_{r+\epsilon}(x))^c}|x_i|^p\le r^p\#S_{r+\epsilon}(x)+\eta^p, \end{aligned}$$

and thus $\#S_{r+\epsilon}(x^*)\le(\frac{\eta}{r})^p+\#S_{r+\epsilon}(x)$. As $\frac{\eta}{r}<1$ by assumption, the minimality property (3.12) yields immediately

$$\#S_{r+\epsilon}(x^*)=\#S_{r+\epsilon}(x)\le k. \tag{3.16}$$

Assumption (3.14) and again (3.13) yield

$$\begin{aligned} r^p\#S_{r+\epsilon}(x^*)+\sum_{i\in(S_{r+\epsilon}(x^*))^c}|x^*_i|^p &= r^p\#S_{r+\epsilon}(x^*)+\sum_{i\in(S_{r+\epsilon}(x^*))^c}|x^*_i|^p_\epsilon \\ \le r^p\#S_{r+\epsilon}(x)+\sum_{i\in(S_{r+\epsilon}(x))^c}|x_i|^p_\epsilon &\le r^p\#S_{r+\epsilon}(x)+\sum_{i\in(S_{r+\epsilon}(x))^c}|x_i|^p. \end{aligned}$$

By this latter inequality and (3.16) we obtain

$$\sum_{i\in(S_{r+\epsilon}(x^*))^c} |x_i^*|^p \leq \sum_{i\in(S_{r+\epsilon}(x))^c} |x_i|^p \leq \eta^p,$$

which implies $x^* \in \mathcal{S}^p_{\eta,k,r+\epsilon}$. We conclude (3.15) by an application of Theorem 3.4. □

Remark 3.8
Let us comment the assumptions of the latter result.

(i) The assumption that x is actually the vector with minimal essential support $S_r(x)$ among the feasible vectors in $\mathcal{F}_\Phi(y)$ corresponds to the request of being the "simplest" explanation to the data;

(ii) As we already mentioned above, the best candidate x^* to fulfill condition (3.13) would be actually a solution of (3.11). In the follow-up paragraph, we present the selective least p-powers (SLP) algorithm to compute x^*, performing a local minimization of $\mathcal{SP}^{p,\epsilon}_r$ in $\mathcal{F}_\Phi(y)$ around a given starting vector x_0, see Algorithm 7. Ideally, the best choice for x_0 would be x itself, so that (3.13) may be fulfilled. As we do not dispose yet of the original vector x, a heuristic rule, which we will show to be very robust in our numerical simulations, is to choose the ℓ_1-minimizer $x_0 = \Delta_1(y) \approx x$. The reasonable hope is that actually $\mathcal{SP}^{p,\epsilon}_r(x^*) \leq \mathcal{SP}^{p,\epsilon}_r(\Delta_1(y)) \approx \mathcal{SP}^{r,\epsilon}_r(x)$. We dedicate the last paragraph in this subsection to such a particular warm-up step;

(iii) The assumption that the outcome x^* of the algorithm has additionally the property $|x_i^*| < r - \epsilon$, for all $i \in (S_{r+\epsilon}(x^*))^c$ is justified by observing that in the actual implementation x^* will be the result of a thresholding operation, i.e., $x_i^* = \mathbb{S}^\mu_p(\xi_i)$, for $i \in (S_{r+\epsilon}(x^*))^c$, where $\mathbb{S}^\mu_p$ is defined as in [9, Formula 3.36]. The particularly steep shape of the thresholding functions $\mathbb{S}^\mu_p$ in the interval $[r-\epsilon, r+\epsilon]$, especially for $p = 2$, see [9, Figure 3.3 (c)], makes it highly unlikely for ϵ sufficiently small that $r - \epsilon \leq |x_i^*|$ for $i \in (S_{r+\epsilon}(x^*))^c$.

The Algorithm SLP It remains to formulate the algorithm (SLP), by which we are able to solve the nonconvex and nonsmooth optimization problem (3.11). To this end, we recall a novel and very robust algorithm for linearly constrained nonconvex and nonsmooth minimization, introduced and analyzed first in [9]. The algorithm is particularly suited for our purpose, since it only requires a C^1-regular functional. This distinguishes it from other well-known methods such as SQP and Newton methods, which require a more restrictive C^2-regularity. All notions and results written in this section are collected more in general in [9] and with a higher level of detail in [7]. Nevertheless we report them directly adapted to our specific case in order to have a

simplified and more immediate application.

The starting values $x_0 = x_{(0,0)} \in \mathbb{R}^N$ and $q_{(0,0)} \in \mathbb{R}^m$ are taken arbitrarily. For a fixed scaling parameter $\lambda > 0$ and an adaptively chosen sequence of integers $(L_\ell)_{\ell\in\mathbb{N}}$, we formulate Algorithm 7.

Algorithm 7 SLP

while $\|x_{\ell-1} - x_\ell\|_{\ell_2} \leq \mathrm{TOL}$ **do**
  $x_{(\ell,0)} = x_{\ell-1} := x_{(\ell-1,L_{\ell-1})}$
  $q_{(\ell,0)} = q_{\ell-1} := q_{(\ell-1,L_{\ell-1})}$
  for $k = 1, \ldots, L_\ell$ **do**
    $x_{(\ell,k)} = \underset{x\in\mathbb{R}^N}{\arg\min} \left(\mathcal{SP}^{p,\epsilon}_{\omega,x_{\ell-1}}(x) - \langle q_{(\ell,k-1)}, \Phi x\rangle + \lambda \|\Phi x - y\|^2_{\ell_2}\right)$
    $q_{(\ell,k)} = q_{(\ell,k-1)} + 2\lambda(y - \Phi x_{(\ell,k)})$
  end for
end while

Obviously, the functional $\mathcal{SP}^{p,\epsilon}_{\omega,x_{\ell-1}}$, which appears in the algorithm, has not yet been defined. In order to understand its definition, we need to introduce the concept of ν-convexity, which plays a key-role in the minimization process. In fact, the Bregman-like inner loop of Algorithm 7 requires this property to converge with an a priori rate.

Definition 3.9 (ν-convexity)
A function $f : \mathbb{R}^N \to \mathbb{R}$ is ν-convex if there exists a constant $\nu > 0$ such that for all $x, x' \in \mathbb{R}^N$ and $\phi \in \partial f(x), \psi \in \partial f(x')$

$$\langle \phi - \psi, x - x'\rangle \geq \nu \|x - x'\|^2_{\ell_2},$$

where ∂f is the subdifferential of the function f (see (2.16)).

By this definition, we can introduce

$$\mathcal{SP}^{p,\epsilon}_{\omega,x'}(x) := \mathcal{SP}^{p,\epsilon}_r(x) + \omega \|x - x'\|^2_{\ell_2},$$

where ω is chosen such that the new functional is ν-convex, in order to have the convergence of the algorithm. The finite adaptively chosen number of inner loop iterates L_ℓ is defined by the condition

$$(1 + \|q_{\ell-1}\|_{\ell_2}) \left\|\Phi x_{(\ell,L_\ell)} - y\right\|_{\ell_2} \leq \frac{1}{\ell^\alpha},$$

for a given parameter $\alpha > 1$, which in our numerical experiments will be set to $\alpha = 1.1$. We refer to [9, Section 2.2] for details on the finiteness of L_ℓ and for the proof of

convergence of Algorithm 7 to critical points of $\mathcal{SP}_r^{p,\epsilon}$ in $\mathcal{F}_\Phi(y)$.

In the formulation of Algorithm 7 we do not yet specify how to minimize the convex functional

$$\mathcal{SP}^{p,\epsilon}_{\omega,x_{\ell-1}}(x) - \langle q_{(\ell,k-1)}, \Phi x\rangle + \lambda \left\|\Phi x - y\right\|_{\ell_2}^2,$$

in the inner loop. For that we can use an iterative thresholding algorithm introduced in [9, Section 3.7], inspired by the previous work [79] for the corresponding *unconstrained* optimization of regularized selective p-potentials. This method ensures the convergence to a minimizer and is extremely agile to be implemented, as it is based on matrix-vector multiplications and very simple componentwise nonlinear thresholdings. By the iterative thresholding algorithm, we actually equivalently minimize the functional

$$\mathcal{SP}^{p,\epsilon}_{\omega,x'}(x,q) = \mathcal{SP}^{p,\epsilon}_{\omega,x'}(x) + \lambda \left\|\Phi x - (y+q)\right\|_{\ell_2}^2,$$

where we set $\lambda = \frac{1}{2}$ only for simplicity. The respective thresholding functions $\mathbb{S}_p^\mu$ are defined in [9, Lemma 3.13], and in [9, Figure 3.3] their typical shape for different choices of $p \in \{1, 3/2, 2\}$ is shown. Through those thresholding functions, the minimizing algorithm in the inner loop is given by the componentwise fixed point iteration, for $n \geq 0$, $i = 1, \ldots, N$,

$$x_i^{n+1} = \mathbb{S}_p^\mu\left(\left\{\frac{1}{2}\left[(I - \frac{1}{2}\Phi^*\Phi) + (1-\omega)I\right]x^n + \frac{1}{2}\Phi^*(y+q) + \omega x'\right\}_i\right). \tag{3.17}$$

We refer to [9, Theorem 3.15] for the convergence properties of this algorithm.

In summary it can be stated that Algorithm 7 can be realized in practice by nesting three loops. One external loop makes slowly vanishing the quadratic convexification, the second external loop updates the Lagrange multipliers $q_{(\ell,k)}$ for a fixed quadratic convexification, and the final inner loop implements (3.17). Although being an effective method for the minimization of such C^1-regular functions, the efficiency of this algorithm, which is composed of three nested loops is rather low.

Choosing ℓ_1-minimization as a warm up In Remark 3.8 (ii), we mentioned that the Algorithm SLP finds only a critical point of the functional $\mathcal{SP}_r^{p,\epsilon}$, so the condition $\mathcal{SP}_r^{p,\epsilon}(x^*) \leq \mathcal{SP}_r^{p,\epsilon}(x)$ (3.13) used in the proof of Theorem 3.7 may not be always valid. In order to enhance the chance of validity of this condition, the choice of an appropriate starting point is crucial. As we know that the ℓ_1-minimization converges to its global minimizer with at least some guarantees given by Theorem 3.2, we use the result of this minimization process as a warm up to select the starting point of Algorithm 7. In the following, we distinguish between SLP which starts at $x_0 = 0$ and ℓ_1+SLP which starts at the ℓ_1-minimizer.

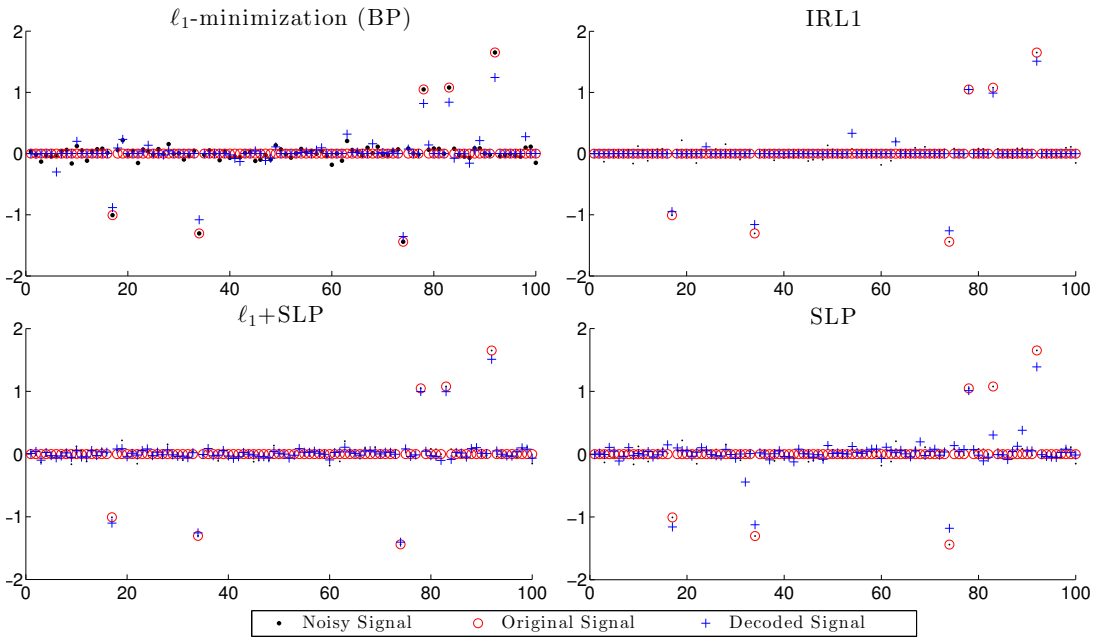


Figure 3.3: The figure reports the results of four different decoding processes (+) of the same test problem where the circles (o) represent the original signal and the points (·) represent the original signal corrupted by the noise.

In Figure 3.3, we illustrate by the results of a single experiment the robustness of ℓ_1+SLP (bottom left subfigure) in comparison to the ℓ_1-minimization based methods and SLP (starting at 0). Here, SLP converged to a feasible critical point, but it is quite evident that the decoding process failed since the large entry at position 83 (signal) was badly recovered and even the entry at position 89 (noise) is larger. If we look at the ℓ_1-minimization result (top left subfigure), the minimization process brings us close to the solution, but the results still significantly lack accuracy. By ℓ_1+SLP we obtain a good approximation of the relevant entries of the original signal and we get a significant correction and an improved recovery. Also IRL1 improves the result of ℓ_1-minimization significantly, but still approximates the large entries worse than ℓ_1+SLP. Although the difference is minor, we observe another important aspect of IRL1: the noise part is sparsely recovered, while ℓ_1+SLP distributes the noise in a more uniform way in a much smaller stripe around zero. This drawback of IRL1 can be crucial when it comes to the distinction of the relevant entries from noise.

In order to confirm these observations, we will return to the comparison of ℓ_1-SLP, SLP, and ℓ_1-based methods in the scope of massive tests on multiple datasets in

Section 3.3.

3.1.3.2 Properties of Iterative Hard Thresholding (IHT-λ)

As already mentioned above, the numerical realization of the algorithm SLP turns out to be computationally demanding as soon as the dimension N gets large. Since the computational time is a crucial point when it comes to practical applications, we show that the method IHT-λ, which is introduced in Section 2.4.3.2, shows similar support identification properties as SLP while being very efficient. In Theorem 3.10, we discuss under which sufficient conditions this method is able to exactly identify $S_r(x)$.

Theorem 3.10
Assume $\Phi \in \mathbb{R}^{m\times N}$ to have the $(2k, \delta_{2k})$-RIP, with $\delta_{2k} < 1$, $\|\Phi\| \leq 1$, and define $\beta(\Phi) > 0$ as in (2.53). Let $x \in \mathcal{S}^p_{\eta,k,r}$, for a fixed $1 \leq p \leq 2$, and $y = \Phi x$ the respective measurements. Assume further

$$r > \eta\left(1 + \frac{1}{1-\delta_{2k}}\left(1 + \frac{1}{\beta(\Phi)}\right)\right), \tag{3.18}$$

and define λ such that

$$\eta < \sqrt{\lambda} < \frac{r - \frac{\eta}{1-\delta_{2k}}}{1 + \frac{1}{(1-\delta_{2k})\beta(\Phi)}}. \tag{3.19}$$

Let x^* be the limit of the sequence generated by Algorithm 6 (IHT-λ), and we assume

$$\mathcal{J}_0(x^*) \leq \mathcal{J}_0(x_{S_r(x)}). \tag{3.20}$$

Then $\Lambda := S_r(x) = \operatorname{supp}(x^*)$, and it holds

$$|x_i - x_i^*| < r - \sqrt{\lambda}, \text{ for all } i \in \Lambda. \tag{3.21}$$

Proof. Assume $\#\operatorname{supp}(x^*) > \#S_r(x) = k$. By (3.20), we have that

$$\begin{aligned}
0 &< \#\operatorname{supp}(x^*) - \#\operatorname{supp}(x_{S_r(x)}) = \#\operatorname{supp}(x^*) - \#S_r(x) \\
&\leq \frac{1}{\lambda}\left(\left\|\Phi\left(x_{S_r(x)}\right) - y\right\|_{\ell_2}^2 - \|\Phi x^* - y\|_{\ell_2}^2\right) \leq \frac{1}{\lambda}\left\|\Phi\left(x_{S_r(x)}\right) - y\right\|_{\ell_2}^2 \\
&= \frac{1}{\lambda}\left\|\Phi\left(x_{(S_r(x))^c}\right)\right\|_{\ell_2}^2 \leq \frac{1}{\lambda}\|\Phi\|^2\left\|x_{(S_r(x))^c}\right\|_{\ell_2}^2 \leq \frac{\eta^2}{\lambda} < 1,
\end{aligned}$$

where the last inequality follows by (3.19). Since $(\#\operatorname{supp}\left(x_{S_r(x)}\right) - \#\operatorname{supp}(x^*)) \in \mathbb{N}$, the upper inequality yields to a contradiction. Thus $\#\operatorname{supp}(x^*) \leq \#S_r(x) = k$ and therefore x^* and $x_{S_r(x)}$ are both k-sparse, and $\left(x^* - x_{S_r(x)}\right)$ is $2k$-sparse. Under the assumptions of this theorem, we can apply Theorem 2.30 to obtain

$$\|\Phi x^* - y\|_{\ell_2} \leq \frac{\sqrt{\lambda}}{\beta(\Phi)}.$$

In addition to this latter estimate, we use the RIP, the sparsity of $x^* - x_{S_r(x)}$, and (3.19) to obtain for all $i \in \{1, \dots, N\}$ that

$$\begin{aligned}
&\left|\left(x_{S_r(x)}\right)_i - x_i^*\right| \leq \left\|\left(x_{S_r(x)}\right) - x^*\right\|_{\ell_2} \leq \frac{\left\|\Phi(x_{S_r(x)} - x^*)\right\|_{\ell_2}}{1-\delta_{2k}} \\
&\leq \frac{\|\Phi(x-x^*)\|_{\ell_2} + \left\|\Phi\left(x_{(S_r(x))^c}\right)\right\|_{\ell_2}}{1-\delta_{2k}} \leq \frac{\|y-\Phi x^*\|_{\ell_2} + \left\|\Phi\left(x_{(S_r(x))^c}\right)\right\|_{\ell_2}}{1-\delta_{2k}} \\
&\leq \frac{\sqrt{\lambda}}{\beta(\Phi)\,(1-\delta_{2k})} + \frac{\eta}{1-\delta_{2k}} < r - \sqrt{\lambda}.
\end{aligned}$$

Assume now that there is $\tilde{i} \in \mathbb{N}$ such that $\tilde{i} \in S_r(x)$ and $\tilde{i} \notin \operatorname{supp}(x^*)$. But then we would also have $|x_{\tilde{i}} - x^*_{\tilde{i}}| = |x_{\tilde{i}}| > r$, which leads to a contradiction. Thus, $S_r(x) \subset \operatorname{supp}(x^*)$, which together with $\#\operatorname{supp}(x^*) \leq \#S_r(x)$ conclude the proof. □

Remark 3.11
Let us discuss some of the assumptions and implications of this latter result.

(i) Since iterative hard thresholding only computes a local minimizer of $\mathcal{J}_0$, condition (3.20) may not be always fulfilled for any given initial iteration x^0. Similarly to the argument in Remark 3.8 (ii), using the ℓ_1-minimizer as the starting point x^0, or equivalently choosing the vector x^0 as composed of the entries of $\Delta_1(\Phi x)$ exceeding $\sqrt{\lambda}$ in absolute value, we may allow us to approach a local minimizer which fulfills (3.20).

(ii) Condition (3.18) is comparable to the one derived in (3.7). If Φ is "well-conditioned", i.e., we have that $(1-\delta_{2k}) \sim 1$, and $\beta(\Phi) \sim 1$, then

$$1 + \frac{1}{1-\delta_{2k}}\left(1 + \frac{1}{\beta(\Phi)}\right) \sim 3.$$

Note, that in contrast to Theorem 3.7, in the latter theorem, we obtain a result on stable support identification without the use of Theorem 3.4. Thus, we do not automatically obtain the estimate (3.9) on the relevant entries, which is a direct consequence of Theorem 3.4, but only the very poor error estimate (3.21), which is not satisfactory. However, the main goals of this section also involved an accurate reconstruction of the relevant entries of the original signal x. In order to overcome this drawback, we have to meet the assumptions of Theorem 3.4, i.e., the conditions $x^* \in \mathcal{S}^p_{\eta,k,r}$, and $\Phi x = \Phi x^*$, which are, so far, in general not fulfilled. In order to obtain a modification x^{**} of x^* that satisfies these conditions, an additional correction

is necessary. It is a natural approach to determine the vector x^{**} as the solution of

$$\begin{aligned} \min_{z\in\mathbb{R}^N} \quad & \|\Phi z - y\|_{\ell_2}^2 \\ \text{s.t.} \quad & \|z_{\Lambda^c}\|_{\ell_p} \leq \eta, \\ & |z_i| \geq r, \text{ for all } i \in \Lambda, \end{aligned} \tag{3.22}$$

being $\Lambda = S_r(x) = \operatorname{supp}(x^*)$ the support already identified.[1] Since the original signal x fulfills $\Phi x - y = 0$, and $x \in \mathcal{S}^p_{\eta,k,r}$, it is actually a solution of problem (3.22). Thus, we conclude that for any minimizer x^{**} of problem (3.22) the objective function equals zero, thus $\Phi x = \Phi x^{**}$ and, simultaneously, $x^{**} \in \mathcal{S}^p_{\eta,k,r}$. The optimization (3.22) is in general nonconvex, but we can easily recast it in an equivalent convex one: Since $|x_i - x_i^*| < r - \sqrt{\lambda}$, and $|x_i| > r$, we know that the relevant entries of x and x^* have the same sign. Since we are searching for solutions which are close to x, the second inequality constraint becomes $\operatorname{sign}(x_i^*)z_i \geq r$, for all $i \in \Lambda$. Together with the equivalence of ℓ_2- and ℓ_p-norm, we rewrite problem (3.22) as

$$\begin{aligned} \min_{z\in\mathbb{R}^N} \quad & \frac{1}{2} z^*(\Phi^*\Phi)z - y^*\Phi z \\ \text{s.t.} \quad & z^*P_0 z - (N-k)^{1-\frac{2}{p}}\eta^2 \leq 0, \\ & z^*P_j z - (\operatorname{sign}(x^*_{i_j})e_{i_j})^* z + r \leq 0, \\ & \qquad \text{for all } i_j \in \Lambda \text{ , } j = 1, \ldots, \#\Lambda, \end{aligned} \tag{3.23}$$

where $P_0 \in \mathbb{R}^{N\times N}$ is defined componentwise by

$$(P_0)_{r,s} := \begin{cases} 1 & \text{if } r = s \in \Lambda, \\ 0 & \text{else,} \end{cases}$$

and $P_j = 0$, $j = 1, \ldots \#\Lambda$. Since $\Phi^*\Phi$, P_0, and P_j, $j = 1, \ldots, \#\Lambda$, are semi-definite, problem (3.23) is a *convex quadratically constrained quadratic program* (QCQP) which can be efficiently solved by standard methods, e.g., interior point methods [147]. Since we combine here three very efficient methods (ℓ_1-minimization, IHT-λ, and a QCQP), the resulting procedure is much faster than the computation of SLP while, as we will show in Section 3.3, keeping similar support identification properties.

3.1.3.3 Summary: The Selectivity Principle

The decoders based on ℓ_1-minimization and iteratively re-weighted ℓ_1-minimization prefer sparse solutions and have the undesirable effect of sparsifying also the noise.

[1] This decoder is slightly different from (2.24) since not only the information of the already identified support but also the threshold r is taken into account.

Thus all the noise may concentrate on fewer entries which, by a balancing principle, may eventually exceed in absolute value the smallest entries of the actual original signal (compare Example 3.1). This makes it impossible to separate the relevant entries of the signal from the noise by only knowing the threshold r which bounds the relevant entries from below. On the contrary, the two methods, presented in this section, follow a *selectivity* principle, where the recovery process focuses on the extraction of the relevant entries, while uniformly distributing the noise elsewhere, allowing for a clear distinction between those two parts.

3.2 Approach 2: Multi-Penalty Regularization

In order to recover from a data vector y a sparse signal u and a non-sparse noise v which are measured according to the model (2.21), i.e., $y = \Phi(\bar{x} + n)$, we use in this section the approach

$$\arg\min_{x,z} \|\Phi(x+z) - y\|_{\ell_2} + \lambda_p \|x\|_{\ell_p}^p + \lambda_q \|z\|_{\ell_q}^q,$$

where the p-(quasi-)norm and q-(quasi-)norm, with $p, q \in \mathbb{R}$, $0 < p, q \leq \infty$, are used in order to promote different statistical characteristics of x and z respectively. In particular, a choice of $p \leq 1$ promotes sparsity in x, and $q > 1$ an equal distribution of the entries in z. Thus, a minimizing pair (x^*, z^*) of the above problem is considered as the reconstruction of the pair $(\bar{x}, n)$. This approach is called *multi-penalty regularization/optimization*. At this point we only use the above imprecise formulation for the sake of a brief conceptual introduction. The reader is referred to the end of this section for the identification of a proper mathematical setting. There, we will also change from the "(x, z)" to a "(u, v)" notation since we change to an infinite dimensional setting.

Although the formulation of such optimization problems is not at all new, as we shall recall below several known results associated to it, based on the findings in [140], we elaborate in the following two relevant new contributions to the field:

1. We propose an iterative alternating algorithm to perform the minimization of a multi-penalty functional by means of simple iterative thresholding steps, whose analysis required a careful adaptation of several previously published techniques on single-parameter regularization because of the potential non-convexity of the functional for $0 < p < 1$;

2. We systematically investigated in the employment of high-dimensional data analysis methods to classify parameters (λ_p, λ_q, p, q).

To the best of our knowledge, we are the first to provide an explicit direct mechanism for the minimization of the multi-penalty functional with non-convex and non-smooth terms. We also highlight its improved accuracy power with respect to more traditional

one-parameter regularizations, as we can see in the numerical tests, which are presented in Section 3.3.

Perhaps as one of the earliest examples of multi-penalty optimization in imaging, we may refer to the one seminal work of Meyer [134], where the combination of the Sobolev space of distributions with negative exponent -1 and the space of bounded variation functions has been used in image reconstruction towards a proper recovery and separation of texture and cartoon image components; we refer also to the follow up papers by Vese and Osher [187, 186]. Also in the framework of multi-penalty sparse regularization, one needs to look at the early work on signal separation by means of ℓ_1-ℓ_1 norm minimizations in the seminal papers of Donoho et al. on the incoherency of Fourier basis and the canonical basis [56, 54]. We mention also the recent work [55], where the authors consider again ℓ_1-ℓ_1 penalization with respect to curvelets and wavelets to achieve separation of curves and point-like features in images. Daubechies and Teschke built on the works [134, 187, 186] providing a sparsity based formulation of the simultaneous image decomposition, deblurring, and denoising [46], by using multi-penalty ℓ_1- and weighted-ℓ_2-norm minimization. The work by Bredies and Holler [21] analyses the regularization properties of the total generalized variation functional for general symmetric tensor fields and provides convergence for multiple parameters for a special form of the regularization term. In more recent work [111], the infimal convolution of total variation type functionals for image sequence regularization has been considered, where an image sequence is defined as a function on a three dimensional space time domain. The motivation for such kind of functionals is to allow suitably combined spatial and temporal regularity constraints. We emphasize also the two recent conceptual papers [139, 76], where the potential of multi-penalty regularization to outperform single-penalty regularization has been discussed and theoretically proven in the Hilbert space settings.

It is worthwhile mentioning that in recent years both regularization with non-convex constraints (see [23, 109, 114, 195, 158] and references therein) and multi-penalty regularization, e.g., [129, 139], have become the focus of interest in several research communities. While in most of the literature these two directions are considered separately, there have also been some efforts to understand regularization and convergence behavior for multiple parameters and functionals, especially for image analysis [21, 191].

Although the results in this section are inspired by sparse recovery and the results in Section 3.1, the range of applicability of the presented approach is not limited to problems in this field. Image reconstruction, adaptive optics, high-dimensional learning, and several other problems are fields where we can expect that multi-penalty regularization can be fruitfully used. We hope the results in this section to be a useful guideline to those scientists in these fields for a proper use of multi-penalty regularization, whenever their problem requires the separation of sparse and non-sparse

components.

In order to accord to the greatest extent with the notation and setting in the respective paper [140], we continue with a proper introduction of the infinite dimensional generalization of the model (2.21): Let $\mathcal{K}$ and $\mathcal{H}$ be (separable) Hilbert spaces and $T : \mathcal{K} \to \mathcal{H}$ be a bounded linear operator, which we do not specify further, yet. We consider a model problem of the type

$$y = T(u^\dagger + v^\dagger), \tag{3.24}$$

where $u^\dagger$ and $v^\dagger$ respectively correspond to the signal and the noise component of $x^\dagger = u^\dagger + v^\dagger$, which we wish to identify and to separate. Since in general, this unmixing problem has an infinite number of solutions, we furthermore define the operator

$$S : \mathcal{K} \times \mathcal{K} \to \mathcal{H}, \quad S \begin{pmatrix} u \\ v \end{pmatrix} := T(u + v).$$

Its kernel is given by

$$\ker S = \left\{ \begin{pmatrix} u \\ v \end{pmatrix} \in \mathcal{K} \times \mathcal{K} : v = -u + \xi, \quad \xi \in \ker T \right\}.$$

If T had closed range then S would have closed range and the operator

$$S/\sim : (\mathcal{K} \times \mathcal{K})/\ker S \to \mathcal{H}, \quad S\left(\left[\begin{pmatrix} u \\ v \end{pmatrix} \right]_\sim \right) \mapsto T(u + v),$$

would be boundedly invertible on the new restricted quotient space $(\mathcal{K} \times \mathcal{K})/\ker S$ of the equivalence classes given by

$$\begin{pmatrix} u \\ v \end{pmatrix} \sim \begin{pmatrix} u' \\ v' \end{pmatrix} \text{ if and only if } (v - v') + (u - u') \in \ker T.$$

Still, even in this well-posed setting, each of these equivalence classes is huge, and very different representatives can be picked as solutions. In order to distinguish a relevant component $u^\dagger$ of the solution from a noise component $v^\dagger$, we assume that $u^\dagger$ can be actually represented as a sparse vector considered as coordinates with respect to a certain orthogonal basis in $\mathcal{K}$, and $v^\dagger$ is supposed to have bounded coefficients up to a certain noise level $\eta > 0$ with respect to the *same* basis. For the sake of simplicity, we shall identify below vectors in $\mathcal{K}$ with their Fourier coefficients in ℓ_2 with respect to the fixed orthonormal basis.

Eventually, we want to provide the reader with a brief guide towards the remainder of the present section. In the Section 3.2.1, a geometrical intuition of the just described situation is given, and based on this vivid presentation the general form of the multi-penalty minimization, so concisely described in the introduction of this section, is derived. Afterwards, it follows in Section 3.2.2 the definition and convergence analysis of an alternating iterative algorithm for solving the multi-penalty minimization problem. Section 3.2.3 then concerns an empirical investigation of the clustering of solutions depending on the parameters and is considered as a generalization of the results in two-dimensions that are presented in the following section.

3.2.1 Geometrical Intuition from a 2D Example

As a very simple and instructive example of the situation described so far, let us assume $\mathcal{K} = \mathcal{H} = \mathbb{R}^2$ and $T = I$ being the identity operator. Notice, that by the use of the exemplary identity operator, we will already observe a diversity of phenomena, which become very likely even more complicated, as soon as the operator is taken from the even more general set of compressed sensing operators, like, e.g. randomly sampled cosine transformation matrices, which map from a high-dimensional signal space to a measurement space of significantly lower dimension. Under the assumptions on the structure of the interesting solution $y = x^\dagger = u^\dagger + v^\dagger$, without loss of generality we write $u^\dagger = (u_1^\dagger, 0)$ for $R = u_1^\dagger > 0$ and $\max\{|v_1^\dagger|, |v_2^\dagger|\} = \eta = |y_2| > 0$. We consider now the following constrained problem: depending on the choice of $R > 0$, find $u, v \in \mathbb{R}^2$ such that

$$\mathcal{P}(R) \qquad u \in B_{\ell_p}(R), v \in B_{\ell_q}(|y_2|) \text{ subject to } u + v = y,$$

where $q = \infty$ and $0 < p < 1$.

Simple geometrical arguments, as illustrated in Figure 3.4, yield to the existence of a special radius $R^* = R^*(\eta, p) > 0$ for which only three situations can occur:

- If $R < R^*$ then problem $\mathcal{P}(R)$ has no solutions;
- If $R > R^*$ then there are infinitely many solutions of $\mathcal{P}(R)$ and the larger R is, the larger is the set of solutions (in measure theoretical sense), including many possible non-sparse solutions in terms of the u component;
- If $R = R^*$ there is only one solution for the problem $\mathcal{P}(R)$, whose $u^\dagger$ components are actually sparse.

Hence, once the noise level η on the solution is fixed, the parameter $R > 0$ can be actually seen as a regularization parameter of the problem, which is smoothly going from the situation where no solution exists, to the situation where there are many solutions, going through the well-posed situation where there is actually only one

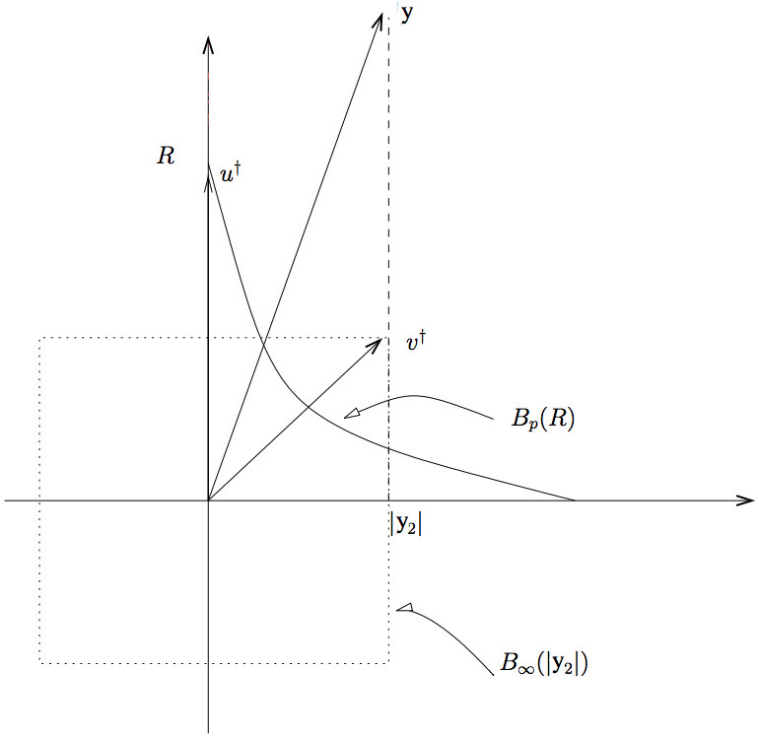


Figure 3.4: Geometrical interpretation of the problem in 2D.

solution. In order to promote uniqueness, one may also reformulate $\mathcal{P}(R)$ in terms of the following optimization problem, depending on $R > 0$ and an additional parameter $\lambda_q > 0$:

$$\mathcal{P}^{\mathrm{opt}}(R, \lambda_q) \qquad \underset{\substack{u \in B_{\ell_p}(R), \\ v \in B_{\ell_q}(|y_2|)}}{\arg\min} \ \|u\|_{\ell_p}^p + \lambda_q \|v\|_{\ell_q}^q \text{ subject to } u + v = y.$$

(Here and later we make an abuse of notation by assuming the convention that $\|\cdot\|_{\ell_q}^q = \|\cdot\|_{\ell_q}$ as soon as $q = \infty$.) The finite dimensional constrained problem $\mathcal{P}(R)$ or its constrained optimization version $\mathcal{P}^{\mathrm{opt}}(R, \lambda_q)$ can be also recast in Lagrangian form as follows:

$$\mathcal{P}(\lambda_p, \lambda_q) \qquad \underset{u,v}{\arg\min} \ \|u + v - y\|_{\ell_2}^2 + \lambda_p \|u\|_{\ell_p}^p + \lambda_q \|v\|_{\ell_q}^q.$$

Due to the equivalence of the problem $\mathcal{P}(R)$ with a problem of the type $\mathcal{P}(\lambda_p, \lambda_q)$ for suitable $\lambda_p = \lambda_p(R) > 0$, $\lambda_q = \lambda_q(R) > 0$, we infer the existence of a parameter

choice $(\lambda_p^*, \lambda_q^*)$ for which $\mathcal{P}(\lambda_p^*, \lambda_q^*)$ has actually a unique solution $(u^\dagger, v^\dagger)$ such that $y = u^\dagger + v^\dagger$. For other choices there might be infinitely many solutions (u, v) for which $\|u + v - y\|_{\ell_2}^2 \geq 0$. While the solution in $\mathbb{R}^2$ of the problem $\mathcal{P}(R)$ follows by simple geometrical arguments, in higher dimension the form $\mathcal{P}(\lambda_p, \lambda_q)$ may allow us to explore solutions via a rather simple algorithm based on alternating minimizations: We shall consider the following iteration, starting from $u^{(0)} = 0 = v^{(0)}$,

$$u^{(n+1)} = \arg\min_u \|u + v^{(n)} - y\|_{\ell_2}^2 + \lambda_p \|u\|_{\ell_p}^p,$$
$$v^{(n+1)} = \arg\min_v \|u^{(n+1)} + v - y\|_{\ell_2}^2 + \lambda_q \|v\|_{\ell_q}^q.$$

As we shall see in details further below, both these two steps are explicitly solved by the help of simple thresholding operations, making this algorithm extremely fast and easy to implement. As we will show in Theorem 3.21 of this article, the algorithm above converges to a solution of $\mathcal{P}(\lambda_p, \lambda_q)$ in the case of $p = 1$ and at least to a local minimal solution in the case of $0 < p < 1$. To get an impression about the operating principle of this alternating algorithm, in the following, we present the results of representative 2D experiments. To this end, we fix $y = (0.3, 1.35)^T$, and consider $0 \leq p < 2$ in order to promote sparsity in $u^\dagger$, $q \geq 2$ in order to obtain a non-sparse $v^\dagger$.

First, consider the case $p = 1$. Due to the strict convexity of $\mathcal{P}(\lambda_p, \lambda_q)$ for $p = 1$ and $q \geq 2$, the computed minimizer is unique. In Figure 3.5 we visually estimate the *regions of solutions* for $u^\dagger$ and $v^\dagger$, which we define as

$$\mathcal{R}_{p,q}^u := \left\{ u^\dagger \,|\, (u^\dagger, v^\dagger) \text{ is the solution of } \mathcal{P}(\lambda_p, \lambda_q), \text{ for } \lambda_p, \lambda_q > 0 \right\},$$

$$\mathcal{R}_{p,q}^v := \left\{ v^\dagger \,|\, (u^\dagger, v^\dagger) \text{ is the solution of } \mathcal{P}(\lambda_p, \lambda_q), \text{ for } \lambda_p, \lambda_q > 0 \right\},$$

by $\times$- and $*$-markers respectively. Notice that this plot does not contain a visualization of the information of which $u^\dagger$ belongs to which $v^\dagger$. In the three plots for $q \in \{2, 4, \infty\}$, the above sets are discretized by showing the solutions for all possible pairs of $\lambda_p, \lambda_q \in \{0.1 \cdot i | i = 1, \ldots, 20\}$.

We immediately observe that the algorithm is computing solutions $u^\dagger, v^\dagger$ in a certain region which is bounded by a parallelogram. In particular, independently of the choice of q, the solutions $u^\dagger$ are distributed only on the upper and left-hand side of this parallelogram, while the solutions $v^\dagger$ may be also distributed in its interior. Depending on the choice of q, the respective regions seem to cover the lower right-hand "triangular" part of the parallelogram, having a straight ($q = 2$), or concave ($q > 2$) boundary. In the case of $q = \infty$, all solutions can be found on the right-hand and lower side of the parallelogram, which represents the limit of the concave case.

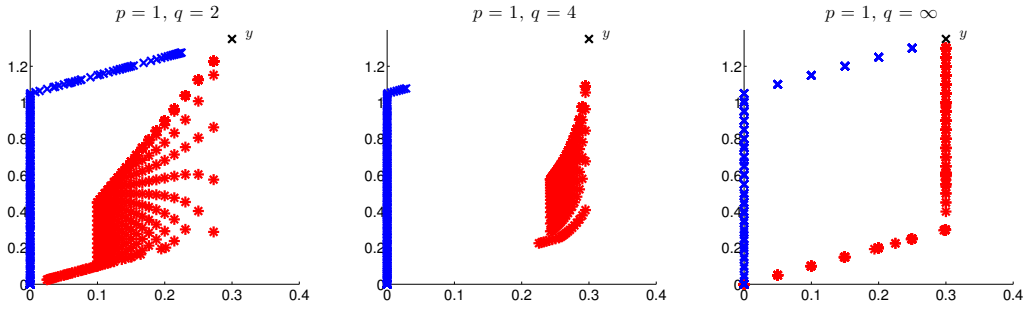


Figure 3.5: Estimated regions of solution for $p = 1$ and $q \in \{2, 4, \infty\}$.

To explain the above results, we have to give a detailed look at a single iteration of the algorithm. As mentioned above, the algorithm is guaranteed to converge to the global minimizer independently on the starting vector. Therefore, for simplicity and transparency we choose $u^{(0)} = v^{(0)} = 0$. The case of $p = 1$ and $q = \infty$ reveals the most "structured" results in terms of the region of solutions, namely piecewise linear paths. Thus, we consider this parameter pair as a reference for the following explanations. In Figure 3.6 we explicitly show the first three iterations of the algorithm, as well as the totality of 13 iterations, setting $\lambda_p = 0.4$ and $\lambda_q = 0.5$. To get a better understanding of what the algorithm is doing, we introduce the notion of *solution path*, which we define as the set of minimizers, depending on λ_p and λ_q respectively,

$$U_p^{n+1} := \left\{ \bar{u} \,|\, \bar{u} = \arg\min_u \ \|u + v^{(n)} - y\|_{\ell_2}^2 + \lambda_p \|u\|_{\ell_p}^p, \lambda_p > 0 \right\},$$

$$V_q^{n+1} := \left\{ \bar{v} \,|\, \bar{v} = \arg\min_v \ \|v + u^{(n+1)} - y\|_{\ell_2}^2 + \lambda_q \|v\|_{\ell_q}^q, \lambda_q > 0 \right\}.$$

As we shall show in Section 3.2.2.1, these sets can be described, explicitly in the case of $p = 1$ and $q = \infty$, by simple thresholding operators $u^{(n+1)} = \mathbb{S}^1_{\lambda_p}(y - v^{(n)})$, and $v^{(n+1)} = \mathbb{S}^\infty_{\lambda_q}(y - u^{(n+1)})$, as defined in (3.25), and (3.26) on page 75.

In Figure 3.6 the solution paths U_1^{n+1} and V_∞^{n+1} are presented as dashed and dotted lines respectively. We observe the particular shape of a piecewise linear one-dimensional path. Naturally, $u^{(n+1)} \in U_1^{n+1}$ and $v^{(n+1)} \in V_\infty^{n+1}$. In our particular setting, we can observe geometrically, and also verify by the above given thresholding functions, that $u^{(n)} \in U_1^1$ for all $n \in \mathbb{N}$. The detailed calculation can be found in the box on page 76. It implies that also the limit has to be in the same set, and, therefore, the set of limit points is included in U_1^1, which is represented by a piecewise linear path between 0 and y.

While we have a well-shaped convex problem in the case of $p = 1$, the situation becomes more complicated for $p < 1$ since multiple minimizers may appear and the

$$\left(\mathbb{S}^1_{\lambda_p}(y-v^{(n)})\right)_i := \max\left\{1-\frac{\lambda_p}{2|y_i-v_i^{(n)}|},0\right\}(y_i-v_i^{(n)}),\quad i=1,2 \tag{3.25}$$

$$\mathbb{S}^\infty_{\lambda_q}(y-u^{(n+1)}) := \begin{cases} \begin{pmatrix}0\\0\end{pmatrix}, & |y_1-u_1^{(n+1)}|+|y_2-u_2^{(n+1)}|<\lambda_q/2,\\ \begin{pmatrix}\operatorname{sign}(y_1-u_1^{(n+1)})(|y_1-u_1^{(n+1)}|-\lambda_q/2)\\ y_2-u_2^{(n+1)}\end{pmatrix}, & |y_2-u_2^{(n+1)}|<|y_1-u_1^{(n+1)}|-\lambda_q/2,\\ \begin{pmatrix}y_1-u_1^{(n+1)}\\ \operatorname{sign}(y_2-u_2^{(n+1)})(|y_2-u_2^{(n+1)}|-\lambda_q/2)\end{pmatrix}, & |y_1-u_1^{(n+1)}|<|y_2-u_2^{(n+1)}|-\lambda_q/2,\\ \frac{|y_1-u_1^{(n+1)}|+|y_2-u_2^{(n+1)}|-\lambda_q/2}{2}\begin{pmatrix}\operatorname{sign}(y_1-u_1^{(n+1)}),\\ \operatorname{sign}(y_2-u_2^{(n+1)})\end{pmatrix}, & \text{else.}\end{cases} \tag{3.26}$$

Table 3.1: Sub-cases related to $\hat{\gamma}$ and $\check{\gamma}$.

	case	equivalent formulation	$y-v^n$ (by (3.27))
A.1	$\check{\gamma} > y_2-\lambda_q/2+y_1$	$\lvert y_1-u_1^{(n)}\rvert+\lvert y_2-u_2^{(n)}\rvert<\lambda_q/2$	$\begin{pmatrix}y_1\\y_2\end{pmatrix}$
A.2	$\check{\gamma} < y_2-\lambda_q/2-y_1$	$\lvert y_1-u_1^{(n)}\rvert<\lvert y_2-u_2^{(n)}\rvert-\lambda_q/2$	$\begin{pmatrix}0\\\check{\gamma}+\lambda_q/2\end{pmatrix}$
A.3	else	else	$\begin{pmatrix}\frac{y_1-y_2+\lambda_q/2+\check{\gamma}}{2}\\(y_2-y_1)+\frac{y_1-y_2+\lambda_q/2+\check{\gamma}}{2}\end{pmatrix}$
B.1	$\hat{\gamma} > y_1-\lambda_q/4$	$\lvert y_1-u_1^{(n)}\rvert+\lvert y_2-u_2^{(n)}\rvert<\lambda_q/2$	$\begin{pmatrix}y_1\\y_2\end{pmatrix}$
B.2	else	else	$\begin{pmatrix}\hat{\gamma}+\lambda_q/4\\(y_2-y_1)+\hat{\gamma}+\lambda_q/4\end{pmatrix}$

Detailed calculation to verify $u^{(n)} \in U_1^1$ for all $n \in \mathbb{N}$.

For representative purposes, we will show $u^{(n)} \in U_1^1$, for all $n \in \mathbb{N}$. Without loss of generality, we assume $y_2 > y_1$ and prove the above statement by induction. By definition $u^{(1)} \in U_1^1$. It remains to show the induction step $u^{(n)} \in U_1^1 \Rightarrow u^{(n+1)} \in U_1^1$. Then, the repeated application of the induction step yields the statement.

If $y - v^{(n)} \in U_1^1$, then there exists an λ_p' such that $y - v^{(n)} = \mathbb{S}^1_{\lambda_p'}(y)$ and by a simple case-by-case analysis, one verifies $u^{(n+1)} = \mathbb{S}^1_{\lambda_p}(\mathbb{S}^1_{\lambda_p'}(y)) = \mathbb{S}^1_{\lambda_p+\lambda_p'}(y) \in U_1^1$. Thus, it remains to show $y - v^{(n)} \in U_1^1$.
We know that by definition

$$y - v^{(n)} = y - \mathbb{S}^\infty_{\lambda_q}(y - u^{(n)}). \tag{3.27}$$

Since, by induction hypothesis, $u^{(n)} \in U_1^1$, there exists a γ such that $u^{(n)} = \mathbb{S}^1_\gamma(y)$. We choose an equivalent but more practical representation for elements in U_1^1 by employing an additional parameterization: There exist two cases:

(A) $u^{(n)} = \begin{pmatrix} 0 \\ \check{\gamma} \end{pmatrix}$, for $\check{\gamma} \in [0, y_2 - y_1]$;

(B) $u^{(n)} = \begin{pmatrix} \hat{\gamma} \\ y_2 - y_1 + \hat{\gamma} \end{pmatrix}$, for $\hat{\gamma} \in [0, y_1]$.

Each of these two cases has to be subdivided into sub-cases related to $\hat{\gamma}$ and $\check{\gamma}$. In Table 3.1, we summarize all sub-cases, an equivalent formulation in terms of the definition of $\mathbb{S}^\infty_{\lambda_q}(y - u^{(n)})$, and the result of $y - v^{(n)}$.

It remains to check for each case if the result of $y - v^{(n)}$ is an element of U_1^1. Obviously in the cases A.1 and B.1, it is true. For the other cases, we check if the result can be expressed in terms of the above given practical representation of elements in U_1^1. In case A.2, by definition we have $0 \leq \check{\gamma} + \lambda_q/2 < y_2 - y_1$. In case A.3, it holds $y_2 - \lambda_q/2 - y_1 \leq \check{\gamma} \leq y_2 - \lambda_q/2 + y_1$ and thus obtain by adding $-y_2 + \lambda_q/2 + y_1$ and division by 2 that $0 \leq \frac{y_1 - y_2 + \lambda_q/2 + \check{\gamma}}{2} \leq y_1$. In case B.2, we immediately get $\hat{\gamma} \leq y_1 - \lambda_q/4$ that $0 \leq \hat{\gamma} + \lambda_q/4 \leq y_1$. Thus, we have shown the statement for all cases.

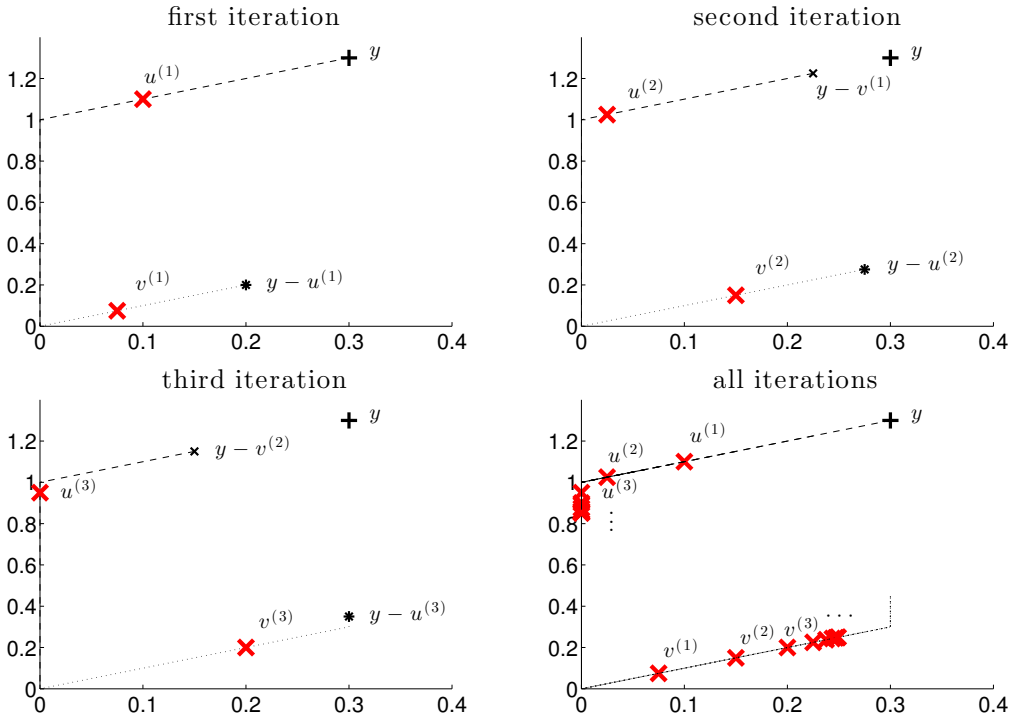


Figure 3.6: Behavior of the algorithm for $p = 1$, $q = \infty$, $\lambda_p = 0.4$, $\lambda_q = 0.5$. The solution path for u and v is represented by the dashed and dotted line respectively.

global minimizer has to be determined. In Figure 3.7 again we visually estimate the regions of solutions $(u^\dagger, v^\dagger)$ with $\times$- and $*$-markers respectively. In the three plots for $p = 0.5$ and $q \in \{2, 4, \infty\}$, the regions of solutions are discretized by showing the solutions for all possible pairs of $\lambda_p, \lambda_q \in \{0.1 \cdot i \,|\, i = 1, \ldots, 40\}$. Compared to the results shown in Figure 3.5, on the one hand, the parallelogram is expanded and on the other hand, two gaps seem to be present in the solution region of $u^\dagger$. Such behavior is due to the non-convexity of the problem. As an extreme case, in Figure 3.8 we present the same experiments only putting $p = 0$. As one can easily see, the obtained results confirm the above observations. Note that in this limit case setting, the gaps become so large, that the solution area of $u^\dagger$ is restricted to three vectors only.

Owing to these first simple results, we obtain the following three preliminary observations:

1. The algorithm promotes a variety of solutions, which form a very particular structure;

2. With decreasing p, and increasing q, the clustering of the solutions is stronger;

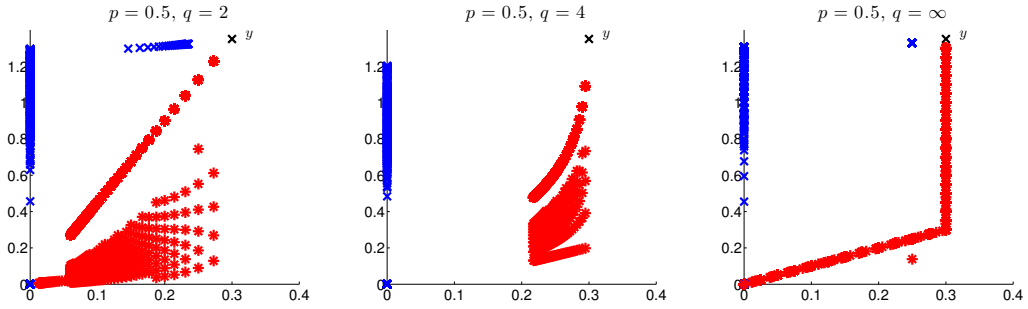


Figure 3.7: Estimated regions of solution for $p = 0.5$, and $q \in \{2, 4, \infty\}$.

3. The set of possible solutions is bounded by a compact set and, thus, many possible solutions can never be obtained for any choice of $q > 2$, $\lambda_p > 0$, and $\lambda_q > 0$.

Inspired by this simple geometrical example of a 2D unmixing problem, we deal in this section with several aspects of optimizations of the type $\mathcal{P}(\lambda_p, \lambda_q)$, recasting the unmixing problem (3.24) into the classical inverse problems framework, where T may have non-closed range and the observed data is additionally corrupted by noise, obtained by folding additive noise on the signal through the measurement operator T, i.e.,

$$y = Tu^\dagger + \xi,$$

where $\xi = Tv^\dagger$ and $\|v^\dagger\|_{\ell_2} \leq \eta,\ \eta \in (0, 1)$. Due to non-closedness of $\mathcal{R}(T)$, the solution $u^\dagger$ does not depend anymore continuously on the data and can be reconstructed in a stable way from y only by means of a regularization method [69].

On the basis of these considerations, we assume that the components u and v of the solution are sequences belonging to suitable spaces ℓ_p and $\ell_2 = \ell_q \cap \ell_2$ respectively, for $0 \leq p < 2$ and $2 \leq q < \infty$. We are interested in the numerical minimization in $\ell_p \times \ell_2$ of the general form of the functionals

$$J_{p,q}(u, v) := \|T(u + v) - y\|_{\mathcal{H}}^2 + \lambda_p \|u\|_{\ell_p}^p + \left(\lambda_q \|v\|_{\ell_q}^q + \varepsilon \|v\|_{\ell_2}^2\right), \tag{3.28}$$

where $\lambda_p, \lambda_q, \varepsilon \in \mathbb{R}_+$, and p, q may all be considered regularization parameters of the problem. The parameter $\varepsilon > 0$ ensures the ℓ_2-coercivity of $J_{p,q}(u, \cdot)$ also with respect to the component v. We shall also take advantage of this additional term in the proof of Lemma 3.27.

In the remainder of this section, we

1. propose in Section 3.2.2 an iterative alternating algorithm to perform the minimization of $J_{p,q}$ by simple iterative thresholding steps; due to the potential non-convexity of the functional for $0 < p < 1$ the analysis of this iteration requires a very careful adaptation of several techniques which are collected in

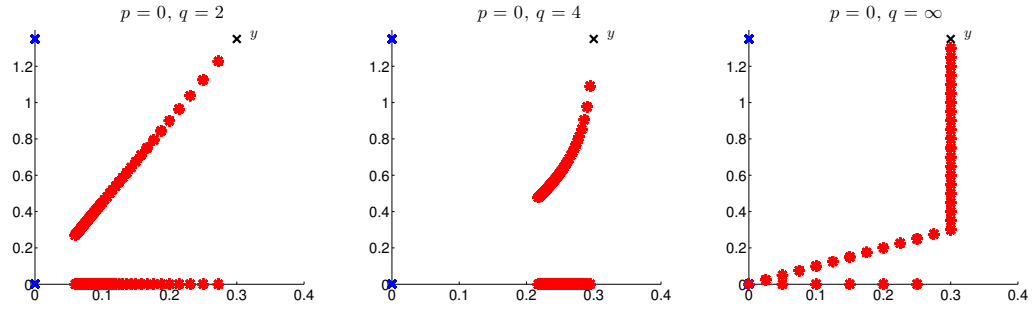


Figure 3.8: Estimated regions of solution for $p = 0$, and $q \in \{2, 4, \infty\}$.

different previous papers of several authors [19, 23, 47, 74] on a single-parameter regularization with a sparsity-promoting ℓ_p-penalty, $0 \leq p < 2$.

2. explore in Section 3.2.3, thanks to this algorithm, and by high-dimensional data analysis methods such as Principle Component Analysis (PCA), the geometry of the computed solutions for different parameters λ_p, λ_q and p, q.

In Section 3.3, we eventually compare the effect of the multi-penalty method in terms of quality of support identification to the effect of the methods which we presented in Section 3.1. Although the theory is presented here for a more general setting, in that section we restrict the range of numerical tests to finite dimensional problems.

3.2.2 An Iterative Algorithm for Multi-Penalty Minimization and its Convergence Properties

We want to minimize $J_{p,q}$ by the suitable instances of the alternating Algorithm 8.

Algorithm 8 Alternating Iterative Thresholding (AIT) - conceptual formulation

1: pick up initial $u^{(0)}, v^{(0)}$
2: **loop**
3: $\quad u^{(n+1)} \approx \arg\min_u J_{p,q}(u, v^{(n)})$
4: $\quad v^{(n+1)} \approx \arg\min_v J_{p,q}(u^{(n+1)}, v)$
5: **end loop**

This formulation is of conceptual nature. That is why we also use the approximation symbol "$\approx$" because in practice we never perform the exact minimization. Instead of optimising $J_{p,q}$ directly, let us introduce auxiliary functionals J_u^s, J_v^s, called the

surrogate functionals of $J_{p,q}$: for some additional parameter a let

$$J_u^s(u,v;a) := J_{p,q}(u,v) + \|u-a\|_2^2 - \|Tu-Ta\|_{\mathcal{H}}^2, \tag{3.29}$$

$$J_v^s(u,v;a) := J_{p,q}(u,v) + \|v-a\|_2^2 - \|Tv-Ta\|_{\mathcal{H}}^2. \tag{3.30}$$

In the following we assume that $\|T\| < 1$. This condition can always be achieved by suitable rescaling of T and y. Observe that

$$\|u-a\|_2^2 - \|Tu-Ta\|_{\mathcal{H}}^2 \geq C\|u-a\|_2^2, \tag{3.31}$$

$$\|v-a\|_2^2 - \|Tv-Ta\|_{\mathcal{H}}^2 \geq C\|v-a\|_2^2, \tag{3.32}$$

for $C = (1-\|T\|)^2$. Hence,

$$J_{p,q}(u,v) = J_u^s(u,v;u) \leq J_u^s(u,v;a), \tag{3.33}$$

$$J_{p,q}(u,v) = J_v^s(u,v;v) \leq J_v^s(u,v;a). \tag{3.34}$$

everywhere, with equality if and only if $u = a$ or $v = a$. Moreover, the functionals decouple the variables u_λ and v_λ so that the above minimization procedure reduces to component-wise minimization (see Section 3.2.2.1 below).

Alternating minimization, as in Algorithm 8, can be performed by minimizing the corresponding surrogate functionals (3.29)–(3.30). This leads to the sequential Algorithm 9.

Algorithm 9 Alternating Iterative Thresholding (AIT)

pick up initial $u^{(0)}, v^{(0)}$
loop
  $u^{(n)} = u^{(n,L)} = u^{(n+1,0)}$
  for $l = 0, \dots, L-1$ **do**
    $u^{(n+1,l+1)} = \operatorname{arg\,min}_{u \in \ell_2(\Lambda)} J_u^s(u, v^{(n)}; u^{(n+1,l)})$
  end for
  $v^{(n)} = v^{(n,M)} = v^{(n+1,0)}$
  for $l = 0, \dots, M-1$ **do**
    $v^{(n+1,l+1)} = \operatorname{arg\,min}_{v \in \ell_2(\Lambda)} J_v^s(u^{(n+1,L)}, v; v^{(n+1,l)})$
  end for
end loop

The main virtue/advantage of Algorithm 9 is the given explicit formulas for computation of the successive $v^{(n)}$ and $u^{(n)}$. The following subsection is dedicated to the efficient computation of the minimizers of $J_u^s(u,v;a)$ and $J_v^s(u,v;a)$, by the help of thresholding functions. Eventually, this allows us to formulate an implementable version of Algorithm 9 in the end of this subsection.

3.2.2.1 New Thresholding Operators for an Iterative Algorithm

We first observe a useful property of the surrogate functionals. Expanding the squared terms on the right-hand side of the expression (3.29), we get

$$\begin{aligned} J_u^s(u,v;a) &= \|u - T^*(y - Ta - Tv) - a\|_2^2 + \lambda_p \|u\|_p^p + \Phi_1 \\ &= \sum_{i\in\Lambda} [(u_i - [(a - T^*Ta - T^*Tv + T^*y)]_i)^2 + \lambda_p |u_i|^p + \Phi_1, \end{aligned}$$

and similarly for the expression (3.30) and $2 \leq q < \infty$

$$\begin{aligned} J_v^s(u,v;a) &= \|v - T^*(y - Ta - Tv) - a\|_2^2 + \lambda_q \|v\|_q^q + \varepsilon \|v\|_2^2 + \Phi_2 \\ &= \sum_{i\in\Lambda} [(v_i - [(a - T^*Ta - T^*Tu + T^*y)]_i)^2 + \lambda_q |v_i|^q + \varepsilon |v_i|^2] + \Phi_2, \end{aligned}$$

where the terms $\Phi_1 = \Phi_1(a, y, v)$ and $\Phi_2 = \Phi_2(a, y, u)$ depend only on a, y, v, and a, y, u respectively. Due to the cancellation of the terms involving $\|Tu\|_2^2$ and $\|Tv\|_2^2$, the variables u_i, v_i in J_u^s and J_v^s respectively are decoupled. Therefore, the minimizers of $J_u^s(u, v; a)$, $J_v^s(u, v; a)$ for a and v or u fixed respectively, can be computed *component-wise* according to

$$u_i^* = \operatorname*{arg\,min}_{t\in\mathbb{R}} [(t - [(a - T^*Ta - T^*Tv + T^*y)]_i)^2 + \lambda_p |t|^p], \ i \in \Lambda, \tag{3.35}$$

$$v_i^* = \operatorname*{arg\,min}_{t\in\mathbb{R}} [(t - [(a - T^*Ta - T^*Tu + T^*y)]_i)^2 + \lambda_q |t|^q + \varepsilon |t|^2]. \tag{3.36}$$

In the case $p = 0,\ p = 0.5,\ p = 1$ and $q = 2$ one can solve (3.35) and (3.36) explicitly; the treatment of the case $q = \infty$ is explained in Remark 3.12; for the general case $0 < p < 2,\ 2 < q < \infty$ we derive an implementable and efficient method to compute $u^{(n)}, v^{(n)}$ from previous iterations.

Minimizers of $J_v^s(u, v; a)$ for a, u fixed We first discuss the minimization of the functional $J_v^s(u, v; a)$ for a generic a, u. For $2 \leq q < \infty$ the summand in $J_v^s(u, v; a)$ is differentiable in v_i, and the minimization reduces to solving the variational equation

$$2(1 + \varepsilon)v_i + \lambda_q q \operatorname{sign}(v_i)|v_i|^{q-1} = 2[a + T^*(y - Tu - Ta)]_i.$$

Setting $\tilde{v}_i := (1 + \varepsilon)v_i$ and recalling that $|\cdot|$ is 1-homogenous, we may rewrite the above equality as

$$\tilde{v}_i + \frac{\lambda_q q}{2} \frac{\operatorname{sign}(\tilde{v}_i)|\tilde{v}_i|^{q-1}}{(1 + \varepsilon)^{q-1}} = [a + T^*(y - Tu - Ta)]_i.$$

Since for any choice of $\lambda_q \geq 0$ and any $q > 1$, the real function

$$F_{\lambda_q,\varepsilon}^q(t) = t + \frac{\lambda_q q}{2(1 + \varepsilon)^{q-1}} \operatorname{sign}(t)|t|^{q-1}$$

is a one-to-one map from $\mathbb{R}$ to itself, we thus find that the minimizer of $J_v^s(u, v; a)$ satisfies

$$v_i^* = v_i = (1+\varepsilon)^{-1} S^q_{\lambda_q,\varepsilon}(a_i + [T^*(y - Tu - Ta)]_i), \tag{3.37}$$

where $S^q_{\lambda_q,\varepsilon}$ is defined by

$$S^q_{\lambda_q,\varepsilon} = (F^q_{\lambda_q,\varepsilon})^{-1} \quad \text{for } q \geq 2.$$

Remark 3.12
In the particular case $q = 2$ the explicit form of the thresholding function $S^2_{\lambda_q,\varepsilon}$ can be easily derived as a proper scaling and we refer the interested reader to [47]. For $q = \infty$ the definition of the thresholding function as

$$\mathbb{S}^\infty_{\lambda_q,\varepsilon}(x) = \arg\min_v \ \|v - x\|_2^2 + \lambda_q \|v\|_{\ell_\infty} + \varepsilon \|v\|_2^2,$$

for vectors $v, x \in \mathbb{R}^M$ was determined explicitly using the polar projection method [78]. Since in our numerical experiments we consider finite-dimensional sequences and the case $q = \infty$, we recall here $\mathbb{S}^\infty_{\lambda_q,\varepsilon}$ explicitly for the case $\varepsilon = 0$ (in finite-dimensions the additional ℓ_2-term $\varepsilon\|v\|_2^2$ is not necessary to have ℓ_2-coercivity).

Let $x \in \mathbb{R}^M$ and $\lambda_q > 0$. Order the entries of x by magnitude such that $|x_{i_1}| \geq |x_{i_2}| \geq \ldots \geq |x_{i_M}|$.

1. If $\|x\|_1 < \lambda_q/2$, then $\mathbb{S}^\infty_{\lambda_q,\varepsilon}(x) = 0$.
2. Suppose $\|x\|_1 > \lambda_q/2$. If $|x_{i_2}| < |x_{i_1}| - \lambda_q/2$, then choose $n = 1$. Otherwise, let $n \in \{2, \ldots, M\}$ be the largest index satisfying

$$|x_{i_n}| \geq \frac{1}{n-1} \left(\sum_{k=1}^{n-1} |x_{i_k}| - \frac{\lambda_q}{2} \right).$$

Then

$$(\mathbb{S}^\infty_{\lambda_q,\varepsilon}(x))_{i_j} = \frac{\operatorname{sign}(x_{i_j})}{n} \left(\sum_{k=1}^{n} |x_{i_k}| - \frac{\lambda_q}{2} \right), \quad j = 1, \ldots, n$$
$$(\mathbb{S}^\infty_{\lambda_q,\varepsilon}(x))_{i_j} = x_{i_j}, \quad j = n+1, \ldots, M.$$

These results cannot be in practice extended to the infinite-dimensional case because one would need to perform the reordering of the infinite-dimensional vector in absolute values. However, in the case of infinite-dimensional sequences, i.e., $x \in \ell_2(\Lambda)$, which is our main interest in the theoretical part of the current manuscript, one can still use the results [78] by employing at the first step an adaptive coarsening approach described in [40]. This approach allows us to obtain an approximation of an infinite-dimensional sequence by its N-dimensional counterpart with optimal accuracy order.

Minimizers of $J_u^s(u, v; a)$ for a, v fixed In this subsection, we want to derive an efficient method to compute $u^{(n)}$. In the special case $1 \leq p < 2$ the iteration $u^{(n)}$ is given by soft-thresholdings [47] (compare also Section 2.4.3.1 for the case $p = 1$); for $p = 0$ the iteration $u^{(n)}$ is defined by hard-thresholding [19] (Section 2.4.3.2). For the sake of brevity, we limit our analysis below to the range $0 < p < 1$, which requires a more careful adaptation of the techniques already included in [19, 47]. The cases $p = 0$ and $1 \leq p < 2$ are actually minor modifications of our analysis and the one of [19, 47]

In order to derive the minimizers of the non-smooth and non-convex functional $J_u^s(u, v; a)$ for generic a, v, we follow the similar approach as proposed in [23], where a general framework for minimization of non-smooth and non-convex functionals based on a generalized gradient projection method has been considered.

Proposition 3.13
For $0 < p < 1$ the minimizer (3.35) for generic a, v can be computed by

$$u_i^* = H_{\lambda_p}^p(a_i + [T^*(y - Tv - Ta)]_i), \quad i \in \Lambda, \tag{3.38}$$

where the function $H_{\lambda_p}^p : \mathbb{R} \to \mathbb{R}$ obeys:

$$H_{\lambda_p}^p(t) = \begin{cases} 0, & |t| \leq \tau_{\lambda_p}, \\ (F_{\lambda_p}^p)^{-1}(t), & |t| \geq \tau_{\lambda_p}, \end{cases}$$

$$|H_{\lambda_p}^p(t)| \in \{0\} \cup \{t \geq \gamma_{\lambda_p}\}.$$

Here, $(F_{\lambda_p}^p)^{-1}(t)$ is the inverse of the function $F_{\lambda_p}^p(s) = s + \frac{\lambda_p p}{2}\operatorname{sign}(s)|s|^{p-1}$, which is defined on $\mathbb{R}_+$, strictly convex and attains a minimum at $s_{\lambda_p} > 0$, and

$$\gamma_{\lambda_p} = (\lambda_p(1-p))^{1/(2-p)}, \quad \tau_{\lambda_p} = F_{\lambda_p}^p(\gamma_{\lambda_p}) = \frac{2-p}{2-2p}(\lambda_p(1-p))^{1/(2-p)}.$$

The thresholding function $H_{\lambda_p}^p$ is continuous except at $|t| = \tau_{\lambda_p}$, where it has a jump discontinuity.

The proof of the proposition follows similar arguments as presented in [23, Lemma 3.10, 3.12] and, thus, for the sake of brevity, it can be omitted here. In Figure 3.9, the thresholding function $H_{\lambda_p}^p$ is plotted for selected values of p and $\lambda_p = 0.1$.

Remark 3.14
Since we consider the case $p = 0.5$ in our numerical experiments, we present here an explicit formulation of the thresholding function $H_{\lambda_p}^{1/2}$, which has been derived recently in [193, 196]. It is given by

$$H_{\lambda_p}^{1/2}(t) = \begin{cases} 0, & |t| \leq \frac{\sqrt[3]{54}}{4}(\lambda_p)^{2/3}, \\ (F_{\lambda_p}^{1/2})^{-1}(t), & |t| \geq \frac{\sqrt[3]{54}}{4}(\lambda_p)^{2/3}, \end{cases}$$

where

$$\left(F_{\lambda_p}^{1/2}\right)^{-1}(t) = \frac{2}{3}t\left(1+\cos\left(\frac{2\pi}{3}-\frac{2}{3}\arccos\left(\frac{\lambda_p}{8}\left(\frac{|t|}{3}\right)^{-3/2}\right)\right)\right).$$

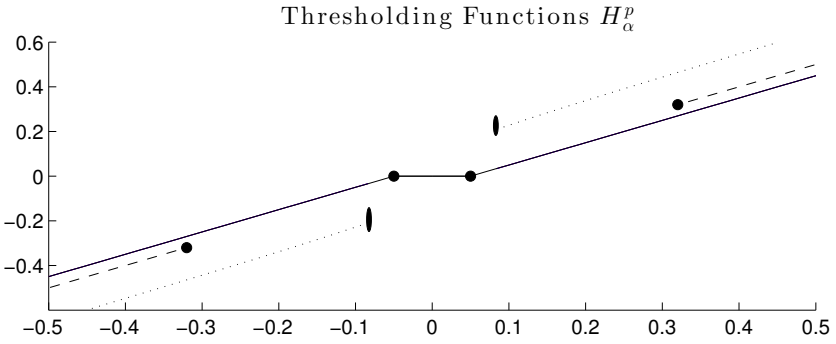


Figure 3.9: The thresholding function $H_{\lambda_p}^p$ for $p = 1$, $p = 0.3$ (dotted), $p = 0$ (dashed) and the parameter $\lambda_p = 0.1$.

For solving the low-level minimization problems in Algorithm 9, we can use an iterative thresholding algorithm induced by (3.37) and (3.38), and thus, reformulate it as Algorithm 10.

Algorithm 10 Alternating Iterative Thresholding (AIT) — implementable formulation

pick up initial $u^{(0)}, v^{(0)}$
loop
  $u^{(n)} = u^{(n,L)} = u^{(n+1,0)}$
  for $l = 0, \ldots, L-1$ **do**
    $u_i^{(n+1,l+1)} = H_{\lambda_p}^p(u_i^{(n+1,l)} + [T^*(y - Tv^{(n,M)} - Tu^{(n+1,l)})]_i)$
  end for
  $v^{(n)} = v^{(n,M)} = v^{(n+1,0)}$
  for $l = 0, \ldots, M-1$ **do**
    $v_i^{(n+1,l+1)} = (1+\varepsilon)^{-1} S_{\lambda_q,\varepsilon}^q(v_i^{(n+1,l)} + [T^*(y - Tu^{(n+1)} - Tv^{(n+1,l)})]_i)$
  end for
end loop

3.2.2.2 Auxiliary Results: On Fixed Points and Fixed Index Sets

In preparation of the proof of weak and strong convergence of Algorithm 10, which will be the main result of our theoretical investigations, it is necessary to present some auxiliary lemmas.

Convergence of the difference of successive iterates The following lemma provides a tool to prove the weak convergence of the algorithm. It is standard when using surrogate functionals (see [19, 47]), and concerns general real-valued surrogate functionals. It holds independently of the specific form of the functional $J_{p,q}$, but does rely on the restriction that $\|T\| < 1$. Since it is more convenient for the proof of the lemma, we use the formulation of Algorithm 9, instead of the one of Algorithm 10. This strategy is of no further consequence since both Algorithms are equivalent.

Lemma 3.15
If $J_u^s(u, v; a)$ and $J_v^s(u, v; a)$ are given as in (3.29) *and* (3.30)*, and the sequences $(u^{(n)})$ and $(v^{(n)})$ are generated by Algorithm 9, then the sequences $J_{p,q}(u^{(n)}, v^{(n)})$, $J_u^s(u^{(n+1)}, v^{(n)}; u^{(n)})$ and $J_v^s(u^{(n+1)}, v^{(n+1)}; v^{(n)})$ are non-increasing as long as $\|T\| < 1$. Moreover,*

$$\|u^{(n+1)} - u^{(n)}\|_2 \to 0, \qquad \|v^{(n+1)} - v^{(n)}\|_2 \to 0,$$

for $n \to \infty$.

Proof. Using (3.33) we have

$$J_{p,q}(u^{(n)}, v^{(n)}) = J_u^s(u^{(n)}, v^{(n)}; u^{(n)}) = J_u^s(u^{(n,L)}, v^{(n,M)}; u^{(n+1,0)}).$$

Since at this point the proof is similar for both u and v, for the sake of brevity, we consider the case of J_u^s in detail only. By definition of $u^{(n+1,1)}$, and its property of being a minimizer in step 5 of Algorithm 9, we have

$$J_u^s(u^{(n,L)}, v^{(n,M)}; u^{(n+1,0)}) \geq J_u^s(u^{(n+1,1)}, v^{(n,M)}; u^{(n+1,0)}).$$

An application of (3.33) gives

$$J_u^s(u^{(n+1,1)}, v^{(n,M)}; u^{(n+1,0)}) \geq J_u^s(u^{(n+1,1)}, v^{(n,M)}; u^{(n+1,1)}).$$

Putting in line these inequalities we get

$$J_{p,q}(u^{(n)}, v^{(n)}) \geq J_u^s(u^{(n+1,1)}, v^{(n,M)}; u^{(n+1,1)}).$$

In particular, from (3.31) we obtain

$$J_{p,q}(u^{(n)}, v^{(n)}) - J_u^s(u^{(n+1,1)}, v^{(n,M)}; u^{(n+1,1)}) \geq C\|u^{(n+1,1)} - u^{(n+1,0)}\|_2^2.$$

By successive iterations of this argument we get

$$\begin{aligned} J_{p,q}(u^{(n)}, v^{(n)}) &\geq J_u^s(u^{(n+1,1)}, v^{(n,M)}; u^{(n+1,1)}) \geq J_u^s(u^{(n+1,L)}, v^{(n,M)}; u^{(n+1,L)}) \\ &= J_{p,q}(u^{(n+1,L)}, v^{(n,M)}), \end{aligned} \tag{3.39}$$

and

$$J_{p,q}(u^{(n,L)}, v^{(n,M)}) - J_{p,q}(u^{(n+1,L)}, v^{(n,M)}) \geq C \sum_{l=0}^{L-1} \|u^{(n+1,l+1)} - u^{(n+1,l)}\|_2^2. \tag{3.40}$$

By definition of $v^{(n+1,1)}$ and its minimal properties

$$J_v^s(u^{(n+1,L)}, v^{(n,M)}; v^{(n+1,0)}) \geq J_v^s(v^{(n+1,L)}, v^{(n,M)}; v^{(n+1,1)}).$$

By similar arguments as above we find

$$\begin{aligned} J_{p,q}(u^{(n+1,L)}, v^{(n,M)}) &\geq J_v^s(u^{(n+1,L)}, v^{(n+1,M)}; v^{(n+1,M)}) \\ &= J_{p,q}(u^{(n+1,L)}, v^{(n+1,M)}), \end{aligned} \tag{3.41}$$

and

$$J_{p,q}(u^{(n+1,L)}, v^{(n,M)}) - J_{p,q}(u^{(n+1,L)}, v^{(n+1,M)}) \geq C \sum_{l=0}^{M-1} \|v^{(n+1,l+1)} - v^{(n+1,l)}\|_2^2. \tag{3.42}$$

From the above discussion it follows that $J_{p,q}(u^{(n)}, v^{(n)}) \geq 0$ is a non-increasing sequence, therefore it converges. From (3.40) and (3.42) and the latter convergence we deduce

$$\sum_{l=0}^{L-1} \|u^{(n+1,l+1)} - u^{(n+1,l)}\|_2^2 \to 0, \quad \sum_{l=0}^{M-1} \|v^{(n+1,l+1)} - v^{(n+1,l)}\|_2^2 \to 0.$$

In particular, by the triangle inequality and the standard inequality $(a+b)^2 \leq 2(a^2+b^2)$ for $a, b > 0$, we also have

$$\begin{aligned} \|u^{(n+1,L)} - u^{(n+1,0)}\|_2^2 &= \left\| \sum_{l=0}^{L-1} (u^{(n+1,l+1)} - u^{(n+1,l)}) \right\|_2^2 \\ \leq \left(\sum_{l=0}^{L-1} \|u^{(n+1,l+1)} - u^{(n+1,l)}\|_2 \right)^2 &\leq C_L \sum_{l=0}^{L-1} \|u^{(n+1,l+1)} - u^{(n+1,l)}\|_2^2 \to 0. \end{aligned}$$

Here C_L is some constant depending on L. Analogously we can show that

$$\|v^{(n+1,M)} - v^{(n+1,0)}\|_2^2 \leq C_M \sum_{l=0}^{M-1} \|v^{(n+1,m+1)} - v^{(n+1,m)}\|_2^2 \to 0.$$

Therefore, we finally obtain

$$\|u^{(n+1,L)} - u^{(n+1,0)}\|_2^2 = \|u^{(n+1)} - u^{(n)}\|_2^2 \to 0,$$
$$\|v^{(n+1,M)} - v^{(n+1,0)}\|_2^2 = \|v^{(n+1)} - v^{(n)}\|_2^2 \to 0.$$

□

Specifying the fixed points As Algorithm 10 may have multiple fixed points, it is important to analyze those in more detail. At first, we specify what we understand as a fixed point.

Definition 3.16 (The set of fixed points *Fix*)
Let us define the functions

$$F_u(\bar{u}, \bar{v}) = \arg\min_u J_u^s(u, \bar{v}; \bar{u}), \tag{3.43}$$
$$F_v(\bar{u}, \bar{v}) = \arg\min_v J_v^s(\bar{u}, v; \bar{v}). \tag{3.44}$$

Then we say that (u^*, v^*) is a fixed point for the equations (3.43) and (3.44) if

$$\begin{cases} u^* = F_u(u^*, v^*), \\ v^* = F_v(u^*, v^*). \end{cases}$$

We define *Fix* to be the set of fixed points for the equations (3.43) and (3.44).

Lemma 3.17
Let $(u^*, v^*) \in Fix$. Define the sets $\Gamma_0 := \{i \in \Lambda | \, u_i^* = 0\}$ and $\Gamma_1 := \{i \in \Lambda | \, |u_i^*| \geq \gamma_{\lambda_p}\}$. Then

$$[T^*(y - Tv^* - Tu^*)]_i = \frac{\lambda_p p}{2} \operatorname{sign}(u_i^*)|u_i^*|^{p-1}, \quad \text{if } i \in \Gamma_1,$$

or

$$|[T^*(y - Tv^* - Tu^*)]_i| \leq \tau_{\lambda_p}, \quad \text{if } i \in \Gamma_0.$$

Proof. By Proposition 3.13

$$u_i^* = H_{\lambda_p}^p(u_i^* + [T^*(y - Tv^* - Tu^*)]_i), \ \forall i \in \Lambda.$$

If $u_i^* = 0$, this equality holds if and only if $|[T^*(y - Tv^* - Tu^*)]_i| \leq \tau_{\lambda_p}$. Similarly for $i \in \Gamma_1$ we get

$$u_i^* = (F_{\lambda_p}^p)^{-1}(u_i^* + [T^*(y - Tv^* - Tu^*)]_i),$$

and by definition of $F_{\lambda_p}^p$ we have $(F_{\lambda_p}^p)^{-1}(u_i^* + \frac{\lambda_p p}{2} \operatorname{sign}(u_i^*)|u_i^*|^{p-1}) = u_i^*$. Thus, the statement of the lemma follows. □

Fixation of the index set Γ_1 To ease notation, we define the operators $\mathbb{H}^p_{\lambda_p}\colon \ell_2(\Lambda) \to \ell_2(\Lambda)$ and $\mathbb{S}_\tau\colon \ell_2(\Lambda) \to \ell_2(\Lambda)$ by their component-wise action

$$(\mathbb{H}^p_{\lambda_p}(u))_i := H^p_{\lambda_p}(u_i),$$
$$(\mathbb{S}_\tau(v))_i := S^q_{\lambda_q,\varepsilon}(v_i),$$

here $\tau = \frac{\lambda_q q}{2(1+\varepsilon)^{q-1}}$.

At the core of the proof of convergence stands the fixation of the "discontinuity set" during the iteration (3.38), at which point the non-convex and non-smooth minimization with respect to u in Algorithm 9 is transformed into a simpler problem.

Lemma 3.18
Consider the iterations

$$u^{(n+1,l+1)} = \mathbb{H}^p_{\lambda_p}(u^{(n+1,l)} + T^*(y - Tv^{(n,M)} - Tu^{(n+1,l)}))$$

and the partition of the index set Λ into

$$\Gamma_1^{n,l} = \{i \in \Lambda |\, |u_i^{(n,l)}| \geq \gamma_{\lambda_p}\},$$
$$\Gamma_0^{n,l} = \{i \in \Lambda |\, u_i^{(n,l)} = 0\},$$

where $(\tau_{\lambda_p}, \gamma_{\lambda_p})$ is the position of the jump-discontinuity of the thresholding function. For sufficiently large $n_0 \in \mathbb{N}$ (after a finite number of iterations), this partition fixes during the iterations, meaning there exists Γ_0 such that for all $n \geq n_0$, for all $l \leq L$, $\Gamma_0^{n,l} = \Gamma_0$ and $\Gamma_1^{n,l} = \Gamma_1 := \Lambda \setminus \Gamma_0$.

Proof. By discontinuity of the thresholding function $H^p_{\lambda_p}$, each sequence component satisfies

- $u_i^{(n,l)} = 0$ if $i \in \Gamma_0^{n,l}$;
- $|u_i^{(n,l)}| \geq \gamma_{\lambda_p}$ if $i \in \Gamma_1^{n,l}$.

Thus, $|u_i^{(n,l+1)} - u_i^{(n,l)}| \geq \gamma_{\lambda_p}$ if $i \in \Gamma_0^{n,l+1} \cap \Gamma_1^{n,l}$, or $i \in \Gamma_0^{n,l} \cap \Gamma_1^{n,l+1}$. At the same time, Lemma 3.15 implies that

$$|u_i^{(n,l+1)} - u_i^{(n,l)}| \leq \|u^{(n,l+1)} - u^{(n,l)}\|_2 \leq \epsilon,$$

for sufficiently large $n \geq n_0(\epsilon)$. In particular, the last inequality implies that Γ_0 and Γ_1 must be fixed once $n \geq n_0(\epsilon)$. □

Since $(u^{(n,l)}) \in \ell_2$, the set Γ_1 is finite. Moreover, fixation of the index set Γ_1 implies that the sequence $(u^{(n)})$ can be considered constrained to a subset of $\ell_2(\Lambda)$ on which the functionals $J_{p,q}(\cdot, v)$ and $J^s_u(\cdot, v; a)$ are differentiable.

3.2.2.3 Convergence of the Iterative Algorithm

Through the results of the previous subsection, we are now able to formulate the two main theoretical results of this section, which concern the weak and strong convergence of the algorithm. While the weak convergence is sufficient to have also strong convergence in a finite dimensional setting, for the sake of a closed theoretical investigation, we also report the proof of strong convergence.

Weak convergence Given that $H^p_{\lambda_p}(u_i^{(n)}) = (F^p_{\lambda_p})^{-1}(u_i^{(n)})$, $i \in \Gamma_1$, after a finite number of iterations, we can use the tools from real analysis to prove that the sequence $(u^{(n)})$ converges to some stationary point. Notice that the convergence of the iterations of the type

$$u^{(n+1)} = F_u(u^{(n)}, \bar{v}), \quad v^{(n+1)} = F_v(\bar{u}, v^{(n)}),$$

to a fixed point $(\bar{u}^*, \bar{v}^*)$ for any $(\bar{u}, \bar{v})$ and F_u, F_v given as in (3.43)–(3.44) has been extensively discussed in the literature, e.g., [19, 23, 47, 78].

Theorem 3.19
Assume $0 < p < 1$ and $2 \leq q < \infty$. Algorithm 9 produces sequences $(u^{(n)})$ and $(v^{(n)})$ in $\ell_2(\Lambda)$ whose weak accumulation points are fixed points of the equations (3.43)–(3.44).

Proof. By Lemma 3.18 the iteration step

$$u_i^{(n+1,l+1)} = H^p_{\lambda_p}(u_i^{(n+1,l)} + [T^*(y - Tv^{(n,M)} - Tu^{(n+1,l)})]_i)$$

becomes equivalent to the step of the form

$$u_i^{(n+1,l+1)} = (F^p_{\lambda_p})^{-1}(u_i^{(n+1,l)} + [T^*(y - Tv^{(n,M)} - Tu^{(n+1,l)})]_i), \ i \in \Gamma_1,$$

after a finite number of iterations and $u_{i'}^{(n+1,l+1)} = 0$, for all $i' \in \Lambda \setminus \Gamma_1 = \Gamma_0$.

From (3.39) and (3.41) we have

$$J_{p,q}(u^{(0)}, v^{(0)}) \geq J_{p,q}(u^{(n)}, v^{(n)}) \geq \lambda_p \|u^{(n)}\|_p^p \geq \lambda_p \|u^{(n)}\|_2^p,$$

and

$$J_{p,q}(u^{(0)}, v^{(0)}) \geq J_{p,q}(u^{(n+1)}, v^{(n)}) \geq \lambda_q \|v^{(n)}\|_q^q + \varepsilon \|v^{(n)}\|_2^2 \geq \varepsilon \|v^{(n)}\|_2^2.$$

This means that $(u^{(n)})$ and $(v^{(n)})$ are uniformly bounded in $\ell_2(\Lambda)$, hence there exist weakly convergent subsequences $(u^{(n_j)})$ and $(v^{(n_j)})$. Let us denote by u^∞ and v^∞ the weak limits of the corresponding subsequences. For simplicity, we rename the corresponding subsequences $(u^{(n)})$ and $(v^{(n)})$. Moreover, since the sequence $J_{p,q}(u^{(n)}, v^{(n)})$ is monotonically decreasing and bounded from below by 0, it is also convergent.

First of all, let us recall that the weak convergence implies component wise convergence, so that $u_i^{(n)} \to u_i^\infty$, $v_i^{(n)} \to v_i^\infty$, and $[T^*Tu^{(n)}]_i \to [T^*Tu^{(\infty)}]_i$, $[T^*Tv^{(n)}]_i \to [T^*Tv^{(\infty)}]_i$.

By definition of $u^{(n+1,L)}$ and $v^{(n+1,M)}$ in Algorithm 10, we have for n large enough

$$\begin{aligned} 0 &= [-2(u^{(n+1,L-1)} + T^*(y - Tv^{(n)}) - T^*Tu^{(n+1,L-1)})]_i + 2u_i^{(n+1,L)} \\ &\quad + \lambda_p p \operatorname{sign}(u_i^{(n+1,L)})|u_i^{(n+1,L)}|^{p-1},\ i \in \Gamma_1, \qquad (3.45) \\ 0 &= [-2(v^{(n+1,M-1)} + T^*(y - Tu^{(n+1,L)}) - T^*Tv^{(n+1,M-1)})]_i + (2+\varepsilon)v_i^{(n+1,M)} \\ &\quad + \lambda_q q \operatorname{sign}(v_i^{(n+1,M)})|v_i^{(n+1,M)}|^{q-1},\ i \in \Lambda. \qquad (3.46) \end{aligned}$$

By taking now the limit for $n \to \infty$ in (3.45) and (3.46), and by using Lemma 3.15 we obtain

$$\begin{aligned} 0 &= [-2(u^\infty + T^*(y - Tv^\infty) - T^*Tu^\infty)]_i + 2u_i^\infty + \lambda_p p \operatorname{sign}(u_i^\infty)|u_i^\infty|^{p-1},\ i \in \Gamma_1, \\ 0 &= [-2(v^\infty + T^*(y - Tu^\infty) - T^*Tv^\infty)]_i + (2+\varepsilon)v_i^\infty + \lambda_q q \operatorname{sign}(v_i^\infty)|v_i^\infty|^{q-1},\ i \in \Lambda. \end{aligned}$$

An application of Lemma 3.17 implies $(u^*, v^*) = (u^\infty, v^\infty)$, i.e.,

$$\begin{aligned} u_i^\infty &= H_{\lambda_p}^p(u_i^\infty + [T^*(y - Tv^\infty - Tu^\infty)]_i),\ i \in \Gamma_1. \\ v_i^\infty &= (1+\varepsilon)^{-1} S_{\lambda_q,\varepsilon}^q(v_i^\infty + [T^*(y - Tv^\infty - Tu^\infty)]_i),\ i \in \Lambda. \end{aligned}$$

The argumentation holds true for every subsequence of $(u^{(n)})$ and $(v^{(n)})$. □

Remark 3.20
The case $q = \infty$ would need a special treatment due to lack of differentiability. For simplicity we further assume that $2 \le q < \infty$.

Minimizers of $J_{p,q}$ In this section we explore the relationship between a limit point (u^*, v^*) of Algorithm 9 and minimizers of the functional (3.28). We shall show that under the FBI property (compare Definition 2.10) the set of fixed points of the algorithm is a subset of the set of local minimizers. We note that here again we provide the proof only for the case $0 < p < 1$, and we refer to [19, 47] for the cases $p = 0$ and $1 \le p < 2$, which follow similarly after minor adaptations.

Theorem 3.21
Let T have the FBI property. Let us denote $\mathcal{L}$ the set of local minimizers of $J_{p,q}$. Then we have the following inclusion

$$Fix \subset \mathcal{L},$$

where Fix is the set of fixed points for equations (3.43)–(3.44).

In order to present a proof of Theorem 3.21, we need to provide two additional propositions. In the first, show that the choice of a sufficiently small $p \in (0,1)$ guarantees that an accumulation point (u^*, v^*) is a local minimizer of the functional with respect to u, where we use the FBI property as a main ingredient. The second proposition makes the respective statement for the component v, but without any additional condition.

Proposition 3.22
Let T satisfy the FBI property. Then there exists $p^* \in (0,1)$ such that for every $0 < p < p^*$ every accumulation point (u^*, v^*) is a local minimizer of the functional $J_{p,q}$ with respect to u, i.e.,

$$J_{p,q}(u^* + du, v^*) \geq J_{p,q}(u^*, v^*)$$

for any $du \in \ell_2(\Lambda)$, $\|du\|_2 \leq \epsilon_1$ for ϵ_1 sufficiently small.

Proof. In the following, we denote by $J_{p,q}^{\Gamma_1}$ the restriction of the functional $J_{p,q}$ to $\ell_2(\Gamma_1)$, i.e.,

$$J_{p,q}^{\Gamma_1}(u,v) := \|T(u+v) - y\|_{\mathcal{H}}^2 + \lambda_p \sum_{i \in \Gamma_1} |u_i|^p + \left(\lambda_q \|v\|_q^q + \varepsilon \|v\|_2^2\right),$$

and by J_{u,Γ_1}^s, J_{v,Γ_1}^s the corresponding surrogate functionals restricted to $\ell_2(\Gamma_1)$.

For the sake of simplicity, let us define

$$F(u) = J_{p,q}(u,v), \qquad F^{\Gamma_1}(u) = J_{p,q}^{\Gamma_1}(u,v).$$

We proceed with the proof of the lemma in two steps:

- We show that an accumulation point (u^*, v^*) is a local minimizer of $F^{\Gamma_1}(u)$;
- We show that (u^*, v^*) is a local minimizer of $F(u)$.

Let us for now consider the $u_i^{(n)}$ for $i \in \Gamma_1$, i.e., $|u_i^{(n)}| \geq \gamma_{\lambda_p}$. Since u^* is an accumulation point for $(u^{(n)})$, by Theorem 3.19 it is also a fixed point. Taking into account the restriction to the set Γ_1, by Lemma 3.17 we get

$$[T^*(Tv^* + Tu^* - y)]_i + \frac{\lambda_p p}{2} \operatorname{sign}(u_i^*)|u_i^*|^{p-1} = 0.$$

As the functional $F^{\Gamma_1}(u)$ is differentiable on $\ell_2(\Gamma_1)$, we compute the Jacobian

$$\nabla F^{\Gamma_1}(u) = 2T^*(Tv + Tu - y) + \lambda_p p u |u|^{p-2},$$

for which holds $\nabla F^{\Gamma_1}(u^*) = 0$, $v = v^*$. Since the mapping is smooth for all $u_i \neq 0$, one can check additionally that the Hessian matrix

$$\nabla^2 F^{\Gamma_1}(u^*) = 2T^*T - \lambda_p p(1-p)\operatorname{diag}(|u^*|^{p-2}),$$

is actually positive definite for $p < p^*$: For z with $\operatorname{supp} z \subset \operatorname{supp} u^*$ we have the following estimate

$$\begin{aligned}\langle z, \nabla^2 F^{\Gamma_1}(u^*)z\rangle &= 2\|Tz\|_{\mathcal{H}}^2 - \lambda_p p(1-p)\sum_{i\in\Gamma_1}|u_i^*|^{p-2}z_i^2 \\ &\geq (c - \lambda_p p(1-p)\gamma_{\lambda_p}^{p-2})\|z\|_2^2 = (c-p)\|z\|_2^2,\end{aligned}$$

where $c > 0$ is the the smallest eigenvalue of T^*T. Therefore, for all $p \in (0, p^*)$, $p^* = \min\{1, c\}$, the Hessian is positive definite and thus u^* is a local minimizer of F^{Γ_1}.

Next we show that u^* is a local minimizer of the functional $F(u)$ without the restriction on the support of u^*. For the sake of transparency, we shall write the restrictions u_{Γ_1}, $u^*_{\Gamma_1}$ and du_{Γ_1} meaning that $u_{\Gamma_1}, u^*_{\Gamma_1}, du_{\Gamma_1} \in \ell_2(\Gamma_1)$, and du_{Γ_0} meaning that $du_{\Gamma_0} \in \ell_2(\Gamma_0)$.

The desired statement of the proposition follows if we can show that $F^{\Gamma_1}(u^*_{\Gamma_1} + du_{\Gamma_1}) \leq F(u^* + du)$. At this point it is convenient to write the functional $F(u^* + du)$ with $\bar{y} := y - Tv^*$ as

$$\begin{aligned}F(u^* + du) = \|T_{\Gamma_1}(u^*_{\Gamma_1} + du_{\Gamma_1}) + T_{\Gamma_0}du_{\Gamma_0} - \bar{y}\|_{\mathcal{H}}^2 \\ + \lambda_p\|u_{\Gamma_1}\|_p^p + \lambda_p\|du_{\Gamma_1}\|_p^p + \lambda_p\|du_{\Gamma_0}\|_p^p + \lambda_q\|v\|_q^q + \varepsilon\|v\|_2^2.\end{aligned}$$

Moreover, the inequality $F^{\Gamma_1}(u^*_{\Gamma_1} + du_{\Gamma_1}) \leq F(u^* + du)$ can be written as

$$-\lambda_p\|du_{\Gamma_0}\|_p^p \leq \|T_{\Gamma_1}(u^*_{\Gamma_1} + du_{\Gamma_1}) + T_{\Gamma_0}du_{\Gamma_0} - \bar{y}\|_{\mathcal{H}}^2 - \|T_{\Gamma_1}(u^*_{\Gamma_1} + du_{\Gamma_1}) - \bar{y}\|_{\mathcal{H}}^2.$$

By developing the squares, we obtain

$$\begin{aligned}-\lambda_p\|du_{\Gamma_0}\|_p^p &\leq 2\langle T_{\Gamma_1}(u^*_{\Gamma_1} + du_{\Gamma_1}) - \bar{y}, T_{\Gamma_0}du_{\Gamma_0}\rangle + \|T_{\Gamma_0}du_{\Gamma_0}\|_{\mathcal{H}}^2 \\ &= 2(\langle T_{\Gamma_1}(u^*_{\Gamma_1} + du_{\Gamma_1}), T_{\Gamma_0}du_{\Gamma_0}\rangle - \langle T_{\Gamma_0}du_{\Gamma_0}, \bar{y}\rangle) + \|T_{\Gamma_0}du_{\Gamma_0}\|_{\mathcal{H}}^2,\end{aligned}$$

for $\|du_{\Gamma_0}\|_2$ sufficiently small. One concludes by observing that for $p < 1$ the term $\|du_{\Gamma_0}\|_p^p$ will always dominate the linear terms on the right-hand side of the above inequality. □

Proposition 3.23
Every accumulation point (u^*, v^*) is a local minimizer of the functional $J_{p,q}$ with respect to v, i.e.,

$$J_{p,q}(u^*, v^* + dv) \geq J_{p,q}(u^*, v^*)$$

for any $dv \in \ell_2(\Lambda)$, $\|dv\|_2 \leq \epsilon_2$ for $\epsilon_2 > 0$ sufficiently small.

Proof. First of all, we claim that $J_v^s(u^*, v^* + dv; v^*) - J_v^s(u^*, v^*; v^*) \geq \|dv\|_2^2$. Indeed, a direct calculation shows that

$$\begin{aligned}
&J_v^s(u^*, v^* + dv; v^*) - J_v^s(u^*, v^*; v^*) \\
=&\|T(u^* + v^* + dv) - y\|_{\mathcal{H}}^2 + \lambda_q \|v^* + dv\|_q^q + \varepsilon \|v^* + dv\|_2^2 \\
&+ \|dv\|_2^2 - \|Tdv\|_{\mathcal{H}}^2 - \|T(u^* + v^*) - y\|_{\mathcal{H}}^2 - \lambda_q \|v^*\|_q^q - \varepsilon \|v^*\|_2^2 \\
=&\|dv\|_2^2 + \lambda_q \sum_{i \in \Lambda} (|v_i^* + dv_i|^q - |v_i^*|^q) + \varepsilon \sum_{i \in \Lambda} (|v_i^* + dv_i|^2 - |v_i^*|^2) \\
&+ \sum_{i \in \Lambda} dv_i [T^*(T(u^* + v^*) - y)]_i \\
\geq&(1 + \varepsilon)\|dv\|_2^2 + \sum_{i \in \Lambda} dv_i ([T^*(T(u^* + v^*) - y)]_i + \lambda_q q \operatorname{sign}(v_i^*)|v_i^*|^{q-1} + 2\varepsilon v_i^*).
\end{aligned}$$

Since by (3.36) the term

$$\langle [T^*(T(u^* + v^*) - y)]_i + \lambda_q q \operatorname{sign}(v_i^*)|v_i^*|^{q-1} + 2\varepsilon v_i^*, t_i - v_i^* \rangle$$

vanishes, the above claim follows. By using the above claim, we get that

$$\begin{aligned}
J_{p,q}(u^*, v^* + dv) &= J_v^s(u^*, v^* + dv; v^*) - \|dv\|_2^2 + \|Tdv\|_{\mathcal{H}}^2 \\
&\geq J_v^s(u^*, v^* + dv; v^*) - \|dv^*\|_2^2 \\
&\geq J_v^s(u^*, v^*; v^*) = J_{p,q}(u^*, v^*). \qquad \square
\end{aligned}$$

With the obtained results we are now able to prove Theorem 3.21. In particular, we shall show that $J_{p,q}(u^*, v^*) \leq J_{p,q}(u^*, v^* + dv) \leq J_{p,q}(u^* + du, v^* + dv)$. The first inequality has been proven in Proposition 3.23. We only need to show the second inequality.

Proof (Proof of Theorem 3.21). Similarly as in Proposition 3.22 we proceed in two steps. First we prove that $J_{p,q}^{\Gamma_1}(u^*, v^* + dv) \leq J_{p,q}^{\Gamma_1}(u^* + du, v^* + dv)$. Since the functional $J_{p,q}^{\Gamma_1}$ is differentiable, a Taylor expansion at $(u^*, v^* + dv)$ yields

$$J_{p,q}^{\Gamma_1}(u^* + du, v^* + dv) = J_{p,q}^{\Gamma_1}(u^*, v^* + dv) + \nabla J_{p,q}^{\Gamma_1}(u^*, v^* + dv)du + \frac{1}{2} du \nabla^2 J_{p,q}^{\Gamma_1}(u^*, v^* + dv)du.$$

Due to Proposition 3.22, $\nabla F^{\Gamma_1}(u^*) = \nabla J_{p,q}^{\Gamma_1}(u^*, v^*) = 0$ and the term $\nabla J_{p,q}^{\Gamma_1}(u^*, v^* + dv) = 2T^*Tdv \approx 0$. Thus,

$$-2\|T\|^2 \|dv\|_2 \|du\|_2 \leq \nabla J_{p,q}^{\Gamma_1}(u^*, v^* + dv)du = 2\langle Tdv, Tdu \rangle \leq 2\|T\|^2 \|dv\|_2 \|du\|_2.$$

Moreover,

$$\nabla^2 J_{p,q}^{\Gamma_1}(u^*, v^* + dv) = \nabla^2 J_{p,q}^{\Gamma_1}(u^*, v^*) + \xi(\|dv\|_2^2),$$

where $\nabla^2 J^{\Gamma_1}_{p,q}(u^*, v^*) \geq 0$ due to the local convexity of the functional $J^{\Gamma_1}_{p,q}$. Choosing $\eta \leq \frac{c-p}{2\|T\|^2}$ and $\|dv\|_2 = \eta\|du\|_2$, and combining the above estimates together, we get

$$\begin{aligned}&\nabla J^{\Gamma_1}_{p,q}(u^*, v^* + dv)du + \frac{1}{2}du\nabla^2 J^{\Gamma_1}_{p,q}(u^*, v^* + dv)du \\ \geq &-2\|T\|^2\|dv\|_2\|du\|_2 + (c-p)\|du\|_2^2 \geq [(c-p) - 2\eta\|T\|^2]\|du\|_2^2 \geq 0,\end{aligned}$$

and thus, $J^{\Gamma_1}_{p,q}(u^* + du, v^* + dv) \geq J^{\Gamma_1}_{p,q}(u^*, v^* + dv)$. The second part of the proof is concerned with the inequality $J_{p,q}(u^* + du, v^* + dv) \geq J_{p,q}(u^*, v^* + dv)$, which works in a completely analogous way to the second part of the proof of Proposition 3.22 and is therefore omitted here. □

Strong convergence In this subsection we show how the previously established weak convergence can be strengthened into norm convergence, also by a series of lemmas. Since the distinction between weak and strong convergence makes sense only when the index set Λ is infinite, we shall prove the strong convergence only for the sequence $v^{(n)}$ since the iterates $u^{(n)}$ are constrained to the finite set after a finite number of iterations.

For the sake of convenience, we introduce the following notation.

$$\begin{aligned}\mu^{n+1} &= v^{(n+1)} - v^*, \qquad & \mu^{n+1/2} &= v^{(n+1,M-1)} - v^*, \\ \eta^{n+1} &= u^{(n+1)} - u^*, \qquad & h &= v^* + T^*(y - Tu^* - Tv^*),\end{aligned}$$

where $v^* = \text{w-lim}_{n\to\infty} v^{(n)}$ and $u^* = \lim_{n\to\infty} u^{(n)}$. Here and below, we use w-lim as a shorthand for weak limit. For the proof of strong convergence we need the following technical lemmas, which are based on the investigations in [47, 78].

Lemma 3.24
The operator $\mathbb{S}_\tau(v)$ is non-expansive, i.e., $\|\mathbb{S}_\tau(u) - \mathbb{S}_\tau(v)\|_2 \leq \|u - v\|_2$.

Lemma 3.25
Assume $\|\mu^{n+1/2}\|_2 > \gamma$ for all n and for a fixed $\gamma > 0$. Then $\|T\mu^{n+1/2}\|^2_{\mathcal{H}} \to 0$ as $n \to \infty$.

Proof. Since

$$\mu^{n+1} - \mu^{n+1/2} = (1+\varepsilon)^{-1}[\mathbb{S}_\tau(h + (I - T^*T)\mu^{n+1/2} - T^*T\eta^{n+1}) - \mathbb{S}_\tau(h)] - \mu^{n+1/2},$$

and

$$\left\|\mu^{n+1} - \mu^{n+1/2}\right\|_2 = \left\|v^{(n+1,M)} - v^{(n+1,M-1)}\right\|_2 \to 0,$$

by Lemma 3.15, we get that

$$\begin{aligned}&\left\|(1+\varepsilon)^{-1}[\mathbb{S}_\tau(h + (I - T^*T)\mu^{n+1/2} - T^*T\eta^{n+1}) - \mathbb{S}_\tau(h)] - \mu^{n+1/2}\right\| \qquad (3.47)\\ \geq &\left|(1+\varepsilon)^{-1}\left\|\mathbb{S}_\tau(h + (I - T^*T)\mu^{n+1/2} - T^*T\eta^{n+1}) - \mathbb{S}_\tau(h)\right\| - \left\|\mu^{n+1/2}\right\|\right| \to 0.\end{aligned}$$

By non-expansiveness of $\mathbb{S}_\tau$, we have the estimate

$$\left\|\mathbb{S}_\tau(h+(I-T^*T)\mu^{n+1/2}-T^*T\eta^{n+1})-\mathbb{S}_\tau(h)\right\|_2 \leq \left\|(I-T^*T)\mu^{n+1/2}-T^*T\eta^{n+1}\right\|_2.$$

Consider now

$$\begin{aligned}
&\|(I-T^*T)\mu^{n+1/2}-T^*T\eta^{n+1}\|_2^2\\
\leq&\|(I-T^*T)\mu^{n+1/2}\|_2^2+\|T^*T\eta^{n+1}\|_2^2-2\langle(I-T^*T)\mu^{n+1/2},T^*T\eta^{n+1}\rangle\\
\leq&\|(I-T^*T)\mu^{n+1/2}\|_2^2+\|T^*T\eta^{n+1}\|_2+2\|(I-T^*T)\mu^{n+1/2}\|_2\|T^*T\eta^{n+1}\|_2\\
\leq&\|(I-T^*T)\mu^{n+1/2}\|_2^2+\delta+2C\delta\\
\leq&\|\mu^{n+1/2}\|_2^2+\epsilon,
\end{aligned} \tag{3.48}$$

for large enough n so that $\|u^{(n+1,L)}-u^*\|_2\leq\delta$. The constant $C>0$ is due to the boundedness of $\|\mu^{n+1/2}\|$. Due to estimate (3.48), we have

$$\begin{aligned}
&\|\mathbb{S}_\tau(h+(I-T^*T)\mu^{n+1/2}-T^*T\eta^{n+1})-\mathbb{S}_\tau(h)\|_2\\
\leq&\|(I-T^*T)\mu^{n+1/2}-T^*T\eta^{n+1}\|_2\leq\sqrt{\|\mu^{(n+1/2)}\|_2+\epsilon}\leq\|\mu^{(n+1/2)}\|_2+\bar{\epsilon}.
\end{aligned}$$

By assumption of the lemma there exists a subsequence $(\mu^{(n_k+1/2)})$ such that $\|\mu^{(n_k+1/2)}\|_2\geq\gamma$ for all k. For simplicity, we rename such subsequence as $(\mu^{(n+1/2)})$ again. Then

$$(1+\varepsilon)^{-1}\|\mathbb{S}_\tau(h+(I-T^*T)\mu^{n+1/2}-T^*T\eta^{n+1})-\mathbb{S}_\tau(h)\|_2\leq\frac{1}{1+\varepsilon}\|\mu^{(n+1/2)}\|_2+\frac{1}{1+\varepsilon}\bar{\epsilon}.$$

For $\bar{\epsilon}\leq\varepsilon\gamma$ we obtain

$$\begin{aligned}
&(1+\varepsilon)^{-1}\|\mathbb{S}_\tau(h+(I-T^*T)\mu^{n+1/2}-T^*T\eta^{n+1})-\mathbb{S}_\tau(h)\|_2\\
\leq&\frac{1}{1+\varepsilon}\|\mu^{(n+1/2)}\|_2+\frac{1}{1+\varepsilon}(1+\varepsilon-1)\gamma\\
\leq&\frac{1}{1+\varepsilon}\|\mu^{(n+1/2)}\|_2+\left(1-\frac{1}{1+\varepsilon}\right)\|\mu^{(n+1/2)}\|_2\\
\leq&\|\mu^{(n+1/2)}\|_2.
\end{aligned}$$

Combining the above inequalities, we get

$$\begin{aligned}
&\|\mu^{(n+1/2)}\|_2^2-\|(I-T^*T)\mu^{n+1/2}-T^*T\eta^{n+1}\|_2^2\\
\leq&\|\mu^{(n+1/2)}\|_2^2-(1+\varepsilon)^{-1}\|\mathbb{S}_\tau(h+(I-T^*T)\mu^{n+1/2}-T^*T\eta^{n+1})-\mathbb{S}_\tau(h)\|_2^2.
\end{aligned}$$

This implies from (3.47) that

$$\lim_{n\to\infty}[\|\mu^{(n+1/2)}\|_2^2-\|(I-T^*T)\mu^{n+1/2}-T^*T\eta^{n+1}\|_2^2]=0.$$

Using (3.48) we get

$$\begin{aligned}&\|\mu^{(n+1/2)}\|_2^2 - \|(I-T^*T)\mu^{n+1/2} - T^*T\eta^{n+1}\|_2^2\\ \geq&\|\mu^{(n+1/2)}\|_2^2 - \|(I-T^*T)\mu^{n+1/2}\|_2^2 - \epsilon\\ =&2\|T\mu^{n+1/2}\|_{\mathcal{H}}^2 - \|T^*T\mu^{n+1/2}\|_2^2 - \epsilon \geq \|T\mu^{n+1/2}\|_{\mathcal{H}}^2 - \epsilon.\end{aligned}$$

This yields $\|T\mu^{n+1/2}\|_{\mathcal{H}}^2 \to 0$ for $n \to \infty$. □

Lemma 3.26
For $h = v^* + T^*(y - Tu^* - Tv^*)$,

$$\|(1+\varepsilon)^{-1}[\mathbb{S}_\tau(h+\mu^{n+1/2}) - \mathbb{S}_\tau(h)] - \mu^{n+1/2}\|_2 \to 0.$$

Proof.

$$\begin{aligned}&\|(1+\varepsilon)^{-1}[\mathbb{S}_\tau(h+\mu^{n+1/2}) - \mathbb{S}_\tau(h)] - \mu^{n+1/2}\|_2\\ \leq&\|(1+\varepsilon)^{-1}\mathbb{S}_\tau(h+\mu^{n+1/2} - T^*T\mu^{n+1/2} - T^*T\eta^{n+1}) - (1+\varepsilon)^{-1}\mathbb{S}_\tau(h) - \mu^{n+1/2}\|_2\\ +&\|(1+\varepsilon)^{-1}[\mathbb{S}_\tau(h+\mu^{n+1/2}) - \mathbb{S}_\tau(h+\mu^{n+1/2} - T^*T\mu^{n+1/2} - T^*T\eta^{n+1})]\|_2\\ \leq&\|(1+\varepsilon)^{-1}[\mathbb{S}_\tau(h+(I-T^*T)\mu^{n+1/2} - T^*T\eta^{n+1} - \mathbb{S}_\tau(h)] - \mu^{n+1/2}\|_2\\ +&(1+\varepsilon)^{-1}\|T^*T(\mu^{n+1/2}+\eta^{n+1})\|_2,\end{aligned}$$

where we used the non-expansivity of the operator. The result follows since both terms in the last bound tend to 0 for $n \to \infty$ because of the previous lemma and Theorem 3.19. □

Lemma 3.27
If for some $a \in \ell_2(\Lambda)$ *and some sequence* $(v^{(n)})$, $\text{w-lim}_{n\to\infty} v^{(n)} = 0$ *and* $\lim_{n\to\infty}\|(1+\varepsilon)^{-1}[\mathbb{S}_\tau(a+v^{(n)}) - \mathbb{S}_\tau(a)] - v^{(n)}\|_2 = 0$, *then* $\|v^{(n)}\|_2 = 0$ *for* $n \to \infty$.

Proof. In the proof of the lemma we mainly follow the arguments in [47]. Since the sequence $(v^{(n)})$ is weakly convergent, it has to be bounded: there is a constant K such that for all n, $\|v^{(n)}\|_2 \leq K$. Reasoning component-wise we can write $|v_i^{(n)}| < K$ for all $i \in \Lambda$.

Let us define the set $\Gamma_0 = \{i \in \Lambda \,|\, |a_i| \geq K\}$ and since $a \in \ell_2(\Lambda)$, this is a finite set. We then have $\forall i \in \Gamma_1 = \Gamma \setminus \Gamma_0$, that $|a_i|$ and $|a_i + v_i^{(n)}|$ are bounded above by $2K$. Recalling the definition of $S^q_{\lambda_q,\varepsilon} = (F^q_{\lambda_q,\varepsilon})^{-1}$, we observe that for $q \geq 1$ and $|t| \leq 2K$,

$$(F^q_{\lambda_q,\varepsilon})'(t) = 1 + \frac{\lambda_q q(q-1)}{2(1+\varepsilon)^{q-1}}|t|^{q-2} \geq 1,$$

and therefore

$$\begin{aligned}&|(1+\varepsilon)^{-1}[S^q_{\lambda_q,\varepsilon}(a_i+v_i^{(n)})-S^q_{\lambda_q,\varepsilon}(a_i)]|\leq(1+\varepsilon)^{-1}(\max_t|(S^q_{\lambda_q,\varepsilon})'(t)|)|v_i^{(n)}|\\ \leq&(1+\varepsilon)^{-1}(\min_t|(F^q_{\lambda_q,\varepsilon})'(t)|)^{-1}|v_i^{(n)}|\leq(1+\varepsilon)^{-1}|v_i^{(n)}|.\end{aligned}$$

In the first inequality, we have used the mean value theorem and in the second inequality we have used the lower bound for $(F^q_{\lambda_q,\varepsilon})'$ to upper bound the derivative $(S^q_{\lambda_q,\varepsilon})'$ since $S^q_{\lambda_q,\varepsilon}=(F^q_{\lambda_q,\varepsilon})^{-1}$. By subtracting $|v_i^{(n)}|$ from the upper inequality and rewriting $(1-(1+\varepsilon)^{-1})=C'\leq 1$, we have for all $i\in\Gamma_1$, that

$$\begin{aligned}C'|v_i^{(n)}|&\leq|v_i^{(n)}|-(1+\varepsilon)^{-1}|S^q_{\lambda_q,\varepsilon}(a_i+v_i^{(n)})-S^q_{\lambda_q,\varepsilon}(a_i)|\\&\leq|v_i^{(n)}-(1+\varepsilon)^{-1}[S^q_{\lambda_q,\varepsilon}(a_i+v_i^{(n)})-S^q_{\lambda_q,\varepsilon}(a_i)]|,\end{aligned}$$

by the triangle inequality which implies

$$\sum_{i\in\Gamma_1}|v_i^n|^2\leq\left(\frac{1}{C'}\right)^2\sum_{i\in\Gamma_1}|v_i^n-(1+\varepsilon)^{-1}[S^q_{\lambda_q,\varepsilon}(a_i+v_i^{(n)})-S^q_{\lambda_q,\varepsilon}(a_i)]|\to 0,\ n\to\infty.$$

On the other hand, since Γ_0 is a finite set, and $(v^{(n)})$ tends to 0 weakly as n tends to ∞, we also obtain

$$\sum_{i\in\Lambda}|v_i^{(n)}|^2\to 0 \text{ as } n\to\infty.$$

□

Theorem 3.28
Algorithm 9 produces sequences $(u^{(n)})$ and $(v^{(n)})$ in $\ell_2(\Lambda)$ that converge strongly to the vectors u^*, v^* respectively. In particular, the sets of strong accumulation points are non-empty.

Proof. Let u^* and v^* be weak accumulation points and let $(u^{(n_j)})$ and $(v^{(n_j)})$ be subsequences weakly convergent to u^* and v^* respectively. Let us denote the latter subsequences $(u^{(n)})$ and $(v^{(n)})$ again.

If $\mu^{n+1/2}$ is such that $\|\mu^{n+1/2}\|_2\to 0$, then the statement of the theorem follows from Lemma 3.15. If, instead, there exists a subsequence, denoted by the same index, that $\|\mu^{n+1/2}\|_2\geq\gamma$, then by Lemma 3.26 we get that $\|T\mu^{n+1/2}\|_{\mathcal{H}}\to 0$. Subsequently applying Lemma 3.26 and Lemma 3.27, we get $\|\mu^{n+1/2}\|_2=0$, which yields a contradiction to the assumption. Thus, by Lemma 3.15 we have that $(v^{(n)})$ converges to v^* strongly. The strong convergence of $(u^{(n)})$ is already guaranteed by Theorem 3.19. □

3.2.3 Empirical Investigation on the Clustering of Solutions

In this section, we continue the discussion, started in Section 3.2.1 for a 2D example, on the geometry of the solution sets for fixed p, q and regularization parameters chosen from the prescribed grids. We extend our preliminary geometrical observations on the sets of computed solutions to the high-dimensional case by means of *Principal Component Analysis*.

We do not present numerical results regarding the quality of the recovered results here, in particular with respect to support identification properties, but return to such an investigation within a much broader context in Section 3.3. In that section, we compare the results for multi-penalty regularization with $0 \leq p \leq 1$, $2 \leq q \leq \infty$ and the corresponding one-penalty regularization scheme. Those results have been motivated by encouraging outcomes obtained in [76, 139] for the Hilbert space setting, where the authors have shown the superiority and robustness of the multi-penalty regularization scheme compared to the "classical" one-parameter regularization methods. In addition to this we compare the multi-penalty regularization with respect to the methods (ℓ_1+)SLP and (ℓ_1+)IHT, which have been introduced in Section 3.1.3.

3.2.3.1 Problem Formulation and Experiment Data Set

We consider the model problem of the type (3.24), where $T \in \mathbb{R}^{m \times N}$ is an i.i.d. Gaussian matrix, $u^\dagger$ is a sparse vector and $v^\dagger$ is a noise vector. The choice of T corresponds to compressed sensing measurements [84]. In the experiments we consider 20 problems of this type with $u^\dagger$ randomly generated with values on $[-3, 3]$ and $\#\operatorname{supp}(u^\dagger) = 7$, and $v^\dagger$ is a random vector whose components are uniformly distributed on $[-1, 1]$, and normalised such that $\|v^\dagger\|_2 = 0.7$, corresponding to a signal to noise ratio of ca. 10 %. In our numerical experiments, we are keeping such a noise level fixed.

In order to create an experimental data set, we were considering for each of the problems the minimization of the functional (3.28) for $p \in \{0, 0.3, 0.5, 0.8, 1\}$ and $q \in \{2, 4, 10, \infty\}$. The regularization parameters λ_p and λ_q were chosen from the grid $Q^\kappa_{\lambda_{p0}} \times Q^\kappa_{\lambda_{q0}}$, where $Q^\kappa_{\lambda_{p0}} := \{\lambda_p = \lambda_{pi} = \lambda_{p0}\kappa^i \,|\, \lambda_{p0} = 0.0009, \kappa = 1.25, i = 0, \ldots, 30\}$, and $Q^\kappa_{\lambda_{q0}} := \{\lambda_q = \lambda_{qi} = \lambda_{q0}\kappa^i \,|\, \lambda_{q0} = 0.0005, \kappa = 1.25, i = 0, \ldots, 30\}$. For all possible combinations of p and q and (λ_p, λ_q) we run Algorithm 10 with number $L = M = 20$ of inner loop iterations and starting values $u^{(0)} = v^{(0)} = 0$. Furthermore, we set $\varepsilon = 0$ since the additional term $\varepsilon\|v\|_2^2$ is necessary for coercivity only in the infinite-dimensional setting. Due to the fact that the thresholding functions for $p \in \{0.3, 0.8\}$ are not given explicitly, we, at first, precomputed them on a grid of points in $[0, 5]$ and interpolated in between, taking also into consideration the jump discontinuity. Respectively, we did the same precomputations for $q \in \{4, 10\}$ on a grid of points in $[0, 1]$.

3.2.3.2 Clustering of Solutions

As we have seen in Section 3.2.1, the 2D experiments revealed certain regions of computed solutions for $u^\dagger$ and $v^\dagger$ with very particular shapes, depending on the parameters p and q. We question if similar clustering of the solutions can also be found for problems in high dimension. To this end, the challenge is the proper geometrical representation of the computed high dimensional solutions, which can preserve the geometrical structure in terms of mutual distances. We consider the set of the computed solutions for fixed p and q in the grid $Q^k_{\lambda_{p0}} \times Q^k_{\lambda_{q0}}$ as point clouds which we investigate independently with respect to the components $u^\dagger$ and $v^\dagger$ respectively. As the solutions are depending on the two scalar parameters (λ_p, λ_q), it is legitimate to assume that they form a 2-dimensional manifold embedded in the higher-dimensional space. Therefore, we expect to be able to visualize the point clouds and analyze their clustering by employing suitable dimensionality reduction techniques. A broad and nearly complete overview although not extended in its details on existing dimensionality reduction techniques as well as a MATLAB toolbox is provided in [125, 183, 182].

For our purposes, we have chosen the Principal Component Analysis (PCA) technique because we want to verify that calculated minimizers u^* and v^* form clusters around the original solutions. In the rest of the subsection, we only consider one fixed problem from the previously generated data set. In the following figures, we report the estimated regions of the solutions u^* and v^*, as well as the corresponding regularization parameters chosen from the grids $Q^k_{\lambda_{p0}} \times Q^k_{\lambda_{q0}}$. We only present *feasible* solutions, i.e., the ones that satisfy the discrepancy condition

$$\#\operatorname{supp}(u^*) \leq \#\operatorname{supp}(u^\dagger), \text{ and } \|T(u^* + v^*) - y\|_2 < 0.1. \tag{3.49}$$

In Figure 3.10 we consider the cases $p = 0.5$ and $q \in \{2, 4\}$, and in Figure 3.11 the corresponding results for $p = 0.3$ and $q \in \{2, 4\}$ are displayed.

First of all, we observe that the set of solutions u^* forms certain structures, visible here as one-dimensional manifolds, as we also observed in the 2D experiments of the introduction. Likewise, the set of solutions v^* are more unstructured, but still clustered. The effect of modifying q from 2 to 4 increases the number of feasible solutions according to (3.49). Concerning the parameter p, by modifying it from 0.5 to 0.3, the range of λ_p's which provide feasible solutions is growing. Since it is still hard from this geometrical analysis on a single problem to extract any qualitative information concerning the accuracy of the reconstruction, we defer the discussion on multiple problems to Section 3.3.

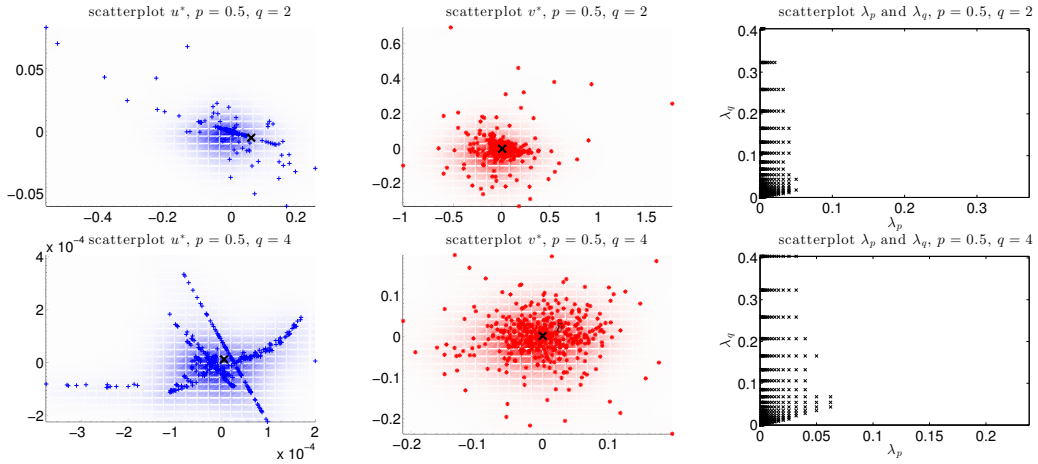


Figure 3.10: Estimated regions of the regularization parameters (right panel) and the corresponding solution u^* (left panel) and v^* (middle panel) for $p = 0.5$, and $q = 2$ (top), and $q = 4$ (bottom) repectively using PCA. The black crosses indicate the real solutions.

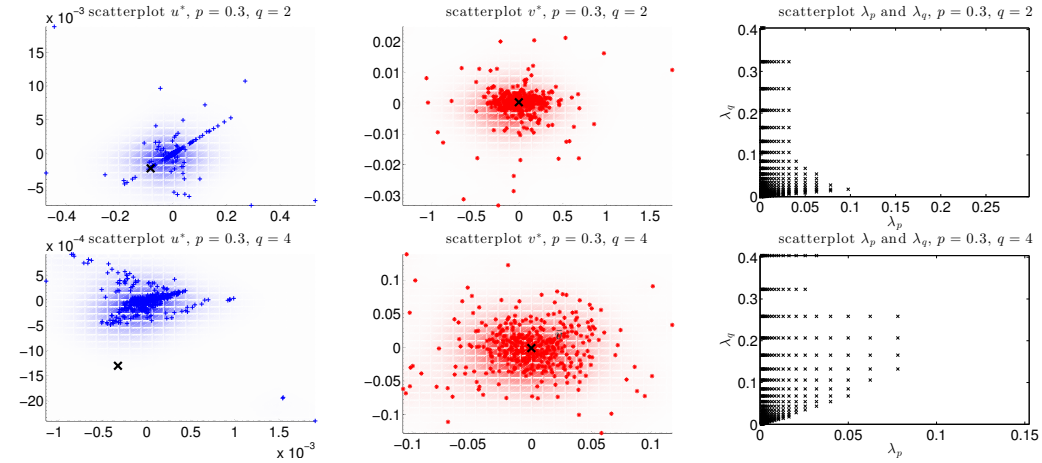


Figure 3.11: Estimated regions of the regularization parameters (right panel) and the corresponding solution u^* (left panel) and v^* (middle panel) for $p = 0.3$, and $q = 2$ (top), and $q = 4$ (bottom) repectively using PCA. The black crosses indicate the real solutions.

3.3 Comparative Numerics

In this section, we empirically support our theoretical results of Section 3.1 and 3.2. Already in the three respective publications [155, 7, 140], we separately showed results, which confirm the superiority of the therein presented methods with respect to more classical ones in terms of robust support identification. However, the methods were compared individually and by the following different assumptions and parameter settings:

- For the methods SLP and IHT of Section 3.1, we confirmed in [155] that they outperform classical ℓ_1-minimization and its reweighted variant in terms of support identification for fixed algorithm parameters. On the one hand, such a test simulates an application case very well since it is in general difficult to adapt the parameters according to a each test problem. Thus, one is interested in a parameter choice which leads to good results of a method for the majority of the test problems. On the other hand, one does not explore the full potential of the methods, which is however of theoretical value since one is naturally also interested in the performance limit of a certain method.

- For the multi-penalty functionals of Section 3.2, we showed in [140] that their minimizers do feature a better approximation of the support than the minimizers of the respective mono-penalty minimization ($\lambda_q = 0$), in particular, for the choice of $0 < p < 1$ and $q = 2$ and the right choice of the regularization parameters λ_p, λ_q. In the respective numerical experiments, we first explored for each pair (p, q) the best possible parameter pair (λ_p, λ_q), and only afterwards we compared those best results to each other (for different p and q). This is a rather non-practical investigation since a simulation is missing, where λ_p and λ_q are fixed for a larger set of test problems. However such a test explores the potential performance limits of those methods.

In this section, we want to close the missing gaps and link both works to the end that all methods (that were compared in both papers) are compared within the same scope. In this spirit, we first explore the performance limits of all methods, allowing a flexible choice of the algorithm parameters (in a certain range depending on the respective method), and we record the parameters which performed best for the entire set of test problems. In order to simulate a more realistic situation, we second compare all methods to each other, but fix the parameters according to their optimal choice from the previous experiment.

In all of the investigations, we place special emphasis on the question, which methods provide a significantly enhanced rate of recovery of the support of the unknown sparse vector as well as a better accuracy in approximating its large entries. In particular, we

compare to the classical convex methods ℓ_1-minimization and its iterative re-weighted version (IRL1) as well as methods which only regularize on a single component (mono-penalty minimization). It is important, to stress once more, that in all experiments, we only consider signals of the class (3.1). We furthermore assume that potential users are in the situation that they do not dispose of the knowledge of the sparsity k, but of the knowledge of the threshold r, which is the lower bound on the absolute value of the relevant entries of the original signal.

3.3.1 Test Setting

We consider a model of the type (2.21), i.e., $y = \Phi(\bar{x} + n)$, where $\Phi \in \mathbb{R}^{m \times N}$ is an i.i.d. Gaussian matrix, $\bar{x}$ is a k-sparse vector and n is a noise vector. The choice of Φ corresponds to compressed sensing measurements [84]. In the experiments, we consider 20 test problems of this type and dimensions $N = 100$ and $m = 40$, with $\bar{x}$ randomly generated with values on $[-3, -1] \cup [1, 3]$ (consequently $r = 1$) and $k = \#\operatorname{supp}(\bar{x}) = 7$. The components of the random noise vector n are uniformly distributed on $[-1, 1]$ and normalized such that $\eta = \|n\|_2 = 0.7$, corresponding to a signal to noise ratio of ca. 10%. In the terms of the class (3.1) in Section 3.1, we therefore have $\bar{x}+n \in \mathcal{S}^2_{0.7,7,1}$.

The following methods are supposed to be compared to each other: We consider

- the minimization of the multi-penalty functional (3.28) for each combination of $p \in \{0, 0.3, 0.5, 0.8, 1\}$ and $q \in \{2, 4, \infty\}$ by Algorithm 10. Depending on the parameter pair, we call the respective method AIT(p, q). The algorithm naturally returns a sparse and a noise component (u^* and v^*, compare Theorem 3.28). The sparse component is the vector of interest and denoted by $x^*_{p,q}$;
- the minimization of the functional (3.28) with only the first penalty for $p \in \{0, 0.3, 0.5, 0.8, 1\}$, by Algorithm 10 in only one component (set the other component to 0). We call the method the respective *mono-penalty* minimization and denote it with AIT(p,0). We extract from the minimizer the sparse component $x^*_{p,0}$, which only contains the elements exceeding r in absolute value. Notice that for $p = 1$ the problem (3.28) becomes the regularized ℓ_1-minimization (2.12), and the method AIT(1,0) is the same as the iterative soft thresholding algorithm (ISTA) (compare Section 2.4.3.1). Since the regularized ℓ_1-minimization is equivalent to the ℓ_1-minimization with inequality constraint (BPDN) (see Section 2.2.1), we do not need to compute the results of AIT(1,0), but use BPDN instead. Note that this also covers ℓ_1-minimization with equality constraint (BP) as a special case;
- the prewhitened ℓ_1-minimization with inequality constraint (PWBPDN (2.22)). We extract from its minimizer the sparse component $x^*_{\ell_1}$, which only contains the elements exceeding r in absolute value;

- iterative re-weighted ℓ_1-minimization (IRL1) in its general form (2.46) with denoising parameter δ. We extract from its minimizer the sparse component x^*_{IRL1}, which only contains the elements exceeding r in absolute value;
- ℓ_1+SLP, which performs Algorithm 7 with the ℓ_1-minimizer as starting value. We extract from the SLP-minimizer the sparse component $x^*_{\ell_1+\mathrm{SLP}}$, which only contains the elements exceeding r in absolute value;
- ℓ_1+IHT, which performs Algorithm 6 with the ℓ_1-minimizer as starting value and the final correction step (3.23). That correction step naturally returns the sparse component $x^*_{\ell_1+\mathrm{IHT}}$ with all entries in absolute value above r.

In the following, we denote with $x^*_{(\cdot)}$ a placeholder for the recovery result of any of the methods that we mentioned above.

In order to properly assess the recovered results $x^*_{(\cdot)}$ with respect to the original sparse signal $\bar{x}$, we compare the following quantities:

(SD) As a measure for the support identification property, i.e., how well the support of the original and recovered vector coincide, the number of elements in the *symmetric difference* (SD) is a convenient quantity. It is denoted by

$$\mathrm{SD}(\bar{x}, x^*_{(\cdot)}) := \#(\mathrm{supp}(\bar{x}) \Delta \, \mathrm{supp}(x^*_{(\cdot)})), \tag{3.50}$$

where the set symmetric difference Δ is defined as follows: $i \in \mathrm{supp}(\bar{x}) \Delta \, \mathrm{supp}(x^*_{(\cdot)})$ if and only if either $i \notin \mathrm{supp}(\bar{x})$ and $i \in \mathrm{supp}(x^*_{(\cdot)})$ or $i \in \mathrm{supp}(\bar{x})$ and $i \notin \mathrm{supp}(x^*_{(\cdot)})$. Thus, $\mathrm{supp}(\bar{x})$ and $\mathrm{supp}(x^*_{(\cdot)})$ are identical if and only if the SD is 0;

(DI) The SD is a relatively simple but effective quantity to estimate the support identification. However, in some applications one might allow a slight tolerance for the support identification, which implies that one is interested in measuring, how far an incorrectly detected entry of $\mathrm{supp}(x^*_{(\cdot)})$ is away from $\mathrm{supp}(\bar{x})$. To put this into a proper measure, we sum up the minimal distances from any entry of $\mathrm{supp}(x^*_{(\cdot)})$ to any entry of $\mathrm{supp}(\bar{x})$. To penalize also the case that $\mathrm{supp}(x^*_{(\cdot)})$ is a strict subset of $\mathrm{supp}(\bar{x})$, we also add the sum of the minimal distances from any entry of $\mathrm{supp}(\bar{x})$ to any entry of $\mathrm{supp}(x^*_{(\cdot)})$. We call the final measure the *support discrepancy* (DI):

$$\mathrm{DI}(\bar{x}, x^*_{(\cdot)}) := \sum_{i \in \mathrm{supp}(x^*_{(\cdot)})} \min_{j \in \mathrm{supp}(\bar{x})} |i-j| + \sum_{i \in \mathrm{supp}(\bar{x})} \min_{j \in \mathrm{supp}(x^*_{(\cdot)})} |i-j|;$$

(AE) The *approximation error* (AE) is the standard ℓ_2-norm difference,

$$\mathrm{AE}(\bar{x}, x^*_{(\cdot)}) := \left\| \bar{x} - x^*_{(\cdot)} \right\|_2 .$$

Naturally, a smaller value in any of those quantities implies a more robust recovery. While the SD and DI make a statement on the robustness of the support identification, the AE only measures the accuracy of the detected entries.

All subsequently presented tests were implemented and run in Matlab R2014a in combination with the CVX toolbox [93, 94], which was used to solve the convex ℓ_1-minimization problem and its variants BP, BPDN, PWBPDN and IRL1, as well as the QCQP problem (3.23).

3.3.2 Parameter Identification

We investigate the performance of the algorithm for parameters of different orders of magnitude. As a prerequisite, let us define the arbitrary exponential grid

$$Q_{i_b,i_f}^{a_0,\kappa} := \{a_0\kappa^i \,|\, i = i_b, \dots, i_f \in \mathbb{Z}\},$$

with positive real a_0, κ, and integer $i_b < i_f$. For each method we explain which parameters are required and from which grid we choose them:

- For AIT(p,q), the regularization parameters λ_p and λ_q are chosen from the grid $Q_{0,30}^{9\text{E-}4,1.25} \times Q_{0,30}^{5\text{E-}4,1.25}$. We furthermore set the number of inner loop iterations to $L = M = 20$ and the starting values to $u^{(0)} = v^{(0)} = 0$. Moreover, $\varepsilon = 0$ since the additional term $\varepsilon\|v\|_2^2$ in (3.28) is only necessary for coercivity in the infinite-dimensional setting. We recall that the thresholding functions for $p \in \{0.3, 0.8\}$ are not given explicitly. Thus, we precomputed them on a grid of points in $[0, 5]$ and interpolated in between, taking also into consideration the jump discontinuity. Respectively, we did the same precomputations for $q \in \{4, 10\}$ on a grid of points in $[0, 1]$.

- For AIT(p,0), $p < 1$, we set the parameters L, ε, $u^{(0)}$ as above and choose λ_p from the grid $Q_{0,30}^{9\text{E-}4,1.25}$. The parameters M and λ_q are not needed (and therefore set to 0).

- To solve the (PW)BPDN, we choose the parameter $\delta \in 0 \cup Q_{-30,5}^{1,1.5}$. Thus, by $\delta = 0$ we also cover the case of the classical ℓ_1-minimization with equality constraint (BP).

- For IRL1, we set the stability parameter a, which avoids the denominator to be zero, to 0.1. In [155] it turned out to be extremely hard to tune the parameter δ in order to obtain the best performances for IRL1 in terms of support identification and accuracy in approximating the large entries of the original vector. In the papers [33, 141] the authors indicated $\delta = \delta_0 := \sqrt{\sigma^2(m + 2\sqrt{2m})}$ as the best

parameter choice for ameliorating the AE with respect to BPDN. However, since in our experiments we wish to determine a suitable parameter, in particular for an ameliorated SD, we choose δ from the grid $Q_{-25,5}^{\delta_0,1.5}$. We execute 8 iterations of IRL1.

- For ℓ_1+SLP, we set the smoothing parameter $\epsilon = 1\text{E-}4$, and choose the regularization parameter $\lambda \in Q_{-5,5}^{4.5,10}$.
- For ℓ_1+IHT, we dispose of the indicating rule (3.19) in order to define the regularization parameter λ. However, this rule depends on the constants δ_{2k} and $\beta(\Phi)$, which are hard to compute. Nevertheless, we can at least roughly bound $0.49 = \eta^2 < \lambda < r^2 = 1$. Thus, we choose $\lambda \in Q_{-15,15}^{0.5,1.25}$, in order to determine an empirical range for λ.

In the following, we investigate in a good parameter choice for the methods AIT(p,q), AIT(p,0), (PW)BPDN, ℓ_1+SLP, IRL1, and ℓ_1+IHT. Therefore we consider the maps $\mathcal{Q}_{**}\colon Q_{i_b,i_f}^{a_0,\kappa} \times Q_{j_b,j_f}^{b_0,\rho} \to \mathbb{N}$ for AIT(p,q), or $\mathcal{Q}_{*}\colon Q_{i_b,i_f}^{a_0,\kappa} \to \mathbb{N}$ for the remaining methods, which assign to a parameter (pair) the value $\mathrm{SD}(\bar{x},x_{(\cdot)}^*)$ of the output $x_{(\cdot)}^*$ of the respective algorithm. For any method and a fixed problem (fixed matrix Φ, measurement y and original signal $\bar{x}$) we are then able to determine the set of best parameters

$$\operatorname*{arg\,min}_{\phi \in Q_{i_b,i_f}^{a_0,\kappa}} \mathcal{Q}_*(\phi), \tag{3.51}$$

or best parameter pairs

$$\operatorname*{arg\,min}_{(\phi,\psi) \in Q_{i_b,i_f}^{a_0,\kappa} \times Q_{j_b,j_f}^{b_0,\rho}} \mathcal{Q}_{**}(\phi,\psi).$$

Notice that the set of best parameters in general is different for each of the 20 test problems. In the following, we make those sets visible, and interpret the respective figures.

We begin with the map $\mathcal{Q}_*$ and therefore the methods AIT(p,0), (PW)BPDN, ℓ_1+SLP, IRL1, and ℓ_1+IHT, where only one parameter is altered. Let us exemplify the visualization of the set (3.51) by the top left plot (AIT($p = 0$,0)) in Figure 3.12. The x-axis represents the different parameter values of λ_p. The y-axis represents the number of the respective test problem. Each horizontal row represents the set of best parameters in the sense that a $\times$-marker is put at the position of the best parameter values (the elements of the set (3.51)). In such a pattern, the number of markers in a vertical column indicates, how often (out of 20 test problems) a parameter was "best" for the respective algorithm. Below each of those columns, we put another marker whose fatness indicates the sum of markers in a column. The fattest markers are colored red. Thus, one is able to quickly identify parameters that lead to the best

performance of an algorithm, independent of the test problem. We draw the following consequences from Figure 3.12:

- In the subfigure IRL1 it is obvious that for nearly half of the test problems, only a very small range of parameters produces the best results. For the remainder of the test problems, nearly all parameters produce the best output, as long as the threshold 4.9E-1 is not exceeded. In particular, we can observe that none of the parameters is optimal for all of the 20 test problems. This indicates, that, in applications, it will be very hard to determine a priori a good parameter for IRL1 for robust support detection.

- In the subfigure ℓ_1+IHT, we observe that each $\lambda \in [0.1, 0.62]$ is the best parameter for at least 17 out of 20 test problems. Note that this interval overlaps with the theoretically estimated interval $[0.49, 1]$. As already mentioned in the previous paragraph, due to the difficult computation of δ_{2k} and $\beta(\Phi)$, we were not able to practically compute the upper bound (and we loosely estimated it by 1). Through this empirical investigation, it is possible to identify this bound by 0.62. Furthermore, these experiments show that the theoretical lower bound 0.49 can be even extended to the much smaller value 0.1, where the method still provides results of similar robustness.

- In the subfigure ℓ_1+SLP, we observe that the method produces equally good results, choosing λ in the comparably huge interval $\lambda \in [4.5\text{E-}2, 4.5\text{E+}3]$. We conclude that the parameter tuning is easy for this method.

- In the subfigures concerning AIT(p,0), $p < 1$, we observe that for $p \in \{0.5, 0.8\}$ there is a relatively large interval for the parameter λ_p, where for a wide range of test problems the best results are obtained. For the problems with strong non-convexity $p \in \{0, 0.3\}$, those intervals are smaller. While the best parameter for $p = 0$ is only able to cover half of the test problems, for the remaining parameters p, the best parameter covers at least 16 out of 20 test problems.

- In the subfigures concerning BPDN and PWBPDN, we observe that both methods produce best results for a large range of parameters where at least 19 out of 20 test problems are covered. It is noteworthy that the best results are only obtained for small values of δ.

For the map $\mathcal{Q}_{**}$ and therefore the methods AIT(p,q), the two parameters λ_p and λ_q are altered. To adopt the visualization idea of the previous paragraph, a three-dimensional visualization is necessary (two axes for the parameters and the third for the problem number). While such a three-dimensional plot is an appropriate presentation in a dynamic environment as, e.g., on a computer display, where one can turn the plot, in our case it is rather counterproductive to put it on the paper and pre-define the

perspective for the reader. That is why we ignore the third dimension (the problem number), and only extend the concept of the vertical sum, which is indicated in the previous Figure 3.12 by markers of different fatness, from the one-dimensional to a two-dimensional visualization. We present the results in Figure 3.13, where we add blue markers with a fatness from 1 to 20, as a legend, to help the reader to visually classify the fatness of the black and red markers. The evaluation of Figure 3.13 is as follows:

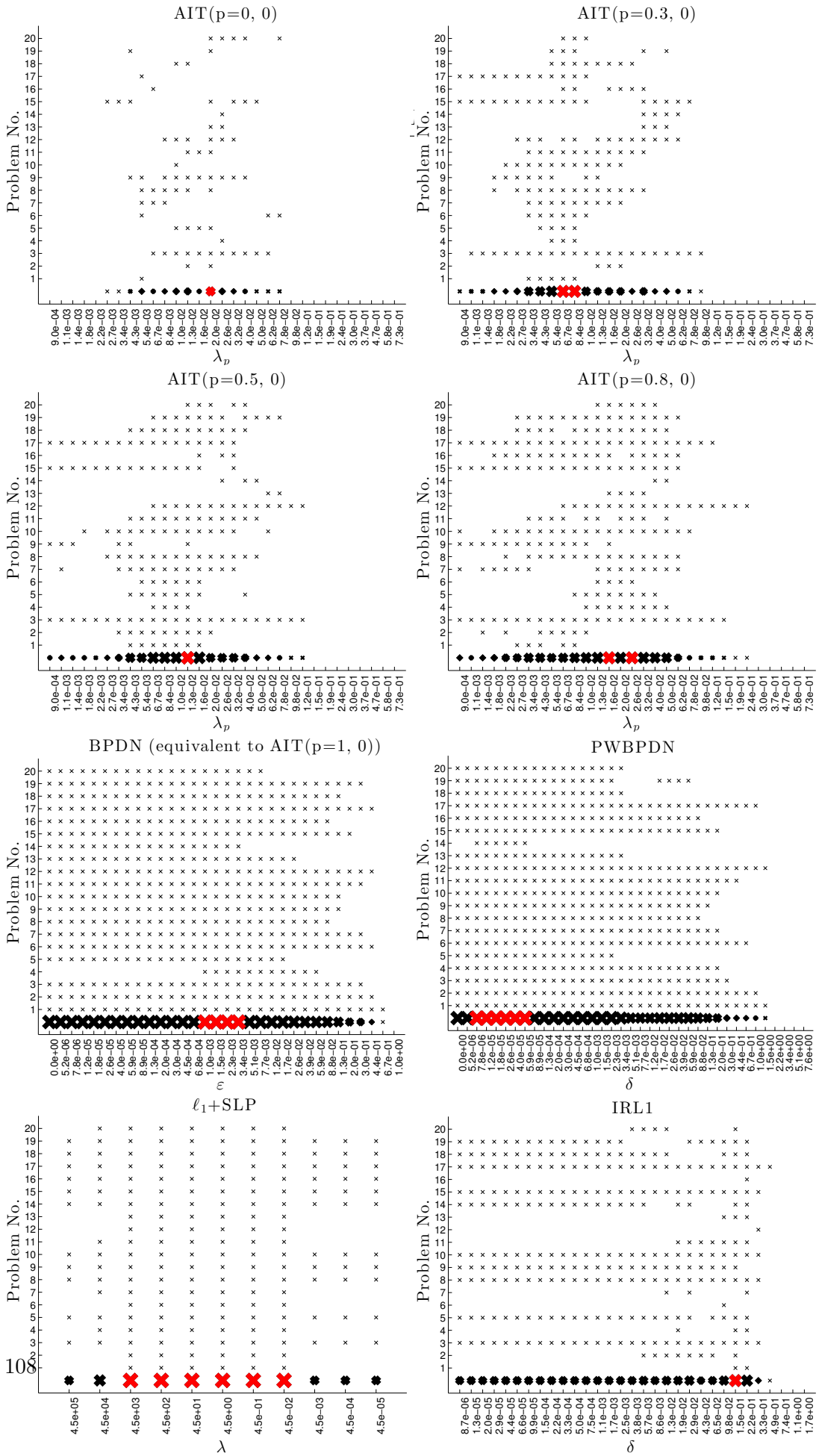
- For $p = 0$ the markers are not particularly fat. It seems that it is nearly impossible to determine a parameter pair for the method, such that the method performs well on all test problems.
- For $p > 0$ the size of the region of good parameter pairs (the ones which are marked fat) is decreasing when q increases. In particular for $q = 2$ the algorithm is not very sensitive with respect to the choice of the parameters.
- Obviously the choice of $0 < p < 1$ does provide parameter pairs which are best for a large percentage of the test problems (markers are particularly fat).

So far, we exclusively considered the problem of a good parameter choice, but we did not compare the actual outcome (SD, DI, AE) for any of those parameters or parameter pairs. This means that even a huge interval that provides "best" parameters is worth nothing if the method produces a bad SD value by these parameters. In the next section, we present this final evaluation in order to obtain the big picture on the performance of the entirety of all methods.

3.3.3 Massive Computations

Remind, that the simulations are supposed to answer the two questions, first, which of the methods presented above has the highest potential, and second, which is the most useful in applications in terms of a robust support identification. The first question is answered if we allow an optimal choice of the parameters depending on the test problem. Thus, according to the parameter identification results presented in Figures 3.12 and 3.13, we choose for each of the 20 test problems the respective "best" parameter (pair) and determine the mean value of SD, DI, and AE of the corresponding output. To answer the second question, we choose the parameter independently of the test problem, i.e., the parameter for which the algorithm performs best for the most of the 20 test problems. Those parameters are presented for each method by red markers. The outcome of the evaluation with variable and fixed parameters is presented in Figure 3.14(a) and Figure 3.14(b) respectively.

The results of Figure 3.14(a) can be interpreted as follows:


AIT(p=0, 0)
AIT(p=0.3, 0)
AIT(p=0.5, 0)
AIT(p=0.8, 0)
BPDN (equivalent to AIT(p=1, 0))
PWBPDN
ℓ_1+SLP
IRL1
Problem No.
λ_p
ε
δ
λ

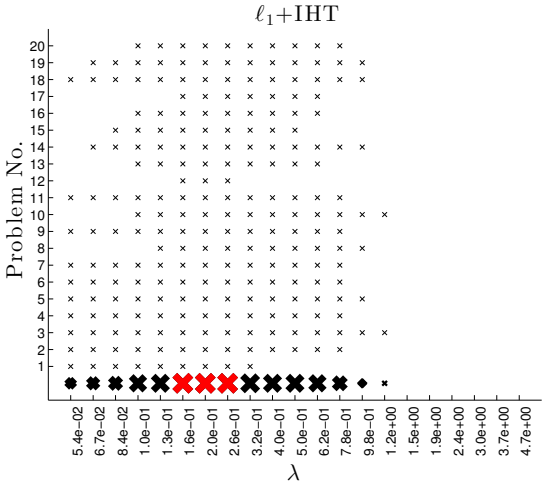
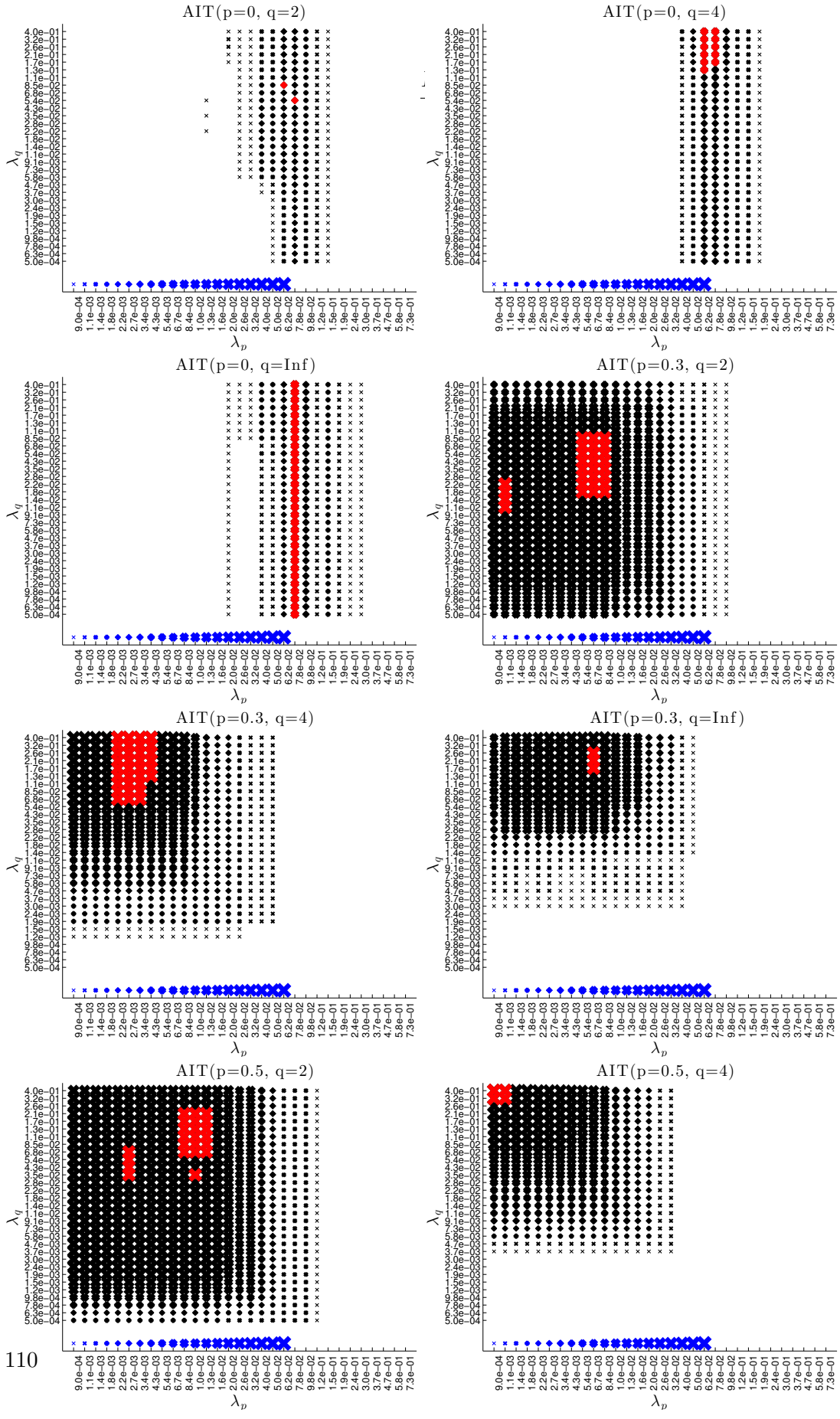


Figure 3.12: For the parameters λ_p (AIT(p,0), $p < 1$), δ ((PW)BPDN, and IRL1), and λ (ℓ_1+SLP, ℓ_1+IHT) respectively, we plot for each of the 20 trial problems row-wise a $\times$-marker in the column of the parameter value, where an optimum in terms of SD was attained. In the bottom row, the sum of markers in each column are presented by markers of different fatness. The fattest markers are colored red.

- Regarding the output of the algorithms AIT(p,q), i.e., the first five bar groups, the most surprising result is the fact that the use of a parameter $0 < p < 1$ is for any of the three quantities SD, DI, AE much better than choosing $p \in \{0, 1\}$. Furthermore, the choice of $q = 2$ is most favorable in our particular setting (simulating compressed sensing problems). In terms of SD and DI, the algorithm AIT(0.8,2) clearly performs best. In terms of AE, it is AIT(0.3,2).

- Comparing for any p the multi-penalty approach AIT(p,q) and the respective mono-penalty approach AIT(p,0), the main observation is that for $p \in \{0, 1\}$, it is a disadvantage to use multiple penalties, while for $0 < p < 1$ the multi-penalty approach is advantageous. Note, that this result contradicts at first sight the findings in the paper [140], where we showed that the multi-penalty should be always preferred over the mono-penalty approach. On a second glance, it is not a contradiction since we assume two different test settings: The subtle difference is that we assume in the present section that the threshold r is known. Thus, the further filtering of the relevant entries with absolute value above r cleans the result from possibly wrongly detected support entries. Having the threshold r not at disposal, the evaluation results would agree with the results in the paper [140].

- As already indicated by the results in Section 3.1.2, the classical ℓ_1-minimization (with and without prewhitening and inequality constraint) is greatly outperformed by ℓ_1+SLP, IRL1, and ℓ_1+IHT. In particular ℓ_1+SLP and ℓ_1+IHT outperform also IRL1. The method ℓ_1+IHT even displays an SD mean value of 0, which means that it is able to correctly recover the exact support of all 20 trials. Furthermore, the method AIT(0.8,2) shows a very low SD value, which is better than the one of ℓ_1+SLP.


AIT(p=0, q=2)
AIT(p=0, q=4)
AIT(p=0, q=Inf)
AIT(p=0.3, q=2)
AIT(p=0.3, q=4)
AIT(p=0.3, q=Inf)
AIT(p=0.5, q=2)
AIT(p=0.5, q=4)
λ_q
λ_p

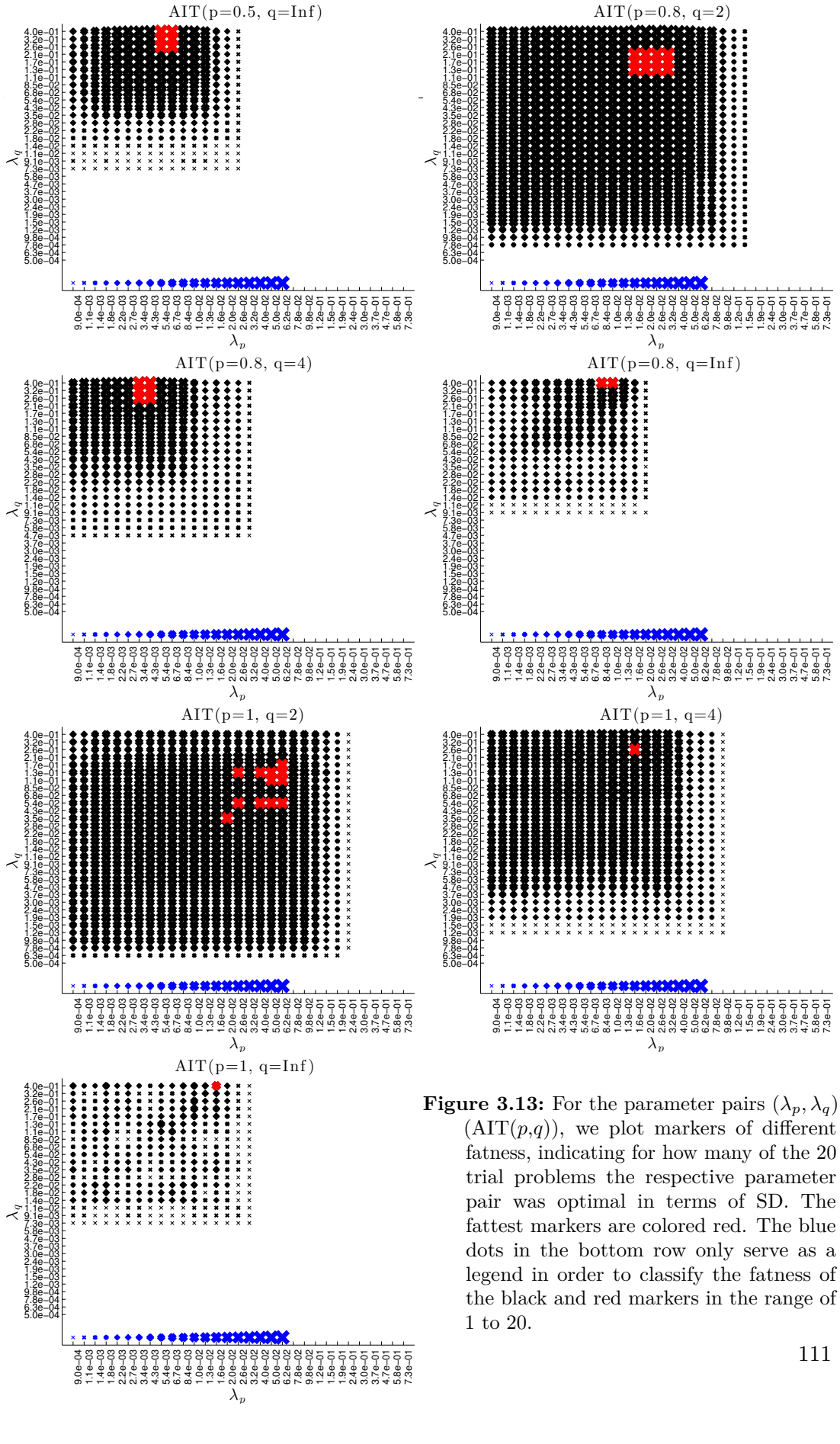
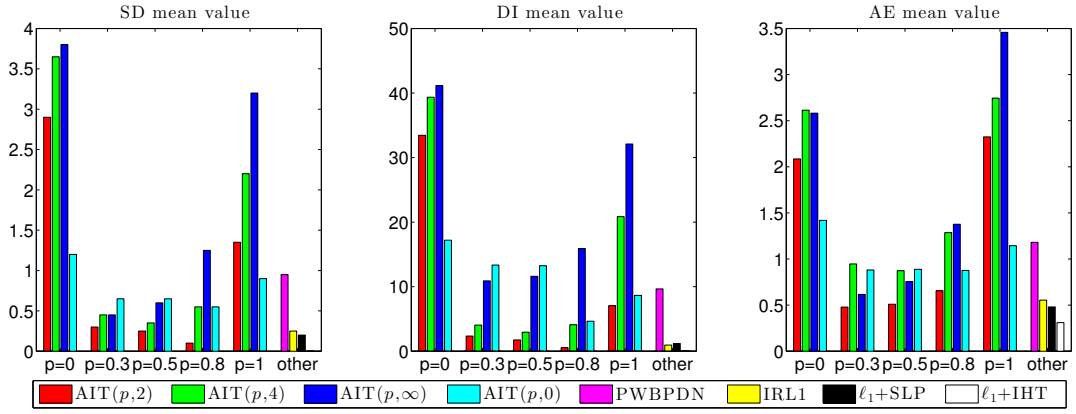
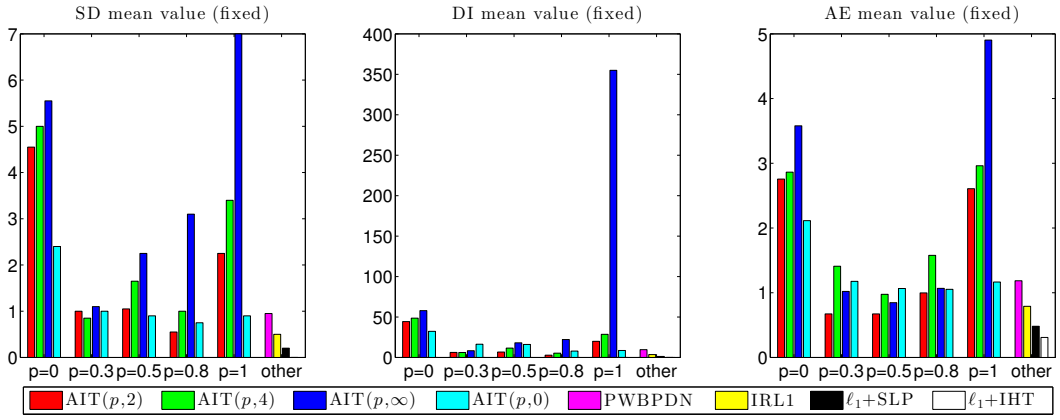


Figure 3.13: For the parameter pairs (λ_p, λ_q) (AIT(p,q)), we plot markers of different fatness, indicating for how many of the 20 trial problems the respective parameter pair was optimal in terms of SD. The fattest markers are colored red. The blue dots in the bottom row only serve as a legend in order to classify the fatness of the black and red markers in the range of 1 to 20.



(a)



(b)

Figure 3.14: The bar plots present the evaluation of the mean value of SD, DI, and AE for the methods AIT(p,q) (first five bar groups), ℓ_1-minimization, ℓ_1+SLP, IRL1, and ℓ_1+IHT (last bar group); compare the legend. In Subfigure 3.14(a), we allow a flexible parameter choice (and choose the best result) and in Subfigure 3.14(b), we fixed the most promising parameter for each method respectively.

- The situation slightly changes regarding the quantity DI. Naturally ℓ_1+IHT has the mean value 0, but when the support is not exactly recovered, it seems that the methods AIT(0.8,2) and IRL1 produce at least results whose support entries are closer to the expected entries than ℓ_1+SLP.
- The quantity AE does behave similar to the quantity SD with the difference that, as already mentioned above, the methods AIT(0.3,2) and AIT(0.5,2) perform

slightly better than AIT(0.8,2), but they are not able to outperform ℓ_1+SLP and ℓ_1+IHT.

- An interesting and surprising result is that PWBPDN does not improve the results of AIT(1,0) (alias BPDN).

In view of the above observations, we conclude that in particular the methods AIT(0.8,2), ℓ_1+SLP, IRL1, and ℓ_1+IHT have the potential to provide robust support identification results with comparably low SD, DI, and AE value. However, choosing the parameters according to the respective test problem is a difficult task in practice. One would rather identify and fix globally valid parameters for a class of problems by simulations or trials where the ground truth is known. The results of such a simulation is presented in Figure 3.14(b), and can be interpreted as follows:

- Regarding the output of the algorithms AIT(p,q), the results for $q = 2$ are still the best, and $0 < p < 1$ can be preferred over $p \in \{0, 1\}$.
- Within the comparison of multi-penalty and mono-penalty approach, it is apparent that the mono-penalty result produces better SD results than the respective multi-penalty method, except for AIT(0.8,2). In DI and AE the mono-penalty approach is still outperformed by the multi-penalty approach for $0 < p < 1$.
- The methods ℓ_1+SLP and ℓ_1+IHT are extremely stable and outperform the remaining methods in terms of any of the quantities SD, DI, and AE. Acceptable results in terms of SD and DI are also obtained by the methods AIT(0.8,2) and IRL1. However, the AE value accounts negatively for AIT(0.8,2).

The presented experiments confirm the superiority of the methods ℓ_1+SLP and ℓ_1+IHT over the classical ℓ_1-minimization and its reweighted version in terms of a robust support recovery, although IRL1 has the potential to keep up in terms of DI if one has the right parameters for the respective problem at disposal. Nevertheless, we showed in the previous subsection that it is very hard to tune the parameter δ for IRL1, while it is much easier for the other two methods. The method AIT(0.8,2) does provide acceptable results without outperforming the previously mentioned methods. However, one has to mention a crucial advantage of AIT(0.8,2): it provides a separation of the sparse and noisy part of the result even without the knowledge of the parameter r. Furthermore, we showed in Figure 3.13, that the choice of the respective parameters λ_p and λ_q is easy since the method provides stable results over a comparably large interval of both parameters.

3.3.4 Phase Transition Diagrams

In the last subsection, we again investigate into the case where we have the threshold r at disposal. The essence of the previous section was that the methods ℓ_1+IHT and

ℓ_1+SLP outperform IRL1 in terms of SD, DI, and AE, if the parameter is fixed. We eventually want to extend the results, given in Figure 3.14(b) for a wider range of m and k. In Figure 3.15, we present phase transition diagrams of success rates in support recovery in presence of nearly maximally allowed noise, i.e., we slightly change the parameters $0.8 = r > \eta = 0.75$. We further use BP as reference algorithm.

To produce phase transition diagrams, we varied the dimension of the measurement vector $m = 1, \ldots, N$ with $N = 100$, and solved 20 different problems for all the admissible $k = \#S_r(x) = 1, \ldots, m$. We colored black all the points (m, k), with $k \leq m$, which reported 100% of correct support identification, and gradually we reduce the tone up to white for the 0% result. The level bound of 50% and 90% is highlighted by a magenta and red line respectively. A visual comparison of the corresponding phase transitions confirms our previous expectations. In particular, ℓ_1+SLP and ℓ_1+IHT very significantly outperform BP in terms of correct support recovery. The difference of both methods towards IRL1 is less significant but still important, which confirms the ranking of the methods, which we already observed in Figure 3.14(a). In the bottom two subfigures of Figure 3.15, we compare the level bounds of 50% and 90% among the four different methods. Observe that the 90% probability bound indicates the largest positive region for ℓ_1+IHT, followed by ℓ_1+SLP, and by IRL1, while the bounds are much closer to each other in the case of the 50% bound. Thus, we confirm that ℓ_1+IHT works in practice better than ℓ_1+SLP for some range of m, and offers the most stable support recovery results.

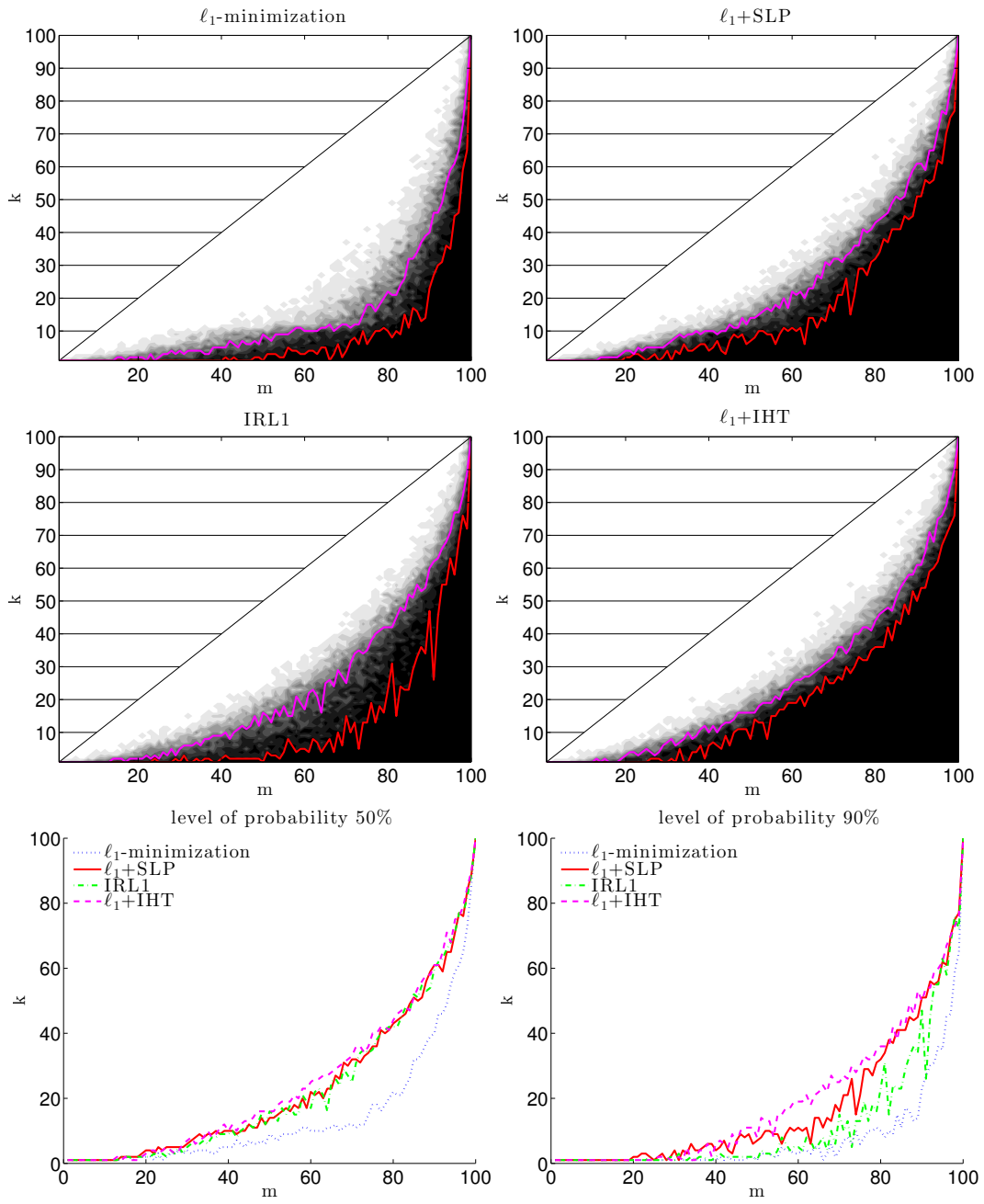


Figure 3.15: Top four subfigures (phase transition diagrams): Phase transition diagrams for BP, ℓ_1+SLP, IRL1, and ℓ_1+IHT. The black area represents the couple (m, k) for which we had 100% of support recovery. Note that the area for $k > m$ is not admissible. The red line shows the level bound of 90% of support recovery, and the magenta line 50% respectively.
Bottom two subfigures: Comparison of phase transition diagrams for BP (dark blue, dotted), ℓ_1+SLP (red), IRL1 (green, dash-dotted), and ℓ_1+IHT (magenta, dashed). The level bound of 50% and 90% as it is displayed in the top four subfigures is compared in the bottom two subfigures respectively. The methods ℓ_1+IHT and ℓ_1+SLP provide highest stability.

Chapter 4

Acceleration Techniques for Sparse Recovery Algorithms

In an environment where more and more data is routinely produced, and the need for higher precision and more detailed results is constantly growing in scientific and industrial applications, also sparse recovery problems are gaining in size and quantity. Since the last decade, where sparse recovery techniques were brought into focus, the treatment of big data problems was always a companion and one of the main drivers for new developments in this research field. In particular the accelerated progress of computer technology, including growing memory sizes and processor speed, did not only help to solve existing problems, but allowed to create new problems of immense data size, leading to an spiral of better and faster solutions for larger and larger problems. While Moore's Law [137], which ensured growing technology for growing problem sizes, may reach its limit soon, due to disproportional development and production costs of new processors, distributed solutions like multi-core processors and cloud computing obtained more attention in the last couple of years and broke their way in the today's quotidian technology.

Based on the fundamental algorithmic concepts that were presented in Section 2.4, we propose in this section two acceleration techniques, which build upon two very different fundamental settings.

If an encoder of a sparse recovery problem is very large but has a certain structure, it is often the case that one can provide an explicit fast matrix-vector multiplication, e.g., for sparse matrices or randomly sampled partial Fourier/cosine transform matrices. In such a case one considers a respective efficient implementation of the matrix-vector multiplication and one does not need to store the full matrix in the memory. However, this technique requires algorithms that do not transform the matrix directly but are able to use the efficient matrix-vector multiplications, e.g., iterative hard and soft thresholding (IHT, ISTA, FISTA). Contrarily, methods like iteratively reweighted least squares (IRLS) need to solve a (weighted) least squares problem in each iteration. This task, which is essentially the solution of a linear system, can be carried out by using direct methods like Gaussian elimination in order to obtain an exact solution.

However those methods work on the matrix directly, and one cannot apply efficient matrix-vector multiplications. Thus, it is more appropriate to use iterative solvers such as the conjugate gradient method (CG), which again have the drawback that they produce in general only an approximate solution. In Section 4.1, we consider IRLS, equipped by CG for the solution of the linear systems. Since a proper analysis of the convergence of this method has not been reported in the literature so far, we clarify in detail—specifically for problems involving matrices Φ with certain spectral properties such as the Null Space Property—how accurately one needs to solve the linear systems by means of CG in order to guarantee convergence and possibly also asymptotic (super-)linear convergence rates. In addition to the analysis we present numerical tests on simulated medium to large scale problems, where we show that the algorithm has the potential of outperforming state-of-the-art methods such as IHT or FISTA. The content of Section 4.1, was published by the author of this dissertation as leading author in [82].

The second acceleration technique that we present, is motivated by the above mentioned recent developments in parallel multi-core architectures and super-computing. In contrast to the previous setting, we assume that the encoder cannot be stored and evaluated efficiently. The only option is to represent it as a full matrix. If we want to store such a full matrix on a computer with entries of the datatype `double` with, e.g., dimensions 50000×100000, we need about 35 GB of memory, which does not fit in the RAM of the most machines of today. Such a requirement leads to the necessity of distributing data and calculations on more than one processor. Also, if the dimensions of a problem are small, one may have to solve several thousands of that problem. Again, one would be required to run many problems in parallel in order to get a result in a reasonable time. While in the second scenario each problem is run on each core, and "only" an efficient scheduling is needed, in the first scenario the problem is unsolvable as long as one does not dispose of parallel algorithms. In Section 4.2, we recall respective parallel solutions for the ℓ_1-regularized least squares problems (2.12) and (2.15). In this scope, we present and analyze a tuned domain-decomposition method, and compare it in numerical experiments to the state-of-the-art.

4.1 A Conjugate Gradient Based Acceleration of Iteratively Re-weighted Least Squares Algorithms

Iteratively re-weighted least squares (IRLS) (Section 2.4.1) is one of the most immediate and intuitive approaches towards non-standard optimizations such as (2.3), i.e.,

$$\operatorname*{arg\,min}_{z \in \mathcal{F}_\Phi(y)} \|z\|_{\ell_p}^p,$$

for the reason that it is based on a relatively simple reformulation of the initial potentially non-convex and non-smooth minimization problem into a more familiar and easily

solvable quadratic optimization. It is perhaps one of the first and popular algorithms beginner practitioners consider for their first experiments. However, despite its simplicity, versatility, and elegant analysis, IRLS does not outperform in general well-established first order methods, which have been proposed recently for similar problems, such as iterative hard thresholding (IHT, Section 2.4.3.2) [20] or fast iterative soft thresholding algorithm (FISTA, Section 2.4.3.1) [13]; see the numerical experiments further below in Section 4.1.4, Figure 4.1 and 4.4. In fact, its complexity very strongly depends on the way the solution of the successive quadratic optimizations is addressed, whether one uses preconditioned iterative methods and exploits fast matrix-vector multiplications or just considers simple direct linear solvers. If the dimensions of the problem are not too large or the involved matrices have no special structure allowing for fast matrix-vector multiplications, then the use of a direct method such as Gaussian elimination can be appropriate. When instead the dimension of the problem is large and one can take advantage of the structure of the matrix to perform fast matrix-vector multiplications (e.g., for partial Fourier or partial circulant matrices), then it is appropriate to use iterative solvers such as the conjugate gradient method (CG). The use of CG in the implementation of IRLS is appearing, for instance, in [188] towards total variation minimization and in [189] towards ℓ_1-norm minimization. However, the price to pay is that such solvers will return only an approximate solution whose precision depends on the number of iterations. A proper analysis of the convergence of the perturbed method in this case has not been reported in the literature so far. Thus, we clarify in this section how accurately one needs to solve the quadratic problems by means of CG in order to guarantee convergence and possibly also asymptotic (super-)linear convergence rates.

Besides analyzing the effect of CG in an IRLS for problems of the type (2.3), we further extend it in Section 4.1.3 to a class of problems of the type (2.39), i.e.,

$$\arg\min_{z\in\mathbb{R}^N} \left(F_{p,\lambda}(z) := ||z||_{\ell_p}^p + \frac{1}{2\lambda}||\Phi z - y||_{\ell_2}^2 \right),$$

for $0 < p \leq 1$, and $\lambda > 0$, used for sparse recovery in signal processing. The problem is equivalent to the ℓ_p-regularized least squares problem (2.14), if "2λ" is replaced by "λ". In order to ease the cross-reading with [82], we prefer to use the formulation (2.39) in this section. In the work [120, 189, 190] a convergence analysis of IRLS towards the solution of (2.39) has been carried out with two limitations:

1. In [120] the authors do not consider the use of an iterative algorithm to solve the appearing system of linear equations and they do not show the behavior of the algorithm when the measurements y are given with additional noise;

2. Also in [189, 190] a precise analysis of convergence is missing when iterative

methods are used to solve the intermediate sequence of systems of linear equations. Furthermore the non-convex case of $p < 1$ is not specifically addressed.

Regarding these gaps, we contribute in this section by

- giving a proper analysis of the convergence when inaccurate CG solutions are used;
- extending the results of convergence in [189, 190] to the case of $0 < p < 1$ by combining our analysis with findings in [158, 195];
- performing numerical tests which evaluate possible speedups via the CG method, also taking problems into consideration where measurements may be affected by noise.

Our work on CG accelerated IRLS for (2.39) does not analytically address rates of convergence because this turned out to be a very technical task.

As mentioned above, we illustrate the theoretical results of this section by several numerical experiments. In order to emphasize the value of those experiments, we anticipate the main outcome and briefly comment it: We first show that our versions of IRLS yield significant improvements in terms of computational time and may outperform state-of-the-art first order methods such as IHT and FISTA, especially in high dimensional problems ($N \geq 10^5$). These results are somehow both surprising and counterintuitive as it is well-known that first order methods should be preferred in higher dimension. However, they can be easily explained by observing that in certain regimes preconditioning in the conjugate gradient method (as we show at the end of Subsection 4.1.4.3) turns out to be extremely efficient. This is perhaps not a completely new discovery, as benefits of preconditioning in IRLS have been reported already in minimization problems involving total variation terms [188]. The second significant outcome of our experiments is that CG-IRLS not only is faster than state-of-the-art first order methods, but also shows higher recovery rates, i.e., requires less measurements for successful sparse recovery. This will be demonstrated with corresponding phase transition diagrams of empirical success rates (Figure 4.3).

In the following, we revisit conjugate gradient methods in Section 4.1.1, before turning to the description and analysis of a CG accelerated IRLS for (2.3) in Section 4.1.2, and for (2.39) in Section 4.1.3. We conclude with the numerical simulations in Section 4.1.4.

4.1.1 Conjugate Gradient Methods Revisited

We summarize the fundamental formulation and respective convergence results of conjugate gradient methods here, in order to facilitate the reading and to provide

a proper notation of the variables that are used in the CG algorithm for the later analysis.

4.1.1.1 Conjugate Gradient Method (CG)

The CG method was originally proposed by Stiefel and Hestenes in [108]. For a positive definite matrix $A \in \mathbb{R}^{N\times N}$ the CG method solves the linear equation $Ax = y$ or equivalently the minimization problem

$$\arg\min_{x\in\mathbb{R}^N} \left(F(x) := \frac{1}{2} x^* Ax - x^* y \right).$$

The algorithm is designed to iteratively compute the minimizer x^i of F on the affine subspace $\tilde{V}_i := x^0 + V_i$ with V_i being the Krylov subspace $V_i := \operatorname{span}\{r^0, Ar^0, \ldots, A^{i-1}r^0\} \subset \mathbb{R}^N$, $x^0 \in \mathbb{R}^N$ a starting vector, and $r^0 := y - Ax^0$ (*minimality property of CG*).

Algorithm 11 Conjugate Gradient (CG) method

Input: initial vector $x^0 \in \mathbb{R}^N$, matrix $A \in \mathbb{R}^{N\times N}$, given vector $y \in \mathbb{R}^N$ and optionally a desired accuracy δ.

1: Set $r^0 = p^0 = y - Ax^0$ and $i = 0$
2: **while** $r^i \neq 0$ (or $\|r^i\|_{\ell_2} > \delta$) **do**
3: $\quad a_i = \langle r^i, p^i\rangle_{\ell_2} / \langle Ap^i, p^i\rangle_{\ell_2}$
4: $\quad x^{i+1} = x^i + a_i p^i$
5: $\quad r^{i+1} = y - Ax^{i+1}$
6: $\quad b_{i+1} = \langle Ap^i, r^{i+1}\rangle_{\ell_2} / \langle Ap^i, p^i\rangle_{\ell_2}$
7: $\quad p^{i+1} = r^{i+1} - b_{i+1} p^i$
8: $\quad i = i + 1$
9: **end while**

Roughly speaking, CG iteratively searches for a minimum of the functional F along conjugate directions p^i with respect to A, i.e., $(p^i)^* Ap^j = 0$, $j < i$. Thus, in step $i+1$ of CG the new iterate x^{i+1} is found by minimizing $F(x^i + a_i p^i)$ with respect to the scalar $a_i \in \mathbb{R}$ along the search direction p^i. Since we perform a minimization in each iteration, this implies monotonicity of the iterates, $F(x^{i+1}) \leq F(x^i)$. If the algorithm produces at some iteration a residual $r^i = 0$, then a solution of the linear system is found. Otherwise it produces a new conjugate direction p^i. One can show that the conjugate directions $p^0, \ldots, p^{i-1}$ also span V_i. Since the conjugate directions are linear independent, we have $V_N = \mathbb{R}^N$ (assumed that $r^i \neq 0$, $i = 0, \ldots, N-1$). Then, according to the above mentioned minimality property, the iterate x^N is the minimizer of F on $\mathbb{R}^N$, which means that CG terminates after at most N iterations. Nevertheless, the algorithm can be stopped after a significantly smaller number of steps

as soon as the machine precision is very high and theoretically convergence already occurred. In view of propagation of errors in practice the algorithm may be run longer than just N iterations though.

The following theorem establishes the convergence and the convergence rate of CG.

Theorem 4.1 ([157, Theorem 4.12])
Let the matrix A be positive definite. The Algorithm CG converges to the solution of the system $Ax = y$ after at most N steps. Moreover, the error $x^i - x$ is such that

$$\left\|A^{\frac{1}{2}}(x^i - x)\right\|_{\ell_2} \leq \frac{2c_A^i}{1 + c_A^{2i}} \left\|A^{\frac{1}{2}}(x^0 - x)\right\|_{\ell_2}, \quad \text{with } c_A = \frac{\sqrt{\kappa_A} - 1}{\sqrt{\kappa_A} + 1} < 1,$$

where $\kappa_A = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}$ is the condition number of the matrix A and $\sigma_{\max}(A)$ (resp. $\sigma_{\min}(A)$) is the largest (resp. smallest) singular value of A.

Remark 4.2
Since $\kappa_A \geq 1$, it follows that $0 \leq c_A < 1$, and also $0 \leq c_A^i < 1$, for positive iteration numbers i. From $0 < (1 - c_A^i)^2 = 1 + c_A^{2i} - 2c_A^i$, we immediately see that $2c_A^i/(1 + c_A^{2i}) < 1$ for all $i \in \mathbb{N}$, and obviously $2c_A^i/(1 + c_A^{2i}) \to 0$ for $i \to +\infty$.

4.1.1.2 Modified Conjugate Gradient Method (MCG)

In the introduction towards the basic IRLS algorithm, in Section 2.4.1.1, we explained that we have to solve in Step 2 of Algorithm 1 the weighted least-squares problem of the form

$$\hat{x} = \operatorname*{arg\,min}_{x \in \mathcal{F}_\Phi(y)} \|x\|_{\ell_2(w)},$$

given $\Phi \in \mathbb{R}^{m \times N}$ with $m \leq N$. As we show in the same section, the minimizer $\hat{x}$ is given explicitly by the (weighted) Moore-Penrose pseudo-inverse

$$\hat{x} = D\Phi^*(\Phi D \Phi^*)^{-1} y,$$

where $D := \operatorname{diag}[w_i^{-1}]_{i=1}^N$. Hence, in order to determine $\hat{x}$, we first solve the system

$$\Phi D \Phi^* \theta = y, \tag{4.1}$$

and then we compute $\hat{x} = D\Phi^*\theta$. Notice that the system (4.1) has the general form

$$BB^*\theta = y, \tag{4.2}$$

with $B := \Phi D^{\frac{1}{2}}$. We consider the application of CG to this system for the matrix $A = BB^*$. This approach leads to the modified conjugate gradient (MCG) method, presented in Algorithm 12, and proposed by J.T. King in [117]. It provides a sequence $(\theta^i)_{i \in \mathbb{N}}$ with

$\theta^i \in U_i := \operatorname{span}\{y, BB^*y, \ldots, (BB^*)^{i-1}y\}$, the Krylov subspace associated to (4.2), with the property that $\bar{x}^i := B^*\theta^i$ minimizes $\|\bar{x}^i - \bar{x}\|_{\ell_2}$, where $\bar{x} = \underset{x \in \mathcal{F}_B(y)}{\arg\min} \|x\|_{\ell_2}$. Finally, we compute $\hat{x} = D^{\frac{1}{2}}\bar{x}$.

Algorithm 12 Modified Conjugate Gradient (MCG) Method

Input: initial vector $\theta^0 \in \mathbb{R}^m$, $B \in \mathbb{R}^{m\times N}$, $y \in \mathbb{R}^m$, desired accuracy δ (optional).

Set $\rho^0 = p^0 = y$ and $i = 0$
while $\rho^i \neq 0$ (or $\|\rho^i\|_{\ell_2} > \delta$) **do**
  $\alpha_i = \langle \rho^i, p^i\rangle_{\ell_2} / \|B^*p^i\|_{\ell_2}^2$
  $\theta^{i+1} = \theta^i + \alpha_i p^i$
  $\rho^{i+1} = y - BB^*\theta^{i+1}$
  $\beta_{i+1} = \langle B^*p^i, B^*\rho^{i+1}\rangle_{\ell_2} / \|B^*p^i\|_{\ell_2}^2$
  $p^{i+1} = \rho^{i+1} - \beta_{i+1}p^i$
  $i = i + 1$
end while
Set $\bar{x}^{i+1} = B^*\theta^{i+1}$

The following theorem provides a precise rate of convergence of MCG. Additionally, we emphasize the monotonic decrease of the error $\|\hat{x}^i - \hat{x}\|_{\ell_2(w)}$, which we use below in Lemma 4.19.

Theorem 4.3
Suppose the matrix B to be surjective. Then the sequence $(\bar{x}^i)_{i\in\mathbb{N}}$ generated by the Algorithm MCG converges to $\bar{x} = B^*(BB^*)^{-1}y$ in at most N steps, and

$$\left\|\bar{x}^i - \bar{x}\right\|_{\ell_2} \leq \frac{2c_{BB^*}^i}{1 + c_{BB^*}^{2i}} \left\|\bar{x}^0 - \bar{x}\right\|_{\ell_2}, \text{ with } c_{BB^*} < 1, \tag{4.3}$$

for all $i \geq 0$, where $c_{BB^*} = \frac{\sqrt{\kappa(BB^*)}-1}{\sqrt{\kappa(BB^*)}+1} = \frac{\sigma_{\max}(B)-\sigma_{\min}(B)}{\sigma_{\max}(B)+\sigma_{\min}(B)}$ is defined as in Theorem 4.1, and $\bar{x}^0 = B^*\theta^0$ is the initial vector. Moreover, by setting $D := \operatorname{diag}[w_i^{-1}]_{i=1}^N$, and $\hat{x}^i = D^{\frac{1}{2}}\bar{x}^i$ as well as $\hat{x} = D^{\frac{1}{2}}\bar{x}$, we obtain

$$\left\|\hat{x}^i - \hat{x}\right\|_{\ell_2(w)} \leq \frac{2c_{BB^*}^i}{1 + c_{BB^*}^{2i}} \left\|\hat{x}^0 - \hat{x}\right\|_{\ell_2(w)}. \tag{4.4}$$

Proof. By Theorem 4.1, we have

$$\left\|(BB^*)^{\frac{1}{2}}(\theta^i - \theta)\right\|_{\ell_2} \leq \frac{2c_{BB^*}^i}{1 + c_{BB^*}^{2i}} \left\|(BB^*)^{\frac{1}{2}}(\theta^0 - \theta)\right\|_{\ell_2},$$

for θ as given in (4.2). By the identity

$$\left\|(BB^*)^{\frac{1}{2}}(\theta^i-\theta)\right\|_{\ell_2}^2 = \langle (BB^*)^{\frac{1}{2}}(\theta^i-\theta), (BB^*)^{\frac{1}{2}}(\theta^i-\theta)\rangle_{\ell_2} = \langle (BB^*)(\theta^i-\theta), \theta^i-\theta\rangle_{\ell_2}$$
$$= \langle B^*(\theta^i-\theta), B^*(\theta^i-\theta)\rangle_{\ell_2} = \langle \bar{x}^i-\bar{x}, \bar{x}^i-\bar{x}\rangle_{\ell_2} = \left\|\bar{x}^i-\bar{x}\right\|_{\ell_2}^2,$$

we obtain the assertion (4.3). Inequality (4.4) follows then from the definition of the diagonal matrix D and the weighted norm $\ell_2(w)$. The fact that the coefficient $2c^i_{BB^*}/(1+c^{2i}_{BB^*}) < 1$ for all $i \in \mathbb{N}$, and $2c^i_{BB^*}/(1+c^{2i}_{BB^*}) \to 0$ for $i \to \infty$ follows as in Remark 4.2. □

4.1.2 Conjugate Gradient Accelerated IRLS Method for ℓ_p-norm Minimization

While we thoroughly introduced the basic IRLS algorithm of [48] in Section 2.4.1.1, we present in the following the modified algorithm that uses CG for the solution of the successive quadratic optimization problems. Afterwards, we provide the results concerning the convergence and the rate of convergence of the modified algorithm. As crucial feature, we give bounds on the accuracies of the (inexact) CG solutions of the intermediate least squares problems, which ensure convergence of the overall IRLS method. In particular, these tolerances must depend on the current iteration and should tend to zero with increasing iteration count. In fact, without this condition, one may observe divergence of the method. The proofs of the theorems are developed into several lemmas.

We recall that p is a fixed parameter such that $0 < p \leq 1$. At some points of the presentation, we explicitly switch to the case of $p = 1$ to prove additional properties of the algorithm which are due to the convexity of the ℓ_1-norm minimization problem.

Instead of solving *exactly* the system of linear equations (2.28) occurring in Step 2 of Algorithm 1 (IRLS), we substitute the exact solution by the approximate solution provided by the iterative algorithm MCG described in Section 4.1.1.2. We shall set a tolerance tol_{n+1}, which provides an upper threshold for the error between the optimal and the approximate solution in the weighted ℓ_2-norm. In the following paragraph, we give a precise and implementable condition on the sequence $(\mathrm{tol}_n)_{n\in\mathbb{N}}$ of the tolerances that guarantees convergence of the modified IRLS presented as Algorithm 13 below.

In contrast to Algorithm 1, the value β in Step 3 is introduced to obtain flexibility in tuning the performance of the algorithm. While we prove in Theorem 4.4 convergence for any positive value of β, Theorem 4.4(iii) guarantees instance optimality only for $\beta < \left(\frac{1-\gamma}{1+\gamma}\frac{K+1-k}{N}\right)^{\frac{1}{p}}$ in the case that $\lim_{n\to\infty} \varepsilon^n \neq 0$. Nevertheless in practice, choices of β which do not necessarily fulfill this condition may work very well. Section 4.1.4 investigates good choices of β numerically. This relation for β also sheds light on the

Algorithm 13 Iteratively Re-weighted Least Squares combined with CG (CG-IRLS)

Set $w^0 := (1, \dots, 1)$, $\varepsilon^0 := 1$, $\beta \in (0, 1]$

1: **while** $\varepsilon^n \neq 0$ **do**

2: Compute $\tilde{x}^{n+1}$ by means of MCG s.t. $\|\hat{x}^{n+1} - \tilde{x}^{n+1}\|^2_{\ell_2(w^n)} \leq \mathrm{tol}_{n+1}$, where $\hat{x}^{n+1} := \operatorname*{arg\,min}_{x \in \mathcal{F}_\Phi(y)} J_p(x, w^n, \varepsilon^n) = \operatorname*{arg\,min}_{z \in \mathcal{F}_\Phi(y)} \|z\|_{\ell_2(w^n)}$. Use the last iterate $\theta^{n,i}$ corresponding to $\tilde{x}^n = T^*\theta^{n,i}$ from MCG of the previous IRLS iteration as initial vector $\theta^0 = \theta^{n+1,0}$ for the present run of MCG.

3: $\varepsilon^{n+1} := \min(\varepsilon^n, \beta r(\tilde{x}^{n+1})_{K+1})$

4: $w^{n+1} := \operatorname*{arg\,min}_{w>0} J_p(\tilde{x}^{n+1}, w, \varepsilon^{n+1})$, i.e.,
$w_j^{n+1} = [|\tilde{x}_j^{n+1}|^2 + (\varepsilon^{n+1})^2]^{-\frac{2-p}{2}}, \quad j = 1, \dots, N$

5: **end while**

role of the parameter K. Furthermore, we see in Theorem 4.4 that Φ has to satisfy the (K, γ_K)-NSP.

From now on, we fix the notation $\hat{x}^{n+1}$ for the exact solution in Step 2 of Algorithm 13, and $\tilde{x}^{n+1,i}$ for its approximate solution in the i-th iteration of Algorithm MCG. We have to make sure that $\|\hat{x}^{n+1} - \tilde{x}^{n+1,i}\|^2_{\ell_2(w^n)}$ is sufficiently small to fall below the given tolerance. To this end, we could use the bound on the error provided by (4.4), but this has the following two unpractical drawbacks:

1. The vector $\hat{x} = \hat{x}^{n+1}$ is not known a priori;
2. The computation of the condition number c_{TT^*} is possible, but it requires the computation of eigenvalues with additional computational cost, which we prefer to avoid.

Hence, we propose an alternative estimate of the error in order to guarantee $\|\hat{x}^{n+1} - \tilde{x}^{n+1}\|^2_{\ell_2(w^n)} \leq \mathrm{tol}_{n+1}$. We use the notation of Algorithm MCG, but add an additional upper index for the outer IRLS iteration, e.g., $\theta^{n+1,i}$ is the θ^i in the $n+1$-th IRLS iteration. After i steps of MCG, we have by means of (2.28) and the definition of D_n in (2.29) that

$$\|\hat{x}^{n+1} - \tilde{x}^{n+1,i}\|^2_{\ell_2(w^n)} = \|D_n\Phi^*(\Phi D_n \Phi^*)^{-1}y - D_n\Phi^*\theta^{n+1,i}\|^2_{\ell_2(w^n)}.$$

We use $\theta^{n+1,i} = (\Phi D_n \Phi^*)^{-1}(y - \rho^{n+1,i})$ from Step 5 of MCG to obtain

$$\begin{aligned}
\|\hat{x}^{n+1} - \tilde{x}^{n+1,i}\|^2_{\ell_2(w^n)} &= \|D_n^{\frac{1}{2}} \Phi^* (\Phi D_n \Phi^*)^{-1} \rho^{n+1,i}\|^2_{\ell_2} \\
&\leq \|D_n\| \|\Phi\|^2 \|(\Phi D_n \Phi^*)^{-1}\|^2 \|\rho^{n+1,i}\|^2_{\ell_2} \\
&= \frac{\max\limits_{1\leq \ell \leq N} \left(|\tilde{x}_\ell|^2 + (\varepsilon^n)^2\right)^{\frac{2-p}{2}} \|\Phi\|^2}{\lambda_{\min}\left(\Phi D_n \Phi^*\right)} \|\rho^{n+1,i}\|^2_{\ell_2} \\
&\leq \left(1 + \max_{1\leq \ell \leq N} \left(\frac{|\tilde{x}^n_\ell|}{\varepsilon^n}\right)^2\right)^{\frac{2-p}{2}} \frac{\|\Phi\|^2}{\sigma_{\min}(\Phi)} \|\rho^{n+1,i}\|^2_{\ell_2}.
\end{aligned}$$

The last inequality above results from $\lambda_{\min}\left(\Phi D_n \Phi^*\right) = \sigma^2_{\min}\left(\Phi D_n^{\frac{1}{2}}\right)$ and

$$\sigma_{\min}\left(\Phi D_n^{\frac{1}{2}}\right) \geq \sigma_{\min}(\Phi)\, \sigma_{\min}\left(D_n^{\frac{1}{2}}\right) \geq (\varepsilon^n)^{2-p} \sigma_{\min}(\Phi).$$

Since ε^n and $\tilde{x}^n$ are known from the previous iteration, and $\|\rho^{n+1,i}\|_{\ell_2}$ is explicitly calculated within the MCG algorithm, $\|\hat{x}^{n+1} - \tilde{x}^{n+1,i}\|^2_{\ell_2(w^n)} \leq \mathrm{tol}_{n+1}$ can be achieved by iterating until

$$\|\rho^{n+1,i}\|^2_{\ell_2} \leq \frac{\sigma_{\min}(\Phi)}{\left(1 + \max\limits_{1\leq \ell \leq N} \left(\frac{|\tilde{x}^n_\ell|}{\varepsilon^n}\right)^2\right)^{\frac{2-p}{2}} \|\Phi\|^2} \mathrm{tol}_{n+1}. \tag{4.5}$$

Consequently, we shall use the minimal $i \in \mathbb{N}$ such that the above inequality is valid and set $\tilde{x}^{n+1} := \tilde{x}^{n+1,i}$, which will be the standard notation for the approximate solution.

In inequality (4.5), the computation of $\sigma_{\min}(\Phi)$ and $\|\Phi\|$ is necessary. The computation of these constants might be demanding, but has to be performed only once before the algorithm starts. Furthermore, in practice it is sufficient to compute approximations of these values and therefore these operations are not critical for the computation time of the algorithm.

4.1.2.1 Convergence Results

After introducing Algorithm 13, we state below the two main results of this section. Theorem 4.4 shows the convergence of the algorithm to a limit point that obeys certain error guarantees with respect to the solution of (2.3). Below K denotes the index used in the ε-update rule, i.e., Step 3 of Algorithm 13.

Theorem 4.4
Let $0 < p \leq 1$. Assume K is such that Φ satisfies the (K, γ_K)-NSP (see Definition 2.5) with $\gamma_K < 1$. If tol_{n+1} in Algorithm 13 is chosen such that

$$\sqrt{\mathrm{tol}_{n+1}} \leq \sqrt{\left(\frac{c_n}{2}\right)^2 + \frac{2a_{n+1}}{p\bar{W}_{n+1}^2}} - \frac{c_n}{2}, \tag{4.6}$$

where

$$c_n := 2W_n \left(\|\tilde{x}^n\|_{\ell_2(w^{n-1})} + \sqrt{\mathrm{tol}_n} \right), \text{ with} \tag{4.7}$$

$$\bar{W}_n := \sqrt{\frac{\max_i |\tilde{x}_i^{n-1}|^{2-p} + (\varepsilon^{n-1})^{2-p}}{(\varepsilon^n)^{2-p}}}, \text{ and } W_n := \left\| D_n^{-\frac{1}{2}} D_{n-1}^{\frac{1}{2}} \right\|, \tag{4.8}$$

for a sequence $(a_n)_{n\in\mathbb{N}}$, which fulfills $a_n \geq 0$ for all $n \in \mathbb{N}$, and $\sum_{i=0}^{\infty} a_n < \infty$, then, for each $y \in \mathbb{R}^m$, Algorithm 13 produces a non-empty set of accumulation points $\mathcal{Z}_p(y)$. Define $\varepsilon := \lim_{n\to\infty} \varepsilon^n$, then the following holds:

(i) If $\varepsilon = 0$, then $\mathcal{Z}_p(y)$ consists of a single K-sparse vector $\bar{x}$, which is the unique ℓ_p-minimizer in $\mathcal{F}_\Phi(y)$. Moreover, we have for any $x \in \mathcal{F}_\Phi(y)$:

$$\|x - \bar{x}\|_{\ell_p}^p \leq c_1 \sigma_K(x)_{\ell_p}, \quad \text{with } c_1 := 2\frac{1+\gamma}{1-\gamma}. \tag{4.9}$$

(ii) If $\varepsilon > 0$, then for each $\bar{x} \in \mathcal{Z}_p(y) \neq \emptyset$, we have $\langle \bar{x}, \eta \rangle_{\hat{w}(\bar{x},\varepsilon,p)} = 0$ for all $\eta \in \mathcal{N}_\Phi$, where $\hat{w}(\bar{x}, \varepsilon, p) = \left[||\bar{x}_i|^2 + \varepsilon^2 \right|^{-\frac{2-p}{2}} \Big]_{i=1}^N$. Moreover, in the case of $p = 1$, $\bar{x}$ is the single element of $\mathcal{Z}_p(y)$ and $\bar{x} = x^{\varepsilon,1} := \operatorname{arg\,min}_{x\in\mathcal{F}_\Phi(y)} \sum_{j=1}^N |x_j^2 + \varepsilon^2|^{\frac{1}{2}}$ (compare (4.27)).

(iii) Denote by $\mathcal{X}_{\varepsilon,p}(y)$ the set of global minimizers of $f_{\varepsilon,p}(x) := \sum_{j=1}^N |x_j^2 + \varepsilon^2|^{\frac{p}{2}}$ on $\mathcal{F}_\Phi(y)$. If $\varepsilon > 0$ and $\bar{x} \in \mathcal{Z}_p(y) \cap \mathcal{X}_{\varepsilon,p}(y)$, then for each $x \in \mathcal{F}_\Phi(y)$ and any $\beta < \left(\frac{1-\gamma}{1+\gamma}\frac{K+1-k}{N}\right)^{\frac{1}{p}}$, we have

$$\|x - \bar{x}\|_{\ell_p}^p \leq c_2 \sigma_k(x)_{\ell_p}, \quad \text{with } c_2 := \frac{1+\gamma}{1-\gamma}\left(\frac{2 + \frac{N\beta^p}{K+1-k}}{1 - \frac{N\beta^p}{K+1-k}\frac{1+\gamma}{1-\gamma}}\right).$$

Remark 4.5
Notice that (4.6) is an implicit bound on tol_{n+1} since it depends on ε^{n+1}, which means that in practice this value has to be updated in the MCG loop of the algorithm.

To be precise, after the update of $\theta^{n+1,i+1}$ in Step 4 of Algorithm 12 we compute $\tilde{x}^{n+1,i+1} = B^*\theta^{n+1,i+1}$ in each iteration i of the MCG loop. If $\tilde{x}^{n+1,i+1}$ is K-sparse for some iteration i, then $\varepsilon^{n+1} = \varepsilon^{n+1,i+1} = \min\left\{\varepsilon^n, \beta r\left(\tilde{x}^{n+1,i+1}\right)_{K+1}\right\} = 0$ and $\mathrm{tol}_{n+1} = 0$ by (4.7) and (4.8). In this case, MCG and IRLS are stopped by definition. The usage of this implicit bound is not efficient in practice since the computation of $r(\tilde{x}^{n+1,i+1})_{K+1}$ requires a sorting of N elements in each iteration of the MCG loop. While the implicit rule is required for the convergence analysis of the algorithm, we demonstrate in Section 4.1.4.2 that an explicit rule is sufficient for convergence in practice, and more efficient in terms of computational time.

Knowing that the algorithm converges and leads to an adequate solution, one is also interested in how fast one approaches this solution. Theorem 4.6 states that a linear rate of convergence can be established in the case of $p = 1$. In the case of $0 < p < 1$ this rate is even asymptotically super-linear.

Theorem 4.6
Assume Φ satisfies the NSP of order K with constant γ such that $0 < \gamma < 1 - \frac{2}{K+2}$, and that $\mathcal{F}_\Phi(y)$ contains a k-sparse vector x^*. Define $\Lambda := \mathrm{supp}(x^*)$. Suppose that $k < K - \frac{2\gamma}{1-\gamma}$ and $0 < \nu < 1$ are such that

$$\begin{aligned} \mu &:= \frac{\gamma(1+\gamma)}{(1-\nu)^{p(2-p)}\left(\min\limits_{j\in\Lambda}|x_j^*|\right)^{p(1-p)}}\left(1+\frac{(N-k)\beta^p}{K+1-k}\right)^{2-p} < 1, \\ R^* &:= \left(\nu \min_{j\in\Lambda}|x_j^*|\right)^p, \\ \tilde{\mu}(R^*)^{1-p} &\leq 1, \end{aligned} \tag{4.10}$$

for some $\tilde{\mu}$ satisfying $\mu < \tilde{\mu} < 1$. Define the error

$$E_n := \|\tilde{x}^n - x^*\|_{\ell_p}^p.$$

Assume there exists n_0 such that

$$E_{n_0} \leq R^*.$$

If a_{n+1} and tol_{n+1} are chosen as in Theorem 4.4 with the additional bound

$$\mathrm{tol}_{n+1} \leq \left(\frac{(\tilde{\mu}-\mu)E_n^{2-p}}{(NC)^{\frac{2-p}{2}}}\right)^{\frac{2}{p}}, \tag{4.11}$$

then for all $n \geq n_0$, we have

$$E_{n+1} \leq \mu E_n^{2-p} + (NC)^{1-\frac{p}{2}}(\mathrm{tol}_{n+1})^{\frac{p}{2}}, \tag{4.12}$$

and

$$E_{n+1} \leq \tilde{\mu} E_n^{2-p}, \tag{4.13}$$

where $C := 3 \sum_{n=1}^{\infty} a_n + J_p\left(\tilde{x}^1, w^0, \varepsilon^0\right)$. *Consequently,* $\tilde{x}^n$ *converges linearly to* x^* *in the case of* $p = 1$. *The convergence is super-linear in the case of* $0 < p < 1$.

Remark 4.7
Note that the second bound in (4.11), which implies (4.13), is only of theoretical nature. Since the value of E_n is unknown it cannot be computed in an implementation. However, heuristic choices of tol_{n+1} may fulfill this bound. Thus, in practice one can only guarantee the "asymptotic" (super-)linear convergence (4.12).

In the remainder of this section we aim to prove both results by means of some technical lemmas which are reported in Section 4.1.2.2 and Section 4.1.2.3.

4.1.2.2 Preliminary Results Concerning the Functional $J_p(x, w, \varepsilon)$

One important issue in the investigation of the dynamics of Algorithm 13 is the relationship between the weighted norm of an iterate and the weighted norm of its predecessor. In the following lemma, we present some helpful estimates.

Lemma 4.8
Let $\hat{x}^n$, $\hat{x}^{n+1}$, $\tilde{x}^n$, $\tilde{x}^{n+1}$ *and the respective tolerances* tol_n *and* tol_{n+1} *as defined in Algorithm 13. Then the inequalities*

$$\left| \left\| \hat{x}^{n+1} \right\|_{\ell_2(w^n)} - \left\| \tilde{x}^{n+1} \right\|_{\ell_2(w^n)} \right| \leq \sqrt{\mathrm{tol}_{n+1}}, \text{ and} \tag{4.14}$$

$$\left\| \hat{x}^{n+1} \right\|_{\ell_2(w^n)} \leq W_n \left(\|\tilde{x}^n\|_{\ell_2(w^{n-1})} + \sqrt{\mathrm{tol}_n} \right), \tag{4.15}$$

hold for all $n \geq 1$, *where* $W_n := \left\| D_n^{-\frac{1}{2}} D_{n-1}^{\frac{1}{2}} \right\|$.

Proof. Inequality (4.14) is a direct consequence of the triangle inequality for norms and the property that $\left\|\hat{x}^{n+1} - \tilde{x}^{n+1}\right\|_{\ell_2(w^n)} \leq \sqrt{\mathrm{tol}_{n+1}}$ of Step 2 in Algorithm 13.
In order to prove inequality (4.15), we first notice that $\hat{x}^n, \hat{x}^{n+1} \in \mathcal{F}_\Phi(y)$. Using that $\hat{x}^{n+1}$ is the minimizer of $\|\cdot\|_{\ell_2(w^n)}$ on $\mathcal{F}_\Phi(y)$, we obtain

$$\begin{aligned}
\left\| \hat{x}^{n+1} \right\|_{\ell_2(w^n)} &\leq \|\hat{x}^n\|_{\ell_2(w^n)} = \left\| D_n^{-\frac{1}{2}} \hat{x}^n \right\|_{\ell_2} = \left\| D_n^{-\frac{1}{2}} D_{n-1}^{\frac{1}{2}} D_{n-1}^{-\frac{1}{2}} \hat{x}^n \right\|_{\ell_2} \\
&\leq \left\| D_n^{-\frac{1}{2}} D_{n-1}^{\frac{1}{2}} \right\| \left\| D_{n-1}^{-\frac{1}{2}} \hat{x}^n \right\|_{\ell_2} = W_n \|\hat{x}^n\|_{\ell_2(w^{n-1})} \\
&\leq W_n \left(\|\tilde{x}^n\|_{\ell_2(w^{n-1})} + \sqrt{\mathrm{tol}_n} \right),
\end{aligned}$$

where the last inequality is due to (4.14). □

The functional $J_p(x, w, \varepsilon)$ obeys the following monotonicity property.

Lemma 4.9
The inequalities

$$J_p\left(\tilde{x}^{n+1}, w^{n+1}, \varepsilon^{n+1}\right) \leq J_p\left(\tilde{x}^{n+1}, w^{n}, \varepsilon^{n+1}\right) \leq J_p\left(\tilde{x}^{n+1}, w^{n}, \varepsilon^{n}\right). \tag{4.16}$$

hold for all $n \geq 0$.

Proof. The first inequality follows from the minimization property of w^{n+1}. The second inequality follows from $\varepsilon^{n+1} \leq \varepsilon^n$. □

The following lemma describes how the difference of the functional, evaluated in the exact and the approximated solution can be controlled by a positive scalar a_{n+1} and an appropriately chosen tolerance tol_{n+1}.

Lemma 4.10
Let a_{n+1} *be a positive scalar,* $\tilde{x}^{n+1}$, w^{n+1}, *and* ε^{n+1} *as described in Algorithm 13, and* $\hat{x}^{n+1} = \underset{x \in \mathcal{F}_\Phi(y)}{\arg\min}\, J_p\left(x, w^n, \varepsilon^n\right)$ *. If we choose* tol_n *as in* (4.6)*, then*

$$\left|J_p\left(\hat{x}^{n+1}, w^{n+1}, \varepsilon^{n+1}\right) - J_p\left(\tilde{x}^{n+1}, w^{n+1}, \varepsilon^{n+1}\right)\right| \leq a_{n+1}, \tag{4.17}$$

$$\left|J_p\left(\hat{x}^{n+1}, w^{n}, \varepsilon^{n}\right) - J_p\left(\tilde{x}^{n+1}, w^{n}, \varepsilon^{n}\right)\right| \leq a_{n+1}, \text{ and} \tag{4.18}$$

$$J_p\left(\hat{x}^{n+1}, w^{n+1}, \varepsilon^{n+1}\right) \leq J_p\left(\hat{x}^{n+1}, w^{n}, \varepsilon^{n}\right) + 2a_{n+1}. \tag{4.19}$$

Proof. The core of this proof is to find a bound on the quotient of the weights from one iteration step to the next and then to use the bound of the difference between $\hat{x}^{n+1}$ and $\tilde{x}^{n+1}$ in the $\ell_2(w^n)$-norm by tol_{n+1}. Starting with the definition of W_{n+1} in Lemma 4.8, the quotient of two successive weights can be estimated by

$$\begin{aligned} W_{n+1} = \left\| D_{n+1}^{-\frac{1}{2}} D_n^{\frac{1}{2}} \right\| &= \sqrt{\max_{\ell=1,\dots,N} \frac{w_\ell^{n+1}}{w_\ell^n}} = \sqrt{\max_{\ell=1,\dots,N} \frac{\left(|\tilde{x}_\ell^n|^2 + (\varepsilon^n)^2\right)^{\frac{2-p}{2}}}{\left(|\tilde{x}_\ell^{n+1}|^2 + (\varepsilon^{n+1})^2\right)^{\frac{2-p}{2}}}} \\ &\leq \sqrt{\frac{\max_{\ell=1,\dots,N} |\tilde{x}_\ell^n|^{2-p} + (\varepsilon^n)^{2-p}}{(\varepsilon^{n+1})^{2-p}}} = \bar{W}_{n+1}, \end{aligned} \tag{4.20}$$

where $\bar{W}_{n+1}$ was defined in (4.8). By choosing tol_{n+1} as in (4.6), we obtain

$$\begin{aligned} &\left|J_p\left(\hat{x}^{n+1}, w^{n+1}, \varepsilon^{n+1}\right) - J_p\left(\tilde{x}^{n+1}, w^{n+1}, \varepsilon^{n+1}\right)\right| \\ &= \left|\frac{p}{2} \sum_{j=1}^{N} \left(|\hat{x}_j^{n+1}|^2 - |\tilde{x}_j^{n+1}|^2\right) w_j^{n+1}\right| \end{aligned}$$

$$
\begin{aligned}
&= \left| \frac{p}{2} \sum_{j=1}^{N} \left(|\hat{x}_j^{n+1}| - |\tilde{x}_j^{n+1}| \right) \left(|\hat{x}_j^{n+1}| + |\tilde{x}_j^{n+1}| \right) w_j^{n+1} \right| \\
&\leq \frac{p}{2} \left(\sum_{j=1}^{N} \left| \hat{x}_j^{n+1} - \tilde{x}_j^{n+1} \right|^2 w_j^{n+1} \right)^{\frac{1}{2}} \left(\sum_{j=1}^{N} \left| |\hat{x}_j^{n+1}| + |\tilde{x}_j^{n+1}| \right|^2 w_j^{n+1} \right)^{\frac{1}{2}} \\
&\leq \frac{p}{2} \max_{\ell=1,\dots,N} \frac{w_\ell^{n+1}}{w_\ell^n} \left(\sum_{j=1}^{N} \left| \hat{x}_j^{n+1} - \tilde{x}_j^{n+1} \right|^2 w_j^n \right)^{\frac{1}{2}} \left(\sum_{j=1}^{N} \left| |\hat{x}_j^{n+1}| + |\tilde{x}_j^{n+1}| \right|^2 w_j^n \right)^{\frac{1}{2}} \\
&\leq \frac{p}{2} \bar{W}_{n+1}^2 \left\| \hat{x}^{n+1} - \tilde{x}^{n+1} \right\|_{\ell_2(w^n)} \left\| |\hat{x}^{n+1}| + |\tilde{x}^{n+1}| \right\|_{\ell_2(w^n)} \\
&\leq \frac{p}{2} \bar{W}_{n+1}^2 \sqrt{\mathrm{tol}_{n+1}} \left(\left\| \hat{x}^{n+1} \right\|_{\ell_2(w^n)} + \left\| \tilde{x}^{n+1} \right\|_{\ell_2(w^n)} \right) \\
&\leq \frac{p}{2} \bar{W}_{n+1}^2 \sqrt{\mathrm{tol}_{n+1}} \left[2W_n \left(\|\tilde{x}^n\|_{\ell_2(w^{n-1})} + \sqrt{\mathrm{tol}_n} \right) + \sqrt{\mathrm{tol}_{n+1}} \right] \\
&\leq \frac{p}{2} \bar{W}_{n+1}^2 \sqrt{\mathrm{tol}_{n+1}} \left[c_n + \sqrt{\mathrm{tol}_{n+1}} \right] \leq a_{n+1},
\end{aligned}
$$

where we have used the Cauchy-Schwarz inequality in the first inequality, (4.14) and (4.15) in the fifth inequality, (4.20) in the third inequality, the definition of c_n in (4.7), and the Assumption (4.6) on tol_{n+1} in the last inequality.

Since $1 \leq \bar{W}_{n+1}$, we obtain (4.18) by

$$
\begin{aligned}
&\left| J_p\left(\tilde{x}^{n+1}, w^n, \varepsilon^n\right) - J_p\left(\hat{x}^{n+1}, w^n, \varepsilon^n\right) \right| \\
&= \left| \frac{p}{2} \sum_{j=1}^{N} \left(|\hat{x}_j^{n+1}|^2 - |\tilde{x}_j^{n+1}|^2 \right) w_j^n \right| \\
&\leq \frac{p}{2} \left(\sum_{j=1}^{N} \left| \hat{x}_j^{n+1} - \tilde{x}_j^{n+1} \right|^2 w_j^n \right)^{\frac{1}{2}} \left(\sum_{j=1}^{N} \left| |\hat{x}_j^{n+1}| + |\tilde{x}_j^{n+1}| \right|^2 w_j^n \right)^{\frac{1}{2}} \\
&\leq \frac{p}{2} \bar{W}_{n+1}^2 \left(\sum_{j=1}^{N} \left| \hat{x}_j^{n+1} - \tilde{x}_j^{n+1} \right|^2 w_j^n \right)^{\frac{1}{2}} \left(\sum_{j=1}^{N} \left| |\hat{x}_j^{n+1}| + |\tilde{x}_j^{n+1}| \right|^2 w_j^n \right)^{\frac{1}{2}} \\
&\leq \frac{p}{2} \bar{W}_{n+1}^2 \sqrt{\mathrm{tol}_{n+1}} \left[c_n + \sqrt{\mathrm{tol}_{n+1}} \right] \leq a_{n+1},
\end{aligned}
$$

with the same arguments as above. Lemma 4.9 yields

$$
\begin{aligned}
J_p\left(\hat{x}^{n+1}, w^{n+1}, \varepsilon^{n+1}\right) &\leq J_p\left(\tilde{x}^{n+1}, w^{n+1}, \varepsilon^{n+1}\right) + a_{n+1} \leq J_p\left(\tilde{x}^{n+1}, w^n, \varepsilon^{n+1}\right) + a_{n+1} \\
&\leq J_p\left(\tilde{x}^{n+1}, w^n, \varepsilon^n\right) + a_{n+1} \leq J_p\left(\hat{x}^{n+1}, w^n, \varepsilon^n\right) + 2a_{n+1},
\end{aligned}
$$

where the first inequality follows from (4.17), the second and third by (4.16), and the last by (4.18). □

In the above lemma, we showed that the error of the evaluations of the functional J_p on the approximate solution $\tilde{x}^n$ and the weighted ℓ_2-minimizer $\hat{x}^n$ can be bounded by choosing an appropriate tolerance in the algorithm. This result will be used to show that the difference between the iterates $\tilde{x}^{n+1}$ and $\tilde{x}^n$ becomes arbitrarily small for $n \to \infty$, as long as we choose the sequence $(a_n)_{n\in\mathbb{N}}$ summable. This will be the main result of this section. Before, we prove some further auxiliary statements concerning the functional $J_p(x, w, \varepsilon)$ and the iterates $\tilde{x}^n$ and w^n.

Lemma 4.11
Let $(a_n)_{n\in\mathbb{N}}$, $a_n \in \mathbb{R}_+$, be a summable sequence with $A := \sum_{n=1}^{\infty} a_n < \infty$, and define $C := 3A + J_p\left(\tilde{x}^1, w^0, \varepsilon^0\right)$ as in Theorem 4.6. For each $n \geq 1$ we have

$$J_p\left(\tilde{x}^{n+1}, w^{n+1}, \varepsilon^{n+1}\right) = \sum_{j=1}^{N} \left(|\tilde{x}_j^{n+1}|^2 + (\varepsilon^{n+1})^2\right)^{\frac{p}{2}}, \tag{4.21}$$

$$\|\tilde{x}^n\|_{\ell_p}^p \leq C, \tag{4.22}$$

$$w_j^n \geq C^{-\frac{2-p}{p}}, j = 1, \ldots, N, \text{ and} \tag{4.23}$$

$$\|x\|_{\ell_2} \leq C^{\frac{2-p}{2p}} \|x\|_{\ell_2(w^n)} \text{ for all } x \in \mathbb{R}^N. \tag{4.24}$$

Proof. Identity (4.21) follows by insertion of the definition of w^{n+1} in Step 4 of Algorithm 13.

By the minimizing property of $\hat{x}^{n+1}$ and the fact that $\hat{x}^n \in \mathcal{F}_\Phi(y)$, we have

$$J_p\left(\hat{x}^{n+1}, w^n, \varepsilon^n\right) \leq J_p\left(\hat{x}^n, w^n, \varepsilon^n\right),$$

and thus, together with (4.19), it follows that

$$J_p\left(\hat{x}^{n+1}, w^{n+1}, \varepsilon^{n+1}\right) \leq J_p\left(\hat{x}^{n+1}, w^n, \varepsilon^n\right) + 2a_{n+1} \leq J_p\left(\hat{x}^n, w^n, \varepsilon^n\right) + 2a_{n+1}.$$

Hence, the telescoping sum

$$\sum_{k=1}^{n} \left(J_p\left(\hat{x}^{k+1}, w^{k+1}, \varepsilon^{k+1}\right) - J_p\left(\hat{x}^k, w^k, \varepsilon^k\right)\right) \leq 2\sum_{k=1}^{n} a_{k+1}$$

leads to the estimate

$$J_p\left(\hat{x}^{n+1}, w^{n+1}, \varepsilon^{n+1}\right) \leq J_p\left(\hat{x}^1, w^1, \varepsilon^1\right) + 2A \leq J_p\left(\tilde{x}^1, w^0, \varepsilon^0\right) + 2A + a_1.$$

Inequality (4.22) then follows from (4.17) and

$$\begin{aligned}\left\|\tilde{x}^{n+1}\right\|_{\ell_p}^p &\leq \sum_{j=1}^{N}\left[|\tilde{x}_j^{n+1}|^2+(\varepsilon^{n+1})^2\right]^{\frac{p}{2}} = J_p\left(\tilde{x}^{n+1},w^{n+1},\varepsilon^{n+1}\right)\\ &\leq J_p\left(\hat{x}^{n+1},w^{n+1},\varepsilon^{n+1}\right)+a_{n+1}\leq C, \qquad \text{for all } n\geq 1.\end{aligned}$$

Consequently, the bound (4.23) follows from

$$(w_j^n)^{-\frac{p}{2-p}} \leq \frac{2-p}{p}(w_j^n)^{-\frac{p}{2-p}} \leq J_p\left(\tilde{x}^n,w^n,\varepsilon^n\right)\leq C.$$

Inequality (4.24) is a direct consequence of (4.23). □

Notice that (4.22) states the boundedness of the iterates. The lower bound (4.23) on the weights w^n will become useful in the proof of Lemma 4.12.

By using the estimates collected so far, we can adapt [48, Lemma 5.1] to our situation. First, we shall prove that the differences between the n-th $\ell_2(w^{n-1})$-minimizer and its successor become arbitrarily small.

Lemma 4.12
Given a summable sequence $(a_n)_{n\in\mathbb{N}}$, $a_n\in\mathbb{R}_+$, the sequence $(\hat{x}^n)_{n\in\mathbb{N}}$ satisfies

$$\sum_{n=1}^{\infty}\left\|\hat{x}^{n+1}-\hat{x}^n\right\|_{\ell_2}^2 \leq \frac{2}{p}C^{\frac{2}{p}},$$

where C is the constant of Lemma 4.11 and $\hat{x}^n = \underset{x\in\mathcal{F}_\Phi(y)}{\arg\min}\, J_p\left(x,w^{n-1},\varepsilon^{n-1}\right)$. As a consequence we have

$$\lim_{n\to\infty}\left\|\hat{x}^n-\hat{x}^{n+1}\right\|_{\ell_2}=0. \tag{4.25}$$

Proof. We have

$$\begin{aligned}&\frac{2}{p}\left[J_p\left(\hat{x}^n,w^n,\varepsilon^n\right)-J_p\left(\hat{x}^{n+1},w^{n+1},\varepsilon^{n+1}\right)+2a_{n+1}\right]\\ &\geq \frac{2}{p}\left[J_p\left(\hat{x}^n,w^n,\varepsilon^n\right)-J_p\left(\hat{x}^{n+1},w^n,\varepsilon^n\right)\right] = \left\langle\hat{x}^n,\hat{x}^n\right\rangle_{w^n}-\left\langle\hat{x}^{n+1},\hat{x}^{n+1}\right\rangle_{w^n}\\ &= \left\langle\hat{x}^n+\hat{x}^{n+1},\hat{x}^n-\hat{x}^{n+1}\right\rangle_{w^n} = \left\langle\hat{x}^n-\hat{x}^{n+1},\hat{x}^n-\hat{x}^{n+1}\right\rangle_{w^n} = \sum_{i=1}^{N}w_j^n|\hat{x}_j^n-\hat{x}_j^{n+1}|^2\\ &\geq C^{-\frac{2-p}{p}}\left\|\hat{x}^n-\hat{x}^{n+1}\right\|_{\ell_2}^2.\end{aligned}$$

Here we used the fact that $\hat{x}^n - \hat{x}^{n+1} \in \mathcal{N}_\Phi$ and therefore, $\langle \hat{x}^{n+1}, \hat{x}^n - \hat{x}^{n+1} \rangle = 0$ and in the last step we applied the bound (4.24). Summing these inequalities over $n \geq 1$, we arrive at

$$\begin{aligned}\sum_{n=1}^{N} \left\| \hat{x}^n - \hat{x}^{n+1} \right\|_{\ell_2}^2 &\leq C^{\frac{2-p}{p}} \sum_{n=1}^{N} \frac{2}{p} \left[J_p\left(\hat{x}^n, w^n, \varepsilon^n\right) - J_p\left(\hat{x}^{n+1}, w^{n+1}, \varepsilon^{n+1}\right) + 2a_{n+1} \right] \\ &\leq \frac{2}{p} C^{\frac{2-p}{p}} \left[J_p\left(\hat{x}^1, w^1, \varepsilon^1\right) + \sum_{n=1}^{N} 2a_{n+1} \right] \leq \frac{2}{p} C^{\frac{2}{p}}.\end{aligned}$$

Letting $N \to \infty$ yields the desired result. □

The following lemma will play a major role in our proof of convergence since it shows that not only (4.25) holds but that also the difference between successive iterates becomes arbitrarily small.

Lemma 4.13
Let $\tilde{x}^n$ be as described in Algorithm 13 and $(a_n)_{n\in\mathbb{N}}$ be a summable sequence. Then

$$\lim_{n\to\infty} \left\| \tilde{x}^n - \tilde{x}^{n+1} \right\|_{\ell_2} = 0.$$

Proof. By (4.24) of Lemma 4.11 and the condition (4.6) on tol_n, we have

$$\begin{aligned}\|\hat{x}^n - \tilde{x}^n\|_{\ell_2} &\leq C^{\frac{2-p}{2p}} \|\hat{x}^n - \tilde{x}^n\|_{\ell_2(w^{n-1})} \leq C^{\frac{2-p}{2p}} \sqrt{\mathrm{tol}_n} \\ &\leq C^{\frac{2-p}{2p}} \left(-\frac{c_n}{2} + \sqrt{\left(\frac{c_n}{2}\right)^2} + \sqrt{\frac{2a_n}{p\bar{W}_n^2}} \right) \leq C^{\frac{2-p}{2p}} \sqrt{\frac{2}{p}} \sqrt{a_n}\end{aligned}$$

since $\bar{W}_n \geq 1$ as defined in Lemma 4.10. Since $(a_n)_{n\in\mathbb{N}}$ is summable, we conclude that

$$\lim_{n\to\infty} \|\hat{x}^n - \tilde{x}^n\|_{\ell_2} = 0. \tag{4.26}$$

Together with Lemma 4.12 we can prove our statement:

$$\begin{aligned}&\lim_{n\to\infty} \left\| \tilde{x}^n - \tilde{x}^{n+1} \right\|_{\ell_2} = \lim_{n\to\infty} \left\| \tilde{x}^n - \hat{x}^n + \hat{x}^n - \hat{x}^{n+1} + \hat{x}^{n+1} - \tilde{x}^{n+1} \right\|_{\ell_2} \\ &\leq \lim_{n\to\infty} \|\tilde{x}^n - \hat{x}^n\|_{\ell_2} + \lim_{n\to\infty} \left\| \hat{x}^n - \hat{x}^{n+1} \right\|_{\ell_2} + \lim_{n\to\infty} \left\| \hat{x}^{n+1} - \tilde{x}^{n+1} \right\|_{\ell_2} = 0,\end{aligned}$$

where the first and last term vanish because of (4.26) and the other term due to (4.25). □

4.1.2.3 The Functional $f_{\varepsilon,p}(z)$

In this section, we introduce an auxiliary functional which is useful for the proof of convergence. From the monotonicity of ε_n, we know that $\varepsilon = \lim_{n\to\infty} \varepsilon_n$ exists and is nonnegative. We introduce the functional

$$f_{\varepsilon,p}(x) := \sum_{j=1}^{N} |x_j^2 + \varepsilon^2|^{\frac{p}{2}}.$$

Note that if we would know that $\tilde{x}^n$ converges to x, then in view of (4.21), $f_{\varepsilon,p}(x)$ would be the limit of $J_p(\tilde{x}^n, w^n, \varepsilon^n)$. When $\varepsilon > 0$, the Hessian is given by $H(f_{\varepsilon,p})(x) = \operatorname{diag}\left[p\frac{x_j^2(p-1)+\varepsilon^2}{|x_j^2+\varepsilon^2|^{\frac{4-p}{2}}}\right]_{i=1}^{N}$. Thus, in particular, $H(f_{\varepsilon,1})(x)$ is strictly positive definite, so that $f_{\varepsilon,1}$ is strictly convex and therefore has a unique minimizer

$$x^{\varepsilon,1} := \operatorname*{arg\,min}_{x\in\mathcal{F}_\Phi(y)} f_{\varepsilon,1}(x). \tag{4.27}$$

In the case of $0 < p < 1$, we denote by $\mathcal{X}_{\varepsilon,p}(y)$ the set of global minimizers of $f_{\varepsilon,p}$ on $\mathcal{F}_\Phi(y)$. For both cases, the minimizers are characterized by the following lemma.

Lemma 4.14
Let $\varepsilon > 0$ and $x \in \mathcal{F}_\Phi(y)$. If $x = x^{\varepsilon,1}$ or $x \in \mathcal{X}_{\varepsilon,p}(y)$, then $\langle x, \eta\rangle_{\hat{w}(x,\varepsilon,p)} = 0$ for all $\eta \in \mathcal{N}_\Phi$, where $\hat{w}(x,\varepsilon,p) = \left[\left||x_i|^2 + \varepsilon^2\right|^{-\frac{2-p}{2}}\right]_{i=1}^{N}$. In the case of $p = 1$ also the converse is true.

Proof. The proof is an adaptation of [48, Lemma 5.2, Section 7], and is presented here for the sake of completeness.
"⇒"(in the case $0 < p \leq 1$)
Let $x = x^{\varepsilon,1}$ or $x \in \mathcal{X}_{\varepsilon,p}(y)$, and $\eta \in \mathcal{N}_\Phi$ arbitrarily. Consider the function

$$G_{\varepsilon,p}(t) := f_{\varepsilon,p}(x + t\eta) - f_{\varepsilon,p}(x)$$

with its first derivative

$$G'_{\varepsilon,p}(t) = p\sum_{i=1}^{N} \frac{x_i\eta_i + t\eta_i^2}{\left[|x_i + t\eta_i|^2 + \varepsilon^2\right]^{\frac{2-p}{2}}}.$$

Now $G_{\varepsilon,p}(0) = 0$ and from the minimization property of $f_{\varepsilon,p}(x)$, $G_{\varepsilon,p}(t) \geq 0$. Therefore,

$$0 = G'_{\varepsilon,p}(0) = \sum_{i=1}^{N} \frac{x_i\eta_i}{\left[x_i^2 + \varepsilon^2\right]^{\frac{2-p}{2}}} = \langle x, \eta\rangle_{\hat{w}(x,\varepsilon,p)}.$$

"⇐"(only in the case $p = 1$)
Now let $x \in \mathcal{F}_\Phi(y)$ and $\langle x, \eta \rangle_{\hat{w}(x,\varepsilon,1)} = 0$ for all $\eta \in \mathcal{N}_\Phi$. We want to show that x is the minimizer of $f_{\varepsilon,1}$ in $\mathcal{F}_\Phi(y)$. Consider the convex univariate function $g(u) := [u^2 + \varepsilon^2]^{\frac{1}{2}}$. For any point u_0 we have from convexity that

$$[u^2 + \varepsilon^2]^{\frac{1}{2}} \geq [u_0^2 + \varepsilon^2]^{\frac{1}{2}} + [u_0^2 + \varepsilon^2]^{-\frac{1}{2}} u_0 (u - u_0)$$

because the right-hand-side is the linear function which is tangent to g at u_0. It follows that for every point $v \in \mathcal{F}_\Phi(y)$ we have

$$f_{\varepsilon,1}(v) \geq f_{\varepsilon,1}(x) + \sum_{i=1}^{N} [x_i^2 + \varepsilon^2]^{-\frac{1}{2}} x_i (v_i - x_i) = f_{\varepsilon,1}(x) + \langle x, v - x \rangle_{\hat{w}(x,\varepsilon,1)} = f_{\varepsilon,1}(x),$$

where we use the orthogonality condition and the fact that $(v - x) \in \mathcal{N}_\Phi$. Since v was chosen arbitrarily, $x = x^{\varepsilon,1}$ as claimed. □

4.1.2.4 Proof of Convergence

By the results of the previous section, we are able to prove the convergence of Algorithm 13. The proof is inspired by the ones of [48, Theorem 5.3, Theorem 7.7], see also [84, Chapter 15.3], which we adapted to our case.

Proof (Proof of Theorem 4.4). Since $0 \leq \varepsilon^{n+1} \leq \varepsilon^n$ the sequence $(\varepsilon^n)_{n\in\mathbb{N}}$ always converges to some $\varepsilon > 0$.

Case $\varepsilon = 0$: Following the first part of the proof of [48, Theorems 5.3 and 7.7], where the boundedness of the sequence $\tilde{x}^n$ and the definition of ε^n is used, we can show that there is a subsequence $(\tilde{x}^{m_j})_{j\in\mathbb{N}}$ of $(\tilde{x}^n)_{n\in\mathbb{N}}$ such that $\tilde{x}^{m_j} \to \bar{x} \in \mathcal{F}_\Phi(y)$ and $\bar{x}$ is the unique ℓ_p-minimizer. It remains to show that also $\tilde{x}^n \to \bar{x}$. To this end, we first notice that $\tilde{x}^{m_j} \to \bar{x}$ and $\varepsilon^{m_j} \to 0$ imply $J_p(\tilde{x}^{m_j}, w^{m_j}, \varepsilon^{m_j}) \to \|\bar{x}\|_{\ell_p}^p$. The convergence of $J_p(\tilde{x}^n, w^n, \varepsilon^n) \to \|\bar{x}\|_{\ell_p}^p$ is established by the following argument: For each $n \in \mathbb{N}$ there is exactly one $i = i(n)$ such that $m_i < n \leq m_{i+1}$. We use (4.19) and (4.17) to estimate the telescoping sum

$$\begin{aligned} &|J_p(\tilde{x}^n, w^n, \varepsilon^n) - J_p(\tilde{x}^{m_{i(n)}}, w^{m_{i(n)}}, \varepsilon^{m_{i(n)}})| \\ \leq &\sum_{k=m_i}^{n-1} \left| J_p\left(\tilde{x}^{k+1}, w^{k+1}, \varepsilon^{k+1}\right) - J_p\left(\tilde{x}^k, w^k, \varepsilon^k\right) \right| \leq 4 \sum_{k=m_{i(n)}}^{n-1} a_{k+1}. \end{aligned}$$

Since $\sum_{k=0}^{\infty} a_k < \infty$ this implies that

$$\lim_{n\to\infty} |J_p(\tilde{x}^n, w^n, \varepsilon^n) - J_p(\tilde{x}^{m_{i(n)}}, w^{m_{i(n)}}, \varepsilon^{m_{i(n)}})| = 0$$

so that

$$\lim_{n\to\infty} J_p\left(\tilde{x}^n, w^n, \varepsilon^n\right) = \|\bar{x}\|_{\ell_p}^p.$$

Moreover (4.21) implies

$$J_p\left(\tilde{x}^n, w^n, \varepsilon^n\right) - N(\varepsilon^n)^p \leq \|\tilde{x}^n\|_{\ell_p}^p \leq J_p\left(\tilde{x}^n, w^n, \varepsilon^n\right),$$

and thus, $\|\tilde{x}^n\|_{\ell_p}^p \to \|\bar{x}\|_{\ell_p}^p$. Finally we invoke Lemma 2.6 with $z' = \tilde{x}^n$ and $z = \bar{x}$ to obtain

$$\limsup_{n\to\infty} \|\tilde{x}^n - \bar{x}\|_{\ell_p}^p \leq \frac{1+\gamma}{1-\gamma}\left(\lim_{n\to\infty} \|\tilde{x}^n\|_{\ell_p}^p - \|\bar{x}\|_{\ell_p}^p\right) = 0,$$

which completes the proof of $\tilde{x}^n \to \bar{x}$ in this case. To see (4.9) and establish (i), invoke Lemma 2.12.

Case $\varepsilon > 0$: By Lemma 4.11, we know that $(\tilde{x}^n)_{n\in\mathbb{N}}$ is a bounded sequence and hence has accumulation points. Let $(\tilde{x}^{m_i})$ be any convergent subsequence of $(\tilde{x}^n)_{n\in\mathbb{N}}$ and let $\bar{x} \in \mathcal{Z}_p(y)$ its limit. By (4.26), we know that also $\bar{x} \in \mathcal{F}_\Phi(y)$. Following the proof of [48, Theorem 5.3 and Theorem 7.7], one shows that $\langle \bar{x}, \eta\rangle_{\hat{w}(\bar{x},\varepsilon,p)} = 0$ for all $\eta \in \mathcal{N}_\Phi$, where $\hat{w}(\bar{x}, \varepsilon, p)$ is defined as in Lemma 4.14. In the case of $p = 1$, Lemma 4.14 implies $\bar{x} = x^{\varepsilon,1}$. Hence, $x^{\varepsilon,1}$ is the unique accumulation point of $(\tilde{x}^n)_{n\in\mathbb{N}}$. This establishes (ii).

To prove (iii), assume that $\bar{x} \in \mathcal{Z}_p(y) \cap \mathcal{X}_{\varepsilon,p}(y)$, and follow the proof of [48, Theorem 5.3, and 7.7] to conclude. □

4.1.2.5 Proof of Rate of Convergence

The proof follows similar steps as in [48, Section 6]. We define the auxiliary sequences of error vectors $\tilde{\eta}^n := \tilde{x}^n - x^*$ and $\hat{\eta}^n := \hat{x}^n - x^*$.

Proof (Proof of Theorem 4.6). We apply the characterization (2.27) with $w = w^n$, $\hat{x} = \hat{x}^{n+1} = x^* + \hat{\eta}^{n+1}$, and $\eta = \hat{\eta}^{n+1} = \hat{x}^{n+1} - x^*$, which gives

$$\sum_{j=1}^{N} (x_j^* + \hat{\eta}_j^{n+1})\hat{\eta}_j^{n+1} w_j^n = 0.$$

Rearranging the terms and using the fact that x^* is supported on Λ, we obtain

$$\sum_{j=1}^{N} |\hat{\eta}_j^{n+1}|^2 w_j^n = -\sum_{j=1}^{N} x_j^* \hat{\eta}_j^{n+1} w_j^n = -\sum_{j\in\Lambda} \frac{x_j^*}{[|\tilde{x}_j^n|^2 + (\varepsilon^n)^2]^{\frac{2-p}{2}}} \hat{\eta}_j^{n+1}. \tag{4.28}$$

By assumption there exists n_0 such that $E_{n_0} \leq R^*$. We prove (4.12), and $E_n \leq R^* \Rightarrow E_{n+1} \leq R^*$ to obtain the validity for all $n \geq n_0$. Assuming $E_n \leq R^*$, we have for all $j \in \Lambda$,

$$|\tilde{\eta}_j^n| \leq \|\tilde{\eta}^n\|_{\ell_p} = \sqrt[p]{E_n} \leq \nu |x_j^*|,$$

and thus

$$|\tilde{x}_j^n| = |x_j^* + \tilde{\eta}_j^n| \geq |x_j^*| - |\tilde{\eta}_j^n| \geq |x_j^*| - \nu|x_j^*|,$$

so that

$$\frac{|x_j^*|}{[|\tilde{x}_j^n|^2 + (\varepsilon^n)^2]^{\frac{2-p}{2}}} \leq \frac{|x_j^*|}{|\tilde{x}_j^n|^{2-p}} \leq \frac{1}{(1-\nu)^{2-p}|x_j^*|^{1-p}}. \tag{4.29}$$

Hence, (4.28) combined with (4.29) and the NSP leads to

$$\begin{aligned}
\left(\sum_{j=1}^N |\hat{\eta}_j^{n+1}|^2 w_j^n\right)^p &\leq \left((1-\nu)^{2-p}\left(\min_{j\in\Lambda}|x_j^*|\right)^{1-p}\right)^{-p} \|\hat{\eta}_\Lambda^{n+1}\|_{\ell_1}^p \\
&\leq \left((1-\nu)^{(2-p)}\left(\min_{j\in\Lambda}|x_j^*|\right)^{(1-p)}\right)^{-p} \|\hat{\eta}_\Lambda^{n+1}\|_{\ell_p}^p \\
&\leq \frac{\gamma}{(1-\nu)^{p(2-p)}\left(\min_{j\in\Lambda}|x_j^*|\right)^{p(1-p)}} \|\hat{\eta}_{\Lambda^c}^{n+1}\|_{\ell_p}^p.
\end{aligned}$$

Combining [48, Proposition 7.4] with the above estimate yields

$$\begin{aligned}
\|\hat{\eta}_{\Lambda^c}^{n+1}\|_{\ell_p}^{2p} &= \left\|\left[\hat{\eta}_i^{n+1}(w_i^n)^{-\frac{1}{p}}\right]_{i\in\Lambda^c}\right\|_{\ell_p(w^n)}^{2p} \leq \left\|\hat{\eta}_{\Lambda^c}^{n+1}\right\|_{\ell_2(w^n)}^{2p} \left\|\left[(w_i^n)^{-\frac{1}{p}}\right]_{i\in\Lambda^c}\right\|_{\ell_{\left[\frac{2p}{2-p}\right]}(w^n)}^{2p} \\
&\leq \left(\sum_{j=1}^N |\hat{\eta}_j^{n+1}|^2 w_j^n\right)^p \left(\sum_{j\in\Lambda^c}\left[|\tilde{\eta}_j^n| + \varepsilon^n\right]^p\right)^{2-p} \\
&\leq \frac{\gamma}{(1-\nu)^{p(2-p)}\left(\min_{j\in\Lambda}|x_j^*|\right)^{p(1-p)}} \|\hat{\eta}_{\Lambda^c}^{n+1}\|_{\ell_p}^p \left(\|\tilde{\eta}^n\|_{\ell_p}^p + (N-k)\,(\varepsilon^n)^p\right)^{2-p}.
\end{aligned}$$

It follows that

$$\|\hat{\eta}_{\Lambda^c}^{n+1}\|_{\ell_p}^p \leq \frac{\gamma}{(1-\nu)^{p(2-p)}\left(\min_{j\in\Lambda}|x_j^*|\right)^{p(1-p)}} \left(\|\tilde{\eta}^n\|_{\ell_p}^p + (N-k)\,(\varepsilon^n)^p\right)^{2-p}.$$

Note that this is also valid if $\hat{\eta}_{\Lambda^c}^{n+1} = 0$ since then the left-hand side is zero and the right-hand side non-negative. We furthermore obtain

$$\begin{aligned}
\|\hat{\eta}^{n+1}\|_{\ell_p}^p &= \|\hat{\eta}_\Lambda^{n+1}\|_{\ell_p}^p + \|\hat{\eta}_{\Lambda^c}^{n+1}\|_{\ell_p}^p \leq (1+\gamma)\|\hat{\eta}_{\Lambda^c}^{n+1}\|_{\ell_p}^p \\
&\leq \frac{\gamma(1+\gamma)}{(1-\nu)^{p(2-p)}\left(\min_{j\in\Lambda}|x_j^*|\right)^{p(p-1)}} \left(\|\tilde{\eta}^n\|_{\ell_p}^p + (N-k)\,(\varepsilon^n)^p\right)^{2-p}.
\end{aligned} \tag{4.30}$$

In addition to this, we know by [48, Lemma 4.1, 7.5] that

$$(J-j)r(x)_J^p \leq \|x-x'\|_{\ell_p}^p + \sigma_j(x')_{\ell_p}.$$

for any $J > j$ and $x, x' \in \mathbb{R}^N$. Thus, we have by the definition of ε^n in Step 3 of Algorithm 13 that

$$\begin{aligned}(N-k)(\varepsilon^n)^p &\leq (N-k)\beta^p \left(r(\tilde{x}^n)_{K+1}\right)^p \leq \frac{(N-k)\beta^p}{K+1-k}(\|\tilde{x}^n - x^*\|_{\ell_p}^p + \sigma_k(x^*)_{\ell_p}) \\ &= \frac{(N-k)\beta^p}{K+1-k}\|\tilde{\eta}^n\|_{\ell_p}^p\end{aligned}$$

since by assumption $\sigma_k(x^*)_{\ell_p} = 0$. Together with (4.30) this yields

$$\begin{aligned}\|\hat{\eta}^{n+1}\|_{\ell_p}^p &\leq \frac{\gamma(1+\gamma)}{(1-\nu)^{p(2-p)}\left(\min\limits_{j\in\Lambda}|x_j^*|\right)^{p(1-p)}}\left(1+\frac{(N-k)\beta^p}{K+1-k}\right)^{2-p}\|\tilde{\eta}^n\|_{\ell_p}^{p(2-p)} \\ &\leq \mu E_n^{2-p}.\end{aligned}$$

Finally, we obtain (4.12) by

$$\begin{aligned}E_{n+1} = \left\|\tilde{\eta}^{n+1}\right\|_{\ell_p}^p &\leq \left\|\hat{\eta}^{n+1}\right\|_{\ell_p}^p + \left\|\tilde{x}^{n+1} - \hat{x}^{n+1}\right\|_{\ell_p}^p \leq \left\|\hat{\eta}^{n+1}\right\|_{\ell_p}^p + N^{1-\frac{p}{2}}\left\|\tilde{x}^{n+1} - \hat{x}^{n+1}\right\|_{\ell_2}^p \\ &\leq \left\|\hat{\eta}^{n+1}\right\|_{\ell_p}^p + (NC)^{1-\frac{p}{2}}\left\|\tilde{x}^{n+1} - \hat{x}^{n+1}\right\|_{\ell_2(w^n)}^p \leq \mu E_n^{2-p} + (NC)^{1-\frac{p}{2}}(\mathrm{tol}_{n+1})^{\frac{p}{2}},\end{aligned}$$

where we used the triangle inequality in the first inequality, (4.24) in the third inequality, and C is the constant from Lemma 4.11. Equation (4.13) then follows by condition (4.11). By means of (4.10), we obtain

$$E_{n+1} \leq \tilde{\mu}E_n^{2-p} \leq \tilde{\mu}\left(R^*\right)^{2-p} \leq R^*,$$

and therefore the linear convergence for $p = 1$, and the super-linear convergence for $p < 1$ as soon as $n \geq n_0$. □

4.1.3 Conjugate Gradient Accelerated IRLS Method for ℓ_p-norm Regularized Least Squares

Similarly to the previous section we propose the combination of Algorithm 2 with the CG method. CG is used to calculate an approximation of the solution of the linear system (2.43) in Step 3 of Algorithm 2. After including the CG method, the modified algorithm which we shall consider is Algorithm 14, where we use the definition of $J_{p,\lambda}$ from (2.40).

Algorithm 14 CG-IRLS-λ

1: Set $w^0 := (1, \dots, 1)$, $\varepsilon^0 := 1$, $\alpha \in (0, 1]$, $\phi \in (0, \frac{1}{4-p})$.
2: **while** $\varepsilon^n > 0$ **do**
3: Compute $\tilde{x}^{n+1}$ by means of CG, s.t. $||\tilde{x}^{n+1} - \hat{x}^{n+1}||_{\ell_2(w^n)} \leq \mathrm{tol}_{n+1}$, where $\hat{x}^{n+1} := \arg\min_x J_{p,\lambda}(x, w^n, \varepsilon^n)$. Use $\tilde{x}^n$ as the initial vector for CG.
4: $\varepsilon^{n+1} := \min\left\{\varepsilon^n, |J_{p,\lambda}(\tilde{x}^{n-1}, w^{n-1}, \varepsilon^{n-1}) - J_{p,\lambda}(\tilde{x}^n, w^n, \varepsilon^n)|^\phi + \alpha^{n+1}\right\}$
5: $w^{n+1} := \arg\min_{w>0} J_{p,\lambda}(\tilde{x}^{n+1}, w, \varepsilon^{n+1})$
6: **end while**

As already mentioned in the introductory Section 2.4.1.3, we use the ε-update rule proposed by Voronin and Daubechies in [190, 189] because it allows to show that the algorithm converges to a minimizer of (2.39) for $p = 1$ and to critical points of (2.39) for $p < 1$. However, we were not able to prove similar statements for the rule of Lai, Xu, and Yin. It only allows to show the convergence of the algorithm to a critical point of the smoothed functional (2.41).

Notice that $\tilde{x}$ always denotes the approximate solution of the minimization with respect to x in Step 3 of Algorithm 14 and $\hat{x}$ the corresponding exact solution. Thus $\hat{x}^{n+1}$ fulfills (2.43) but not $\tilde{x}^{n+1}$.

Theorem 4.1 provides a stopping condition for the CG method, but as in the previous section it is not practical for us since we do not dispose of the minimizer and the computation of the condition number is computationally expensive. Therefore, we provide an alternative stopping criterion to make sure that $\|\tilde{x}^{n+1} - \hat{x}^{n+1}\|_{\ell_2(w^n)} \leq \mathrm{tol}_{n+1}$ is fulfilled in Step 3 of Algorithm 14.

Let $\tilde{x}^{n+1,l}$ be the l-th iterate of the CG method and define

$$A_n := \Phi^*\Phi + \mathrm{diag}\left[\lambda p w_j^n\right]_{j=1}^N.$$

Notice that the matrix $\Phi^*\Phi$ is positive semi-definite and $\lambda p D_n^{-1} = \lambda p\, \mathrm{diag}\left[w_j^n\right]_{j=1}^N$ is positive definite. Therefore, A_n is positive definite and invertible, and furthermore

$$\lambda_{\min}(A_n) \geq \lambda_{\min}(\mathrm{diag}\left[\lambda p w_j^n\right]_{j=1}^N). \tag{4.31}$$

We obtain

$$\begin{aligned} \left\|\hat{x}^{n+1} - \tilde{x}^{n+1,l}\right\|_{\ell_2(w^n)} &\leq \left\|A_n^{-1}\left(\Phi^* y - A_n \tilde{x}^{n+1,l}\right)\right\|_{\ell_2(w^n)} \\ &\leq \left\|D_n^{-\frac{1}{2}}\right\| \left\|A_n^{-1}\right\| \left\|r^{n+1,l}\right\|_{\ell_2}, \end{aligned} \tag{4.32}$$

where $r^{n+1,l} := \Phi^* y - A_n \tilde{x}^{n+1,l}$ is the residual as it appears in Step 5 of Algorithm 11.

The first factor on the right-hand side of (4.32) can be estimated by

$$\left\|D_n^{-\frac{1}{2}}\right\| = \lambda_{\max}\left(D_n^{-\frac{1}{2}}\right) = \sqrt{\max_j w_j^n} = \sqrt{\max_j \left(\left(\tilde{x}_j^n\right)^2 + (\varepsilon^n)^2\right)^{-\frac{2-p}{2}}} \leq (\varepsilon^n)^{-\frac{2-p}{2}}.$$

The second factor of (4.32) is estimated by

$$\begin{aligned}\left\|A_n^{-1}\right\| = (\lambda_{\min}(A_n))^{-1} &\leq \left(\lambda_{\min}(\operatorname{diag}\left[\lambda p w_j^n\right]_{j=1}^N)\right)^{-1} \\ &= \left(\lambda p\left(\left(\max_j |\tilde{x}_j^n|\right)^2 + (\varepsilon^n)^2\right)^{-\frac{2-p}{2}}\right)^{-1},\end{aligned}$$

where we used (4.31) in the inequality. Thus, we obtain

$$\left\|\tilde{x}^{n+1} - \hat{x}^{n+1,l}\right\|_{\ell_2(w^n)} \leq \frac{\left(\left(\max_j |\tilde{x}_j^n|\right)^2 + (\varepsilon^n)^2\right)^{\frac{2-p}{2}}}{(\varepsilon^n)^{\frac{2-p}{2}}\,\lambda p}\left\|r^{n+1,l}\right\|_{\ell_2},$$

and the suitable stopping condition

$$\left\|r^{n+1,l}\right\|_{\ell_2} \leq \frac{(\varepsilon^n)^{\frac{2-p}{2}}\,\lambda p}{\left(\left(\max_j |\tilde{x}_j^n|\right)^2 + (\varepsilon^n)^2\right)^{\frac{2-p}{2}}}\,\mathrm{tol}_{n+1}. \tag{4.33}$$

In the remainder of this section, we clarify how to choose the tolerance tol_{n+1}, and establish a convergence result of the algorithm. In the case of $p = 1$, the problem (2.39) is the minimization of the regularized least squares functional (2.12), and the optimality conditions can be stated in terms of subdifferential inclusions (compare Section (2.2.2)). We are able to show that at least a subsequence of the algorithm is converging to a solution of (2.39). If $0 < p < 1$, the problem is non-convex and non-smooth. Necessary first order optimality conditions for a global minimizer of this functional were derived in [23, Proposition 3.14], and [114, Theorem 2.2]. In our case, we are able to show that the non-zero components of the limits of the algorithm fulfill the respective conditions. However, as soon as the algorithm is producing zeros in some components of the limit, so far, we were not able to verify the conditions mentioned above. On this account, we pursue a different strategy, which originates from [195]. We do not directly show that the algorithm computes a solution of problem (2.39). Instead we show that a subsequence of the algorithm is at least computing a point $x^\dagger$, whose transformation $\breve{x}^\dagger = \mathcal{N}_{v/p}^{-1}(x^\dagger)$ is a critical point of the functional

$$\breve{F}_{v,\lambda}(x) := \|x\|_{\ell_v}^v + \frac{1}{2\lambda}\left\|\Phi\mathcal{N}_{v/p}(x) - y\right\|_{\ell_2}^2, \tag{4.34}$$

where

$$\mathcal{N}_\zeta \colon \mathbb{R}^N \to \mathbb{R}^N, \quad \left(\mathcal{N}_\zeta(x)\right)_j := \operatorname{sign}(x_j)|x_j|^\zeta, \quad j = 1, \ldots, N, \tag{4.35}$$

is a continuous bijective mapping and $1 < \upsilon \leq 2$. It was shown in [195, 158] that assuming $\breve{x}^\dagger$ is a global minimizer of $\breve{F}_{\upsilon,\lambda}(x)$ implies that $x^\dagger$ is a global minimizer of $F_{p,\lambda}$, i.e., a solution of problem (2.39). Furthermore, it was also shown that this result can be partially extended to local minimizers. We comment on this issue in Remark 4.18. These considerations allow us to state the main convergence result.

Theorem 4.15

Let $0 < p \leq 1$, $\lambda > 0$, $\Phi \in \mathbb{R}^{m\times N}$, and $y \in \mathbb{R}^m$. Define the sequences $(\tilde{x}^n)_{n\in\mathbb{N}}$, $(\varepsilon^n)_{n\in\mathbb{N}}$ and $(w^n)_{n\in\mathbb{N}}$ as the ones generated by Algorithm 14. Choose the accuracy tol_n of the CG-method, such that

$$\begin{aligned} \mathrm{tol}_n \quad \leq \quad \min\Bigg\{ & a_n \left(\sqrt{2\bar{J}p} C_{w^{n-1}} + 2\sqrt{\frac{2\bar{J}}{\lambda}} \sqrt{\left(\frac{2-p}{p\bar{J}}\right)^{-\frac{2-p}{p}} ||\Phi||} \right)^{-1}, \\ & \sqrt{a_n} \left(\frac{p}{2} + \frac{||\Phi||^2}{2\lambda} \left(\frac{2-p}{p\bar{J}}\right)^{-\frac{2-p}{p}} \right)^{-\frac{1}{2}} \Bigg\}, \end{aligned} \tag{4.36}$$

$$\text{with } C_{w^{n-1}} \quad := \quad \left(\frac{\max_j (\tilde{x}_j^{n-1})^2 + (\varepsilon^{n-1})^2}{(\varepsilon^n)^2} \right)^{1-\frac{p}{2}}, \tag{4.37}$$

where $(a_n)_{n\in\mathbb{N}}$ is a positive sequence satisfying $\sum_{n=0}^{\infty} a_n < \infty$ and $\bar{J} := J_{p,\lambda}(\tilde{x}^1, w^0, \varepsilon^0)$.

Then the sequence $(\tilde{x}^n)_{n\in\mathbb{N}}$ has at least one convergent subsequence $(\tilde{x}^{n_k})_{n_k\in\mathbb{N}}$. In the case that $p = 1$ and $x^\lambda \neq 0$, any convergent subsequence is such that its limit x^λ is a minimizer of $F_{1,\lambda}(x)$. In the case that $0 < p < 1$, the subsequence $(\tilde{x}^{n_k})_{n_k\in\mathbb{N}}$ can be chosen such that the transformation of its limit $\breve{x}^\lambda := \mathcal{N}_{\upsilon/p}^{-1}(x^\lambda)$, $1 < \upsilon \leq 2$, as defined in (4.35), is a critical point of (4.34). If $\breve{x}^\lambda$ is a global minimizer of (4.34), then x^λ is also a global minimizer of $F_{p,\lambda}(x)$.

Remark 4.16

Note that the bound (4.36) on tol_n is—in contrast to the one in Theorem 4.4—not implicit. Although tol_n depends on ε^n, the latter only depends on $\tilde{x}^{n-1}$, ε^{n-1}, w^{n-1}, and $\tilde{x}^{n-2}$, ε^{n-2}, w^{n-2}. Since in particular ε^n does not depend on $\tilde{x}^n$, we are able to exchange the Steps 3 and 4 in Algorithm 14.

As we argued in Remark 4.5, a possible relaxation of the tolerance bound (4.6) is allowed to further boost the convergence, the same applies to the bound (4.36).

Remark 4.17

In the case $0 < p < 1$, the theorem includes the possibility that there may exist several converging subsequences with different limits. Potentially only one of these

limits may have the nice property that its transformation is a critical point. In the proof of the theorem, which follows further below, an appropriate subsequence is constructed. Actually this construction leads to the following hint, how to practically choose the subsequence: Take a converging subsequence x_{n_l} for which the n_l satisfy equation (4.57).

It will be important below that a minimizer $x^\sharp$ of $F_{1,\lambda}(x)$ is characterized by the conditions

$$-(\Phi^*(y-\Phi x^\sharp))_j = \lambda \operatorname{sign}(x_j^\sharp) \text{ if } x_j^\sharp \neq 0, \tag{4.38}$$

$$|(\Phi^*(y-\Phi x^\sharp))_j| \leq \lambda \text{ if } x_j^\sharp = 0, \tag{4.39}$$

which have been derived in Section 2.2.2. Note that in the (less important) case $x^\lambda = 0$, the theorem does not give a conclusion about x^λ being a minimizer of $F_{1,\lambda}(x)$.

Remark 4.18
The result of Theorem 4.15 for $0 < p < 1$ can be partially extended towards local minimizers. For the sake of completeness we sketch the argument from [158]. Assume that $\breve{x}^\lambda$ is a local minimizer. Then there is a neighborhood $U_\epsilon(\breve{x}^\lambda)$ with $\epsilon > 0$ such that for all $x' \in U_\epsilon(\breve{x}^\lambda)$:

$$\breve{F}_{\upsilon,\lambda}(x') \geq \breve{F}_{\upsilon,\lambda}(\breve{x}^\lambda).$$

By continuity of $\mathcal{N}_{\upsilon/p}$ there exists an $\hat{\epsilon} > 0$ such that the neighborhood $U_{\hat{\epsilon}}(x^\lambda) \subset \mathcal{N}_{\upsilon/p}(U_\epsilon(\breve{x}^\lambda))$. Thus, for all $x \in U_{\hat{\epsilon}}(x^\lambda)$, we have $x' = \mathcal{N}_{\upsilon/p}^{-1}(x) \in U_\epsilon(\breve{x}^\lambda)$, and obtain

$$\begin{aligned}
F_{p,\lambda}(x) &= ||x||_{\ell_p}^p + \frac{1}{2\lambda}||\Phi x - y||_{\ell_2}^2 = ||\mathcal{N}_{\upsilon/p}(x')||_{\ell_p}^p + \frac{1}{2\lambda}||\Phi\mathcal{N}_{\upsilon/p}(x') - y||_{\ell_2}^2 \\
&= ||x'||_{\ell_\upsilon}^\upsilon + \frac{1}{2\lambda}||\Phi\mathcal{N}_{\upsilon/p}(x') - y||_{\ell_2}^2 = \breve{F}_{\upsilon,\lambda}(x') \\
&\geq \breve{F}_{\upsilon,\lambda}(\breve{x}^\lambda) = ||\breve{x}^\lambda||_{\ell_\upsilon}^\upsilon + \frac{1}{2\lambda}||\Phi\mathcal{N}_{\upsilon/p}(\breve{x}^\lambda) - y||_{\ell_2}^2 \\
&= ||x^\lambda||_{\ell_p}^p + \frac{1}{2\lambda}||\Phi x^\lambda - y||_{\ell_2}^2 = F_{p,\lambda}(x^\lambda).
\end{aligned}$$

For the proof of Theorem 4.15, we proceed similarly to Section 4.1.2, by first presenting a sequence of auxiliary lemmas on properties of the functional $J_{p,\lambda}$ and the dynamics of Algorithm 14.

4.1.3.1 Properties of the Functional $J_{p,\lambda}$

Lemma 4.19
For the functional $J_{p,\lambda}$ defined in (2.40), and the iterates $\tilde{x}^n$, w^n, and ε^n produced by Algorithm 14, the following inequalities hold true:

$$\begin{aligned} J_{p,\lambda}(\tilde{x}^{n+1}, w^{n+1}, \varepsilon^{n+1}) &\leq J_{p,\lambda}(\tilde{x}^{n+1}, w^n, \varepsilon^{n+1}) && (4.40)\\ &\leq J_{p,\lambda}(\tilde{x}^{n+1}, w^n, \varepsilon^n) && (4.41)\\ &\leq J_{p,\lambda}(\tilde{x}^n, w^n, \varepsilon^n). && (4.42) \end{aligned}$$

Proof. The first inequality holds because w^{n+1} is the minimizer and the second inequality holds since $\varepsilon^{n+1} \leq \varepsilon^n$. In the third inequality we use the fact that the CG-method is a descent method, decreasing the functional in each iteration. Since we take $\tilde{x}^n$ as the initial estimate in the first iteration of CG, the output $\tilde{x}^{n+1}$ of CG must have a value of the functional that is less or equal to the one of the initial estimate.□

The iterative application of Lemma 4.19 leads to the fact that for each $n \in \mathbb{N}^+$ the functional $J_{p,\lambda}$ is bounded:

$$0 \leq J_{p,\lambda}(\tilde{x}^n, w^n, \varepsilon^n) \leq J_{p,\lambda}(\tilde{x}^1, w^0, \varepsilon^0) = \bar{J}. \tag{4.43}$$

Since the functional is composed of positive summands, its definition and (4.43) imply

$$\begin{aligned} ||\Phi\tilde{x}^n - y||_{\ell_2} &\leq \sqrt{2\lambda\bar{J}},\\ ||\tilde{x}^n||_{\ell_2(w^n)} = \sqrt{\sum_{j=1}^{N} \left(\tilde{x}_j^n\right)^2 w_j^n} &\leq \sqrt{\frac{2\bar{J}}{p}}, \quad \text{and} \\ w_j^n &\geq \left(\frac{2-p}{p\bar{J}}\right)^{\frac{2-p}{p}}, \quad j = 1, \ldots, N. \end{aligned} \tag{4.44}$$

The last inequality leads to a general relationship between the ℓ_2-norm and $\ell_2(w^n)$-norm for arbitrary $x \in \mathbb{R}^N$:

$$||x||_{\ell_2(w^n)} \geq \sqrt{\left(\frac{2-p}{p\bar{J}}\right)^{\frac{2-p}{p}}} ||x||_{\ell_2}. \tag{4.45}$$

In order to show convergence to a critical point or minimizer of the functional $F_{p,\lambda}$, we will use the first order condition (2.42). Since this property is only valid for the exact solution $\hat{x}^{n+1}$, we need a connection between $\hat{x}^{n+1}$ and $\tilde{x}^{n+1}$. Observe that

$$J_{p,\lambda}(\hat{x}^{n+1}, w^n, \varepsilon^n) \leq J_{p,\lambda}(\tilde{x}^{n+1}, w^n, \varepsilon^n) \tag{4.46}$$

since $\hat{x}^{n+1}$ is the exact minimizer. From (4.46) we obtain

$$\frac{p}{2}\sum_{j=1}^{N}\left(\hat{x}_j^{n+1}\right)^2 w_j^n + \frac{1}{2\lambda}||\Phi\hat{x}^{n+1}-y||_{\ell_2}^2 \leq \frac{p}{2}\sum_{j=1}^{N}\left(\tilde{x}_j^{n+1}\right)^2 w_j^n + \frac{1}{2\lambda}||\Phi\tilde{x}^{n+1}-y||_{\ell_2}^2$$

which leads to

$$\frac{p}{2}||\hat{x}^{n+1}||_{\ell_2(w^n)}^2 \leq \frac{p}{2}||\tilde{x}^{n+1}||_{\ell_2(w^n)}^2 + \frac{1}{2\lambda}\left(||\Phi\tilde{x}^{n+1}-y||_{\ell_2}^2 - ||\Phi\hat{x}^{n+1}-y||_{\ell_2}^2\right). \quad (4.47)$$

Since (4.46) holds in addition to (4.42) and (4.43), we conclude, also for the exact solution $\hat{x}^{n+1}$, the bound

$$||\Phi\hat{x}^n - y||_{\ell_2} \leq \sqrt{2\lambda J_{p,\lambda}(\hat{x}^n, w^{n-1}, \varepsilon^{n-1})} \leq \sqrt{2\lambda\bar{J}}, \quad (4.48)$$

for all $n \in \mathbb{N}$, and

$$||\hat{x}^{n+1}||_{\ell_2(w^n)} \leq \sqrt{\frac{2J_{p,\lambda}(\hat{x}^{n+1}, w^n, \varepsilon^n)}{p}} \leq \sqrt{\frac{2\bar{J}}{p}}. \quad (4.49)$$

Additionally using (4.48), we are able to estimate the second summand of (4.47) by

$$\begin{aligned}
&\left(||\Phi\tilde{x}^{n+1}-y||_{\ell_2}^2 - ||\Phi\hat{x}^{n+1}-y||_{\ell_2}^2\right)\\
&\leq \left|\left(||\Phi\tilde{x}^{n+1}-y||_{\ell_2}^2 - ||\Phi\hat{x}^{n+1}-y||_{\ell_2}^2\right)\right|\\
&= \left|||\Phi\tilde{x}^{n+1}-\Phi\hat{x}^{n+1}||_{\ell_2}^2 + 2\left\langle\Phi\tilde{x}^{n+1}-\Phi\hat{x}^{n+1}, \Phi\hat{x}^{n+1}-y\right\rangle_{\ell_2}\right|\\
&\leq ||\Phi\tilde{x}^{n+1}-\Phi\hat{x}^{n+1}||_{\ell_2}\left(||\Phi\tilde{x}^{n+1}-\Phi\hat{x}^{n+1}||_{\ell_2} + 2||\Phi\hat{x}^{n+1}-y||_{\ell_2}\right)\\
&\leq ||\Phi\tilde{x}^{n+1}-\Phi\hat{x}^{n+1}||_{\ell_2}\left(||\Phi\tilde{x}^{n+1}-y||_{\ell_2} + 3||\Phi\hat{x}^{n+1}-y||_{\ell_2}\right)\\
&\leq 4\sqrt{2\lambda\bar{J}}||\Phi||\,||\tilde{x}^{n+1}-\hat{x}^{n+1}||_{\ell_2},
\end{aligned} \quad (4.50)$$

where we used the Cauchy-Schwarz inequality in the second inequality, the triangle inequality in the third inequality, and the bounds in (4.44) and (4.48) in the last inequality.

The following pivotal result of this section allows us to control the difference between the exact and approximate solution of the linear system in Step 3 of Algorithm 14.

Lemma 4.20
For a given positive number a_{n+1} and a choice of the accuracy tol_{n+1} satisfying (4.36), the functional $J_{p,\lambda}$ fulfills the two monotonicity properties

$$J_{p,\lambda}(\hat{x}^{n+1}, w^{n+1}, \varepsilon^{n+1}) - J_{p,\lambda}(\tilde{x}^{n+1}, w^{n+1}, \varepsilon^{n+1}) \leq a_{n+1} \quad (4.51)$$

and

$$J_{p,\lambda}(\tilde{x}^{n+1}, w^n, \varepsilon^n) - J_{p,\lambda}(\hat{x}^{n+1}, w^n, \varepsilon^n) \leq a_{n+1}. \quad (4.52)$$

Proof. By means of the relation

$$w_j^{n+1} = w_j^n \frac{w_j^{n+1}}{w_j^n} \leq w_j^n \left(\frac{(\tilde{x}_j^n)^2 + (\varepsilon^n)^2}{(\tilde{x}_j^{n+1})^2 + (\varepsilon^{n+1})^2} \right)^{1-\frac{p}{2}}$$
$$\leq w_j^n \left(\frac{\max_j (\tilde{x}_j^n)^2 + (\varepsilon^n)^2}{(\varepsilon^{n+1})^2} \right)^{1-\frac{p}{2}} = w_j^n C_{w^n},$$

where C_{w^n} was defined in (4.37), we can estimate

$$\begin{aligned}
&J_{p,\lambda}(\hat{x}^{n+1}, w^{n+1}, \varepsilon^{n+1}) - J_{p,\lambda}(\tilde{x}^{n+1}, w^{n+1}, \varepsilon^{n+1}) \\
&\leq \frac{p}{2} \sum_{j=1}^{N} \left(\hat{x}_j^{n+1} - \tilde{x}_j^{n+1} \right) \left(\hat{x}_j^{n+1} + \tilde{x}_j^{n+1} \right) w_j^{n+1} + \left| \frac{1}{2\lambda} ||\Phi \hat{x}^{n+1} - y||_{\ell_2}^2 - ||\Phi \tilde{x}^{n+1} - y||_{\ell_2}^2 \right| \\
&\leq \frac{p}{2} \left| \left\langle \hat{x}^{n+1} - \tilde{x}^{n+1}, \hat{x}^{n+1} + \tilde{x}^{n+1} \right\rangle_{\ell_2(w^{n+1})} \right| + \frac{4\sqrt{2\lambda \bar{\bar{J}}}}{2\lambda} ||\Phi|| ||\tilde{x}^{n+1} - \hat{x}^{n+1}||_{\ell_2} \\
&\leq \frac{p}{2} \sqrt{\sum_{j=1}^{N} (\hat{x}_j^{n+1} - \tilde{x}_j^{n+1})^2 w_j^{n+1}} \sqrt{\sum_{j=1}^{N} (\hat{x}_j^{n+1} + \tilde{x}_j^{n+1})^2 w_j^{n+1}} \\
&\qquad + \frac{4\sqrt{2\lambda \bar{\bar{J}}}}{2\lambda} ||\Phi|| ||\tilde{x}^{n+1} - \hat{x}^{n+1}||_{\ell_2} \\
&\leq \frac{p}{2} C_{w^n} ||\hat{x}^{n+1} - \tilde{x}^{n+1}||_{\ell_2(w^n)} ||\hat{x}^{n+1} + \tilde{x}^{n+1}||_{\ell_2(w^n)} + \frac{4\sqrt{2\lambda \bar{\bar{J}}}}{2\lambda} ||\Phi|| ||\tilde{x}^{n+1} - \hat{x}^{n+1}||_{\ell_2} \\
&\leq C_{w^n} ||\hat{x}^{n+1} - \tilde{x}^{n+1}||_{\ell_2(w^n)} 2 \max \left\{ \frac{p}{2} ||\hat{x}^{n+1}||_{\ell_2(w^n)}, \frac{p}{2} ||\tilde{x}^{n+1}||_{\ell_2(w^n)} \right\} \\
&\qquad + \frac{4\sqrt{2\lambda \bar{\bar{J}}}}{2\lambda} ||\Phi|| ||\tilde{x}^{n+1} - \hat{x}^{n+1}||_{\ell_2} \\
&\leq \sqrt{2\bar{J}p} C_{w^n} ||\hat{x}^{n+1} - \tilde{x}^{n+1}||_{\ell_2(w^n)} + \frac{4\sqrt{2\lambda \bar{\bar{J}}}}{2\lambda} \sqrt{\left(\frac{2-p}{p\bar{J}} \right)^{-\frac{2-p}{p}}} ||\Phi|| ||\tilde{x}^{n+1} - \hat{x}^{n+1}||_{\ell_2(w^n)} \\
&\leq \left(\sqrt{2\bar{J}p} C_{w^n} + \frac{4\sqrt{2\lambda \bar{\bar{J}}}}{2\lambda} \sqrt{\left(\frac{2-p}{p\bar{J}} \right)^{-\frac{2-p}{p}}} ||\Phi|| \right) ||\tilde{x}^{n+1} - \hat{x}^{n+1}||_{\ell_2(w^n)} \leq a_{n+1},
\end{aligned}$$

where we used (4.50) in the second inequality, Cauchy-Schwarz in the third inequality, and (4.45), (4.44), and (4.49) in the sixth inequality. Thus we obtain (4.51). To show (4.52), we use (4.45) in the second to last inequality, condition (4.36) in the last inequality and the fact that $\hat{x}^{n+1} = \arg\min_x J_{p,\lambda}(x, w^n, \varepsilon^n)$ (and thus fulfilling (2.42))

in the second identity below:

$$\begin{aligned}
&J_{p,\lambda}(\tilde{x}^{n+1}, w^n, \varepsilon^n) - J_{p,\lambda}(\hat{x}^{n+1}, w^n, \varepsilon^n) \\
&= \frac{p}{2}\sum_{j=1}^{N}\left((\tilde{x}_j^{n+1})^2 - (\hat{x}_j^{n+1})^2\right) w_j^n \\
&\qquad + \frac{1}{2\lambda}\left(||\Phi\tilde{x}^{n+1} - \Phi\hat{x}^{n+1}||_{\ell_2}^2 + 2\left\langle \Phi(\tilde{x}^{n+1} - \hat{x}^{n+1}), \Phi\hat{x}^{n+1} - y\right\rangle_{\ell_2}\right) \\
&= \frac{p}{2}\sum_{j=1}^{N}\left((\tilde{x}_j^{n+1})^2 - (\hat{x}_j^{n+1})^2 - 2\hat{x}_j^{n+1}\tilde{x}_j^{n+1} + 2\left(\hat{x}_j^{n+1}\right)^2\right) w_j^n \\
&\qquad + \frac{1}{2\lambda}||\Phi\tilde{x}^{n+1} - \Phi\hat{x}^{n+1}||_{\ell_2}^2 \\
&\leq \frac{p}{2}\sum_{j=1}^{N}\left((\tilde{x}_j^{n+1})^2 + (\hat{x}_j^{n+1})^2 - 2\hat{x}_j^{n+1}\tilde{x}_j^{n+1}\right) w_j^n + \frac{1}{2\lambda}||\Phi||||\tilde{x}^{n+1} - \hat{x}^{n+1}||_{\ell_2}^2 \\
&\leq \left(\frac{p}{2} + \frac{||\Phi||^2}{2\lambda}\left(\frac{2-p}{p\bar{J}}\right)^{-\frac{2-p}{p}}\right)||\tilde{x}^{n+1} - \hat{x}^{n+1}||_{\ell_2(w^n)}^2 \leq a_{n+1}.
\end{aligned} \tag{4.53}$$

□

Besides Lemma 4.20 there are two more helpful properties of the functional. First, the identity

$$J_{p,\lambda}(\hat{x}^n, w^n, \varepsilon^n) - J_{p,\lambda}(\hat{x}^{n+1}, w^n, \varepsilon^n) = \frac{p}{2}||\hat{x}^n - \hat{x}^{n+1}||_{\ell_2(w^n)}^2 + \frac{1}{2\lambda}||\Phi\hat{x}^n - \Phi\hat{x}^{n+1}||_{\ell_2}^2$$

can be shown by the same calculation as in (4.53), by means of replacing $\tilde{x}^{n+1}$ by $\hat{x}^n$. Second, it follows in particular that

$$\begin{aligned}
\frac{p}{2}\sqrt{\left(\frac{2-p}{p\bar{J}}\right)^{\frac{2-p}{p}}}||\hat{x}^{n+1} - \hat{x}^n||_{\ell_2}^2 &\leq \frac{p}{2}||\hat{x}^{n+1} - \hat{x}^n||_{\ell_2(w^n)}^2 \\
&\leq J_{p,\lambda}(\hat{x}^n, w^n, \varepsilon^n) - J_{p,\lambda}(\hat{x}^{n+1}, w^n, \varepsilon^n).
\end{aligned} \tag{4.54}$$

where the estimate (4.45) is used in the first inequality.

4.1.3.2 Proof of Convergence

We need to show that the difference $\hat{x}^{n+1} - \hat{x}^n$ between two successive *exact* iterates and the one between the exact and approximated iterates, $\hat{x}^n - \tilde{x}^n$, become arbitrarily small. This result is used in the proof of Theorem 4.15 to show that both $(\hat{x}^n)_{n\in\mathbb{N}}$ and $(\tilde{x}^n)_{n\in\mathbb{N}}$ converge to the same limit.

Lemma 4.21
Consider a summable sequence $(a_n)_{n\in\mathbb{N}}$ *and choose the accuracy of the CG solution* tol_n *satisfying* (4.36) *for the n-th iteration step. Then the sequences* $(\hat{x}^n)_{n\in\mathbb{N}}$ *and* $(\tilde{x}^n)_{n\in\mathbb{N}}$ *have the properties*

$$\lim_{n\to\infty} ||\hat{x}^n - \hat{x}^{n+1}||_{\ell_2} = 0 \tag{4.55}$$

and

$$\lim_{n\to\infty} ||\tilde{x}^{n+1} - \hat{x}^{n+1}||_{\ell_2} = 0. \tag{4.56}$$

Proof. We use the properties of J, which we derived in the previous subsection. First, we show (4.55):

$$\begin{aligned}
&\frac{p}{2}\sqrt{\left(\frac{2-p}{p\bar{J}}\right)^{\frac{2-p}{p}} \sum_{n=1}^{M} ||\hat{x}^{n+1} - \hat{x}^n||_{\ell_2}^2} \\
&\leq \sum_{n=1}^{M} J_{p,\lambda}(\hat{x}^n, w^n, \varepsilon^n) - J_{p,\lambda}(\hat{x}^{n+1}, w^n, \varepsilon^n) \\
&\leq \sum_{n=1}^{M} J_{p,\lambda}(\hat{x}^n, w^n, \varepsilon^n) - J_{p,\lambda}(\tilde{x}^{n+1}, w^n, \varepsilon^n) + a_{n+1} \\
&\leq \sum_{n=1}^{M} J_{p,\lambda}(\hat{x}^n, w^n, \varepsilon^n) - J_{p,\lambda}(\tilde{x}^{n+1}, w^{n+1}, \varepsilon^{n+1}) + a_{n+1} \\
&\leq \sum_{n=1}^{M} J_{p,\lambda}(\hat{x}^n, w^n, \varepsilon^n) - J_{p,\lambda}(\hat{x}^{n+1}, w^{n+1}, \varepsilon^{n+1}) + 2a_{n+1} \\
&= J_{p,\lambda}(\hat{x}^1, w^1, \varepsilon^1) - J_{p,\lambda}(\tilde{x}^{M+1}, w^{M+1}, \varepsilon^{M+1}) + 2\sum_{n=1}^{M} a_{n+1} \\
&\leq \bar{J} + 2\sum_{n=1}^{M} a_{n+1}.
\end{aligned}$$

We used (4.54) in the first inequality, (4.52) in the second inequality, (4.40) and (4.41) in the third inequality, (4.51) in the fourth inequality and a telescoping sum in the identity. Letting $M \to \infty$ we obtain

$$\frac{p}{2}\left(\frac{2-p}{p\bar{J}}\right)^{\frac{2-p}{p}} \sum_{n=1}^{\infty} ||\hat{x}^{n+1} - \hat{x}^n||_{\ell_2}^2 \leq \bar{J} + 2\sum_{n=1}^{\infty} a_{n+1} < \infty$$

and thus (4.55).

Second, we show (4.56). From line 1 and 3 of (4.53) we know that

$$\begin{aligned}
&J_{p,\lambda}(\tilde{x}^{n+1}, w^n, \varepsilon^n) - J_{p,\lambda}(\hat{x}^{n+1}, w^n, \varepsilon^n) \\
&= \frac{p}{2}\sum_{j=1}^{N}\left((\tilde{x}_j^{n+1})^2 - (\hat{x}_j^{n+1})^2 - 2\hat{x}_j^{n+1}\tilde{x}_j^{n+1} + 2\left(\hat{x}_j^{n+1}\right)^2\right) w_j^n \\
&\qquad + \frac{1}{2\lambda}||\Phi\tilde{x}^{n+1} - \Phi\hat{x}^{n+1}||_{\ell_2}^2 \\
&= \frac{p}{2}||\tilde{x}_j^{n+1} - \hat{x}_j^{n+1}||_{\ell_2(w^n)}^2 + \frac{1}{2\lambda}||\Phi\tilde{x}^{n+1} - \Phi\hat{x}^{n+1}||_{\ell_2}^2.
\end{aligned}$$

Since the second summand is positive, we conclude

$$J_{p,\lambda}(\tilde{x}^{n+1}, w^n, \varepsilon^n) - J_{p,\lambda}(\hat{x}^{n+1}, w^n, \varepsilon^n) \geq \frac{p}{2}||\tilde{x}_j^{n+1} - \hat{x}_j^{n+1}||_{\ell_2(w^n)}^2.$$

Together with (4.52) we find that

$$\begin{aligned}
\frac{p}{2}\left(\frac{2-p}{p\bar{J}}\right)^{\frac{2-p}{p}}||\tilde{x}^{n+1} - \hat{x}^{n+1}||_{\ell_2}^2 &\leq \frac{p}{2}||\tilde{x}^{n+1} - \hat{x}^{n+1}||_{\ell_2(w^n)}^2 \\
&\leq J_{p,\lambda}(\tilde{x}^{n+1}, w^n, \varepsilon^n) - J_{p,\lambda}(\hat{x}^{n+1}, w^n, \varepsilon^n) \leq a_{n+1},
\end{aligned}$$

and thus taking limits on both sides we get

$$\frac{p}{2}\left(\frac{2-p}{p\bar{J}}\right)^{\frac{2-p}{p}}\limsup_{n\to\infty}||\tilde{x}^{n+1} - \hat{x}^{n+1}||_{\ell_2}^2 \leq \lim_{n\to\infty} a_{n+1} = 0,$$

which implies (4.56). □

Remark 4.22
By means of Lemma 4.21 we obtain

$$\lim_{n\to\infty}||\tilde{x}^n - \tilde{x}^{n+1}||_{\ell_2} \leq \lim_{n\to\infty}||\tilde{x}^n - \hat{x}^n||_{\ell_2} + \lim_{n\to\infty}||\hat{x}^n - \hat{x}^{n+1}||_{\ell_2} + \lim_{n\to\infty}||\hat{x}^{n+1} - \tilde{x}^{n+1}||_{\ell_2} = 0.$$

The following lemma provides a lower bound for the ε^n, which is used to show a contradiction in the proof of Theorem 4.15. Recall that $\phi \in (0, \frac{1}{4-p})$ is the parameter appearing in the update rule for ε in Step 4 of both the Algorithms 14 and 2.

Lemma 4.23 ([189, Lemma 4.5.4, Lemma 4.5.6])
Let $p = 1$ and thus $w_j^n = ((\tilde{x}_j^n)^2 + (\varepsilon^n)^2)^{-1/2}$, $j \in \{1, \ldots, N\}$. There exists a strictly increasing subsequence $(n_l)_{l\in\mathbb{N}}$ and some constant $C > 0$ such that

$$(\varepsilon^{n_l+1})^2 \geq C((w_j^{n_l})^{-1})^{2p\phi}|(w_j^{n_l-1})^{-1} - (w_j^{n_l})^{-1}|^{4\phi}.$$

Proof. Since $J_{p,\lambda}(\tilde{x}^n, w^n, \varepsilon^n)$ is decreasing with n due to Lemma 4.19 and bounded below by 0, the difference $|J_{p,\lambda}(\tilde{x}^{n-1}, w^{n-1}, \varepsilon^{n-1}) - J_{p,\lambda}(\tilde{x}^n, w^n, \varepsilon^n)|$ is converging to 0 for $n \to \infty$. In addition $\alpha^{n+1} \to 0$ for $n \to \infty$, and thus by definition also $\varepsilon^n \to 0$. Consequently there exists a subsequence $(n_l)_{l\in\mathbb{N}}$ such that

$$\varepsilon^{n_l+1} = |J_{p,\lambda}(\tilde{x}^{n_l-1}, w^{n_l-1}, \varepsilon^{n_l-1}) - J_{p,\lambda}(\tilde{x}^{n_l}, w^{n_l}, \varepsilon^{n_l})|^{\phi} + \alpha^{n_l+1}. \tag{4.57}$$

Following exactly the steps of the proof of [189, Lemma 4.5.6.] yields the assertion. Observe that all of these steps are also valid for $0 < p < 1$, although in [189, Lemma 4.5.6] the author restricted it to the case $p \geq 1$. □

Remark 4.24
The observation in the previous proof that (ε^n) converges to 0 will be again important below.

We are now prepared for the proof of Theorem (4.15).

Proof (Proof of Theorem 4.15). Consider the subsequence $(\tilde{x}^{n_l})_{l\in\mathbb{N}}$ of Lemma 4.23. Since $\|\tilde{x}^{n_l}\|_{\ell_2}$ is bounded by (4.44), there exists a converging subsequence $(\tilde{x}^{n_k})_{k\in\mathbb{N}}$, which has limit x^{λ}.

Consider the case $p = 1$ and $x^{\lambda} \neq 0$. We first show that

$$-\infty < \lim_{n\to\infty} \tilde{x}_j^{n_k+1} w_j^{n_k} = \lim_{n\to\infty} \hat{x}_j^{n_k+1} w_j^{n_k} < \infty, \text{ for all } j = 1, \ldots, N. \tag{4.58}$$

It follows from equation (2.42) and the boundedness of the residual (4.48) that the sequence $(\hat{x}^{n_k+1} w_j^{n_k})_{n_k}$ is bounded, i.e.,

$$\left\| \left[\hat{x}_j^{n_k+1} w_j^{n_k} \right]_j \right\|_2 = \frac{1}{\lambda} \|\Phi^*(\Phi \hat{x}^{n_k+1} - y)\| \leq C.$$

Therefore, there exists a converging subsequence, for simplicity again denoted by $(\hat{x}^{n_k+1} w_j^{n_k})_{n_k}$. To show the identity in (4.58), we estimate

$$\begin{aligned}
|\hat{x}_j^{n_k+1} w_j^{n_k} - \tilde{x}_j^{n_k+1} w_j^{n_k}| &\leq \frac{\mathrm{tol}_{n_k+1}}{\sqrt{(\tilde{x}_j^{n_k})^2 + (\varepsilon^{n_k})^2}} \leq \frac{a_{n_k+1}}{\sqrt{2\bar{J}} C_{w^{n_k}} \sqrt{(\tilde{x}_j^{n_k})^2 + (\varepsilon^{n_k})^2}} \\
&= \frac{a_{n_k+1} \varepsilon^{n_k+1}}{\sqrt{2\bar{J}} \sqrt{\max_\ell (\tilde{x}_\ell^{n_k})^2 + (\varepsilon^{n_k})^2} \sqrt{(\tilde{x}_j^{n_k})^2 + (\varepsilon^{n_k})^2}} \\
&\leq \frac{a_{n_k+1} \varepsilon^{n_k+1}}{\sqrt{2\bar{J}} (\max_\ell |\tilde{x}_\ell^{n_k}|)(\varepsilon^{n_k})} \leq \frac{a_{n_k+1}}{\sqrt{2\bar{J}} (\max_\ell |\tilde{x}_\ell^{n_k}|)},
\end{aligned}$$

for all $j = 1, \ldots, N$, where the second inequality follows by the upper bound of tol_n in (4.36), and the last inequality is due to the definition of ε^{n+1}, which yields

$\frac{\varepsilon^{n+1}}{\varepsilon^n} \leq 1$. Since we assumed $\lim_{k\to\infty} \tilde{x}^{n_k} = x^\lambda \neq 0$, there is a k_0 such that for all $k \geq k_0$, we have that $\max_j |\tilde{x}_j^{n_k}| \geq c > 0$. Since (a_{n_k}) tends to 0, we conclude that $\lim_{n\to\infty} |\hat{x}_j^{n_k+1} w_j^{n_k} - \tilde{x}_j^{n_k+1} w_j^{n_k}| = 0$, and therefore we have (4.58). Note that we will use the notation k_0 several times in the presentation of this proof, but for different arguments. We do not mention it explicitly, but we assume a newly defined k_0 to be always larger or equal to the previously defined one.

Next we show that x^λ is a minimizer of $F_{1,\lambda}$ by verifying conditions (4.38) and (4.39). To this end we notice that by Lemma 4.21 and Remark 4.22 it follows that $\lim_{k\to\infty} \hat{x}_j^{n_k} = \lim_{k\to\infty} \tilde{x}_j^{n_k} = \lim_{k\to\infty} \tilde{x}_j^{n_k-1} = x_j^\lambda$. By means of this result, in the case of $x_j^\lambda \neq 0$, we have, due to continuity arguments, (2.42) and Remark 4.24,

$$\begin{aligned} -(\Phi^*(y - \Phi x^\lambda))_j &= \lim_{k\to\infty} -(\Phi^*(y - \Phi \hat{x}^{n_k}))_j = \lim_{k\to\infty} \lambda \hat{x}_j^{n_k} w_j^{n_k-1} \\ &= \lambda \lim_{k\to\infty} \hat{x}_j^{n_k} ((\tilde{x}_j^{n_k-1})^2 + (\varepsilon^{n_k-1})^2)^{-\frac{1}{2}} \\ &= \lambda x_j^\lambda ((x_j^\lambda)^2 + (0)^2)^{-\frac{1}{2}} = \lambda \operatorname{sign}(x_j^\lambda), \end{aligned}$$

and thus (4.38).

In order to show condition (4.39) for j such that $x_j^\lambda = 0$, we follow the main idea in the proof of [189, Lemma 4.5.9]. Assume

$$\lim_{k\to\infty} \hat{x}_j^{n_k} w_j^{n_k-1} > 1. \tag{4.59}$$

Then there exists an $\epsilon > 0$ and a $k_0 \in \mathbb{N}$, such that for all $k \geq k_0$ the inequality $(\hat{x}_j^{n_k} w_j^{n_k-1})^2 > 1 + \epsilon$ holds. Due to (4.58), we can furthermore choose k_0 large enough such that also $(\tilde{x}_j^{n_k} w_j^{n_k-1})^2 > 1 + \epsilon$ for all $k \geq k_0$. Recalling the identity for w_j^n from Lemma 4.23, we obtain

$$\begin{aligned} (\tilde{x}_j^{n_k})^2 &> (1+\epsilon)((w_j^{n_k-1})^{-1})^2 = (1+\epsilon)((\tilde{x}_j^{n_k-1})^2 + (\varepsilon^{n_k-1})^2) \\ &\geq (1+\epsilon)(\varepsilon^{n_k+1})^2 \geq (1+\epsilon)C|(w_j^{n_k})^{-1}|^{2\phi}|(w_j^{n_k-1})^{-1} - (w_j^{n_k})^{-1}|^{4\phi} \\ &\geq (1+\epsilon)C|\tilde{x}_j^{n_k}|^{2\phi}|(w_j^{n_k-1})^{-1} - (w_j^{n_k})^{-1}|^{4\phi}, \end{aligned} \tag{4.60}$$

where the second inequality follows by the definition of the ε^n, and the third inequality follows from Lemma 4.23. Furthermore, in the last inequality we used that $w_j^n \leq |\tilde{x}_j^n|^{-1}$, which follows directly from the definition of w_j^n. By means of this estimate, we conclude

$$(w_j^{n_k-1})^{-1} \geq (w_j^{n_k})^{-1} - |(w_j^{n_k-1})^{-1} - (w_j^{n_k})^{-1}| > |\tilde{x}_j^{n_k}| - ((1+\epsilon)C)^{-\frac{1}{4\phi}} |\tilde{x}_j^{n_k}|^{\frac{2-2\phi}{4\phi}}.$$

Since $0 < \phi < \frac{1}{3}$, the exponent $\frac{2-2\phi}{4\phi} > 1$. In combination with the fact that $\tilde{x}_j^{n_k}$ is vanishing for $k \to \infty$, we are able to choose k_0 large enough to have $((1+\epsilon)C)^{-\frac{1}{4\phi}} |\tilde{x}_j^{n_k}|^{\frac{2-2\phi}{4\phi}-1} <$

$\bar{\epsilon} := 1 - (1+\epsilon)^{-\frac{1}{2}}$ for all $k \geq k_0$ and therefore

$$(w_j^{n_k-1})^{-1} \geq |\tilde{x}_j^{n_k}|(1-\bar{\epsilon}). \tag{4.61}$$

The combination of (4.60) and (4.61) yields

$$|\tilde{x}_j^{n_k}|^2 > (1+\epsilon)\left(w_j^{n_k-1}\right)^{-2} \geq (1+\epsilon)|\tilde{x}_j^{n_k}|^2(1-\bar{\epsilon})^2. \tag{4.62}$$

Since we have $|\tilde{x}_j^{n_k} w_j^{n_k-1}| > 1+\epsilon$ for all $k \geq k_0$, we also have $\tilde{x}_j^{n_k} \neq 0$, and thus, we can divide in (4.62) by $|\tilde{x}_j^{n_k}|$ and insert the definition of $\bar{\epsilon}$ to obtain

$$1 > (1+\epsilon)(1-\bar{\epsilon})^2 = 1,$$

which is a contradiction, and thus the assumption (4.59) is false. By means of this result and again a continuity argument, we show condition (4.39) by

$$(\Phi^T(y-\Phi x^\lambda))_j = \lim_{k\to\infty}(\Phi^T(y-\Phi\hat{x}^{n_k}))_j = \lambda \lim_{k\to\infty} \hat{x}_j^{n_k} w_j^{n_k-1} \leq \lambda.$$

At this point, we have shown that at least the convergent subsequence $(\tilde{x}^{n_k})_{n_k\in\mathbb{N}}$ is such that its limit x^λ is a minimizer of $F_{1,\lambda}(x)$. To show that this is valid for any convergent subsequence of $(\tilde{x}^n)_{n\in\mathbb{N}}$, we remind that the subsequence $(\tilde{x}^{n_k})_{n_k\in\mathbb{N}}$ is the one of Lemma 4.23, and therefore fulfills (4.57). Thus, we can adapt [189, Lemma 4.6.1] to our case, following the arguments in the proof. These arguments only require the monotonicity of the functional $J_{p,\lambda}$, which we show in Lemma 4.19. Consequently the limit x^λ of any convergent subsequence of $(\tilde{x}^n)_{n\in\mathbb{N}}$ is a minimizer of $F_{1,\lambda}(x)$.

Consider the case $0 < p < 1$. The transformation $\mathcal{N}_\zeta(x)$ defined in (4.35) is continuous and bijective. Thus, $\breve{x}^\lambda := \mathcal{N}_{\upsilon/p}^{-1}(x^\lambda)$ is well-defined, and $x_j^\lambda = 0$ if and only if $\breve{x}_j^\lambda = 0$. At a critical point of the *differentiable* functional $\breve{F}_{p,\lambda}$ its first derivative has to vanish, which is equivalent to the conditions

$$\frac{\upsilon}{p}|x_j|^{\frac{\upsilon-p}{p}}\left(\Phi^*y - \Phi^*\Phi\mathcal{N}_{\upsilon/p}(x)\right)_j + \lambda\upsilon\,\mathrm{sign}(x_j)|x_j|^{\upsilon-1} = 0, \quad j = 1,\dots,N. \tag{4.63}$$

We show now that $\breve{x}^\lambda$ fulfills this first order optimality condition. It is obvious that for all j such that $\breve{x}_j^\lambda = 0$ the condition is trivially fulfilled. Thus, it remains to consider all j where $\breve{x}_j^\lambda \neq 0$. As in the case of $p = 1$, we conclude by Lemma 4.21 and Remark 4.22 that $\lim_{k\to\infty}\hat{x}_j^{n_k} = \lim_{k\to\infty}\tilde{x}_j^{n_k} = \lim_{k\to\infty}\tilde{x}_j^{n_k-1} = x_j^\lambda$. Therefore continuity arguments as well as (2.42) yield

$$\begin{aligned}
-(\Phi^*(y-\Phi x^\lambda))_j &= \lim_{k\to\infty} -(\Phi^*(y-\Phi\hat{x}^{n_k}))_j = \lim_{k\to\infty}\lambda p\hat{x}_j^{n_k}w_j^{n_k-1} \\
&= \lambda p \lim_{k\to\infty}\hat{x}_j^{n_k}((\tilde{x}_j^{n_k-1})^2 + (\varepsilon^{n_k-1})^2)^{-\frac{2-p}{2}} \\
&= \lambda p x_j^\lambda((x_j^\lambda)^2 + (0)^2)^{-\frac{2-p}{2}} = \lambda p\,\mathrm{sign}(x_j^\lambda)|x_j|^{p-1}.
\end{aligned}$$

We replace $x^\lambda = \mathcal{N}_{\upsilon/p}(\breve{x}^\lambda)$ and obtain

$$\begin{aligned}-(\Phi^*(y - \Phi\mathcal{N}_{\upsilon/p}(\breve{x}^\lambda))_j &= \lambda p \operatorname{sign}((\mathcal{N}_{\upsilon/p}(\breve{x}^\lambda))_j)|(\mathcal{N}_{\upsilon/p}(\breve{x}^\lambda))_j|^{p-1} \\ &= \lambda p \operatorname{sign}(\breve{x}_j^\lambda)|\breve{x}_j^\lambda|^{\upsilon-\frac{\upsilon}{p}}.\end{aligned}$$

We multiply this identity by $\frac{\upsilon}{p}|x_j|^{\frac{\upsilon-p}{p}}$ and obtain (4.63).

If $\breve{x}^\lambda$ is also a global minimizer of $\breve{F}_{\upsilon,\lambda}$, then x^λ is a global minimizer of $F_{p,\lambda}$. This is due to the equivalence of the two problems, which was shown in [158, Proposition 2.4] based on the continuity and bijectivity of the mapping $\mathcal{N}_{\upsilon/p}$ [195, Proposition 3.4]. □

4.1.4 Simulations

We illustrate the theoretical results of the sections 4.1.2, and 4.1.3 by means of several numerical experiments. We first show that our modified versions of IRLS yield significant improvements in terms of computational time and often outperform the state-of-the-art methods iterative hard thresholding (IHT) [20] and fast iterative soft thresholding algorithm (FISTA) [13].

Before going into the detailed presentation of the numerical tests, we raise two plain numerical disclaimers concerning the numerical stability of Algorithm 13 (CG-IRLS) and 14 (CG-IRLS-λ):

- The first issue concerns IRLS methods in general: The case where $\varepsilon^n \to 0$, i.e., $x_j^n \to 0$, for some $j \in \{1, \dots, N\}$ and $n \to \infty$, is very likely since our goal is the computation of sparse vectors. In this case w_j^n will for some n become too large to be properly represented by a computer. Thus, in practice, we have to provide a lower bound for ε by some $\varepsilon^{\min} > 0$. Imposing such a limit has the theoretical disadvantage that in general the algorithms are only calculating an approximation of the respective problems (2.3) and (2.39). Therefore, to obtain a "sufficiently good" approximation, one has to choose $\varepsilon^{\min}$ sufficiently small. This raises yet another numerical issue: If we choose, e.g., $\varepsilon^{\min} = 1\text{E-}8$ and assume that also $x_j^n \ll 1$, then w_j^n is of the order 1E+8. Compared to the entries of the matrix Φ, which are of the order 1, any multiplication or addition by such a value will cause serious numerical errors. In this context we cannot expect that the IRLS method reaches high accuracy, and saturation effects of the error are likely to occur before machine precision. For details, we refer to the Section 2.4.1.2, which is exclusively dedicated to this issue.

- The second issue concerns the CG method: In Algorithm 11 and Algorithm 12 we have to divide at some point by $\|B^*p^i\|_{\ell_2}^2$ or $\langle Ap^i, p^i\rangle_{\ell_2}$ respectively. As soon as the residual decreases, also p^i decreases with the same order of magnitude. If the above vector products are at the level of machine precision, e.g. 1E-16,

this would mean that the norm of the residual is of the order of its square-root, here 1E-8. But this is the measure of the stopping criterion. Thus, if we ask for a higher precision of the CG method, the algorithm might become numerically unstable, depending on the machine precision. Such saturation of the error is an intrinsic property of the CG method, and here we want to mention it just as a disclaimer. As described further below, we set the lower bound of the CG tolerance to the value 1E-12, i.e., as soon as this accuracy is reached, we consider the result as "numerically exact". For this particular bound the method works stably on the machine that we used.

In the following, we start with a description of the general test settings, which will be common for both Algorithms 13 and 14. Afterwards we independently analyze the speed of both methods and compare them with state-of-the-art algorithms, namely IHT-k (see Section 2.4.3.2) and FISTA (see Section 2.4.3.1). We respectively start with a single trial, followed by a speed-test on a variety of problems. We will also compare the performance of both CG-IRLS and CG-IRLS-λ for the noiseless case, which leads to surprising results.

4.1.4.1 Test Settings

All tests are performed with MATLAB version R2014a. To exploit the advantage of fast matrix-vector multiplications and to allow high dimensional tests, we use randomly sampled partial discrete cosine transformation matrices Φ. We perform tests in three different dimensional settings (later we will extend them to higher dimension) and choose different values N of the dimension of the signal, the amount m of measurements, the respective sparsity k of the synthesized solutions, and the index K in Algorithm (CG-)IRLS:

	Setting A	**Setting B**	**Setting C**
N	2000	4000	8000
m	800	1600	3200
k	30	60	120
K	50	100	200

For each of these settings, we draw at random a set of 100 synthetic problems on which a speed-test is performed. For each synthetic problem the support Λ is determined by the first k entries of a random permutation of the numbers $1, \dots, N$. Then we draw the sparse vector x^* at random with entries $x^*_i \sim \mathcal{N}(0,1)$ for $i \in \Lambda$ and $x^*_{\Lambda^c} = 0$, and a randomly row sampled normalized discrete cosine matrix Φ, where the full non-normalized discrete cosine matrix is given by

$$\Phi^{\text{full}}_{i,j} = \begin{cases} 1, & i = 1, j = 1, \dots, N, \\ \sqrt{2}\cos\left(\frac{\pi(2j-1)(i-1)}{2N}\right), & 2 \le i \le N, 1 \le j \le N. \end{cases}$$

For a given noise vector e of entries $e_i \sim \mathcal{N}(0, \sigma^2)$, we eventually obtain the measurements $y = \Phi x^* + e$. Later we need to specify the noise level and we will do so by fixing a signal to noise ratio. By assuming that Φ has the RIP of order k (see Definition 2.7), i.e., $\|\Phi z\|_{\ell_2} \sim \|z\|_{\ell_2}$, for all $z \in \mathbb{R}^N$ with $\#\operatorname{supp}(z) \leq k$, we can estimate the measurement signal to noise ratio by

$$\mathrm{MSNR} := \frac{\mathbb{E}(\|\Phi x^*\|_{\ell_2}^2)}{\mathbb{E}(\|e\|_{\ell_2}^2)} \sim \frac{k}{m\sigma^2}.$$

In practice, we set the MSNR first and choose the noise level $\sigma = \frac{\sqrt{k}}{\sqrt{\mathrm{MSNR}m}}$. If $\mathrm{MSNR} = \infty$, the problem is noiseless, i.e., $e = 0$.

4.1.4.2 Algorithm CG-IRLS

Specific settings. We restrict the maximal number of outer iterations to 15. Furthermore, we modify (4.6), so that the CG-algorithm also stops as soon as $\|\rho^{n+1,i}\|_{\ell_2} \leq$ 1E-12. As soon as the residual undergoes this particular threshold, we call the CG solution (numerically) "exact". The ε-update rule is extended as in (2.31) by imposing the lower bound $\varepsilon^n = \varepsilon^n \vee \varepsilon^{\min}$ where $\varepsilon^{\min} =$ 1E-9$/N$. The summable sequence $(a_n)_{n\in\mathbb{N}}$ in Theorem 4.4 is defined by $a_n = 100 \cdot (1/2)^n$.

As we define the synthetic tests by choosing the solution x^* of the linear system $\Phi x^* = y$ (here we assume $e = 0$), we can use it to determine the error of the iterations $\|\tilde{x}^n - x^*\|_{\ell_2}$.

IRLS vs. CG-IRLS To get an immediate impression about the general behavior of CG-IRLS, we compare its performance in terms of accuracy and speed to IRLS, where the intermediate linear systems are solved exactly via Gaussian elimination (i.e., by the standard MATLAB backslash operator). We choose IHT as a first order state-of-the-art benchmark, to get a fair comparison with another method which can exploit fast matrix-vector multiplications.

In this first single trial experiment, we choose an instance of setting B, and set $p = 1$ for CG-IRLS and compare it to IRLS with different values of p. The result is presented in the left plot of Figure 4.1. We show the decrease of the relative error in ℓ_2-norm as a function of the computational time. One sees that the computational time of IRLS is significantly outperformed by CG-IRLS and by the exploitation of fast matrix-vector multiplications. The standard IRLS is not competitive in terms of computational time, even if we choose $p < 1$, which is known to yield super-linear convergence [48]. With increasing dimension of the problem, in general the advantage of using the CG method becomes even more significant. However CG-IRLS does not outperform yet IHT in terms of computational time. We also observe the expected numerical error saturation

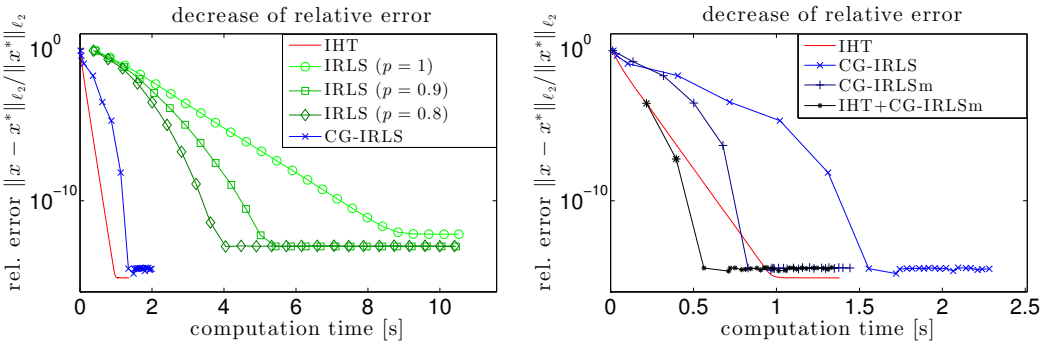


Figure 4.1: Single trial of Setting B. Left: Relative error plotted against the computational time for IRLS[$p = 1$] (light green, $\circ$), IRLS[$p = 0.9$] (green, $\square$), IRLS[$p = 0.8$] (dark green, $\diamond$), CG-IRLS (blue, $\times$), and IHT (red, $-$). Right: Relative error plotted against computational time for CG-IRLS (blue, $\times$), CG-IRLSm (dark blue, $+$), IHT+CG-IRLSm (black, $*$), and IHT (red, $-$).

(as mentioned at the beginning of this section), which appears as soon as the accuracy falls below 1E-13.

For this test, we set the parameter β in the ε-update rule to 2. We comment on the choice of this particular parameter in a dedicated paragraph below.

Modifications to CG-IRLS As we have shown by a single trial in the previous paragraph, CG-IRLS as it is presented in Section 4.1.2 is not able to outperform IHT. Therefore, we introduce the following practical modifications to the algorithm:

(i) We introduce the parameter `maxiter_cg`, which defines the maximal number of inner CG iterations. Thus, the inner loop of the algorithm stops as soon as `maxiter_cg` iterations were performed, even if the theoretical tolerance tol_n is not reached yet.

(ii) CG-IRLS includes a stopping criterion depending on tol_{n+1}, which is only *implicitly* given as a function of ε^{n+1} (compare Section 4.1.2.2, and in particular formulas (4.6) and (4.7)), which in turn depends on the current $\tilde{x}^{n+1}$ by means of sorting and a matrix-vector multiplication. To further reduce the computational cost of each iteration, we avoid the aforementioned operations by only updating tol_{n+1} outside the MCG loop, i.e., after the computation of $\tilde{x}^{n+1}$ with fixed tol_{n+1} we update ε^{n+1} as in Step 3 of Algorithm 13 and subsequently update tol_{n+2} which again is fixed for the computation of $\tilde{x}^{n+2}$.

(iii) The left plot of Figure 4.1 reveals that in the beginning CG-IRLS reduces the

error more slowly than IHT, and it gets faster after it reached a certain ball around the solution. Therefore, we use IHT as a warm up for CG-IRLS, in the sense that we apply a number `start_iht` of IHT iterations to compute a proper starting vector for CG-IRLS.

We call *CG-IRLSm* the algorithm with modifications (i) and (ii), and *IHT+CG-IRLSm* the algorithm with modifications (i), (ii), and (iii). We set `maxiter_cg` $= \lfloor m/12 \rfloor$, `start_iht` $= 150$, and we set β to 0.5. If these algorithms are executed on the same trial as in the previous paragraph, we obtain the result which is shown on the right plot in Figure 4.1. For this trial, the modified algorithms show a significantly reduced computational time with respect to the unmodified version and they now converge faster than IHT. However, the introduction of the practical modifications (i)–(iii) does not necessarily comply anymore with the assumptions of Theorem 4.4. Therefore, we do not have rigorous convergence and recovery guarantees anymore and recovery might potentially fail more often. In the next paragraph, we empirically investigate the failure rate and explore the performance of the different methods on a sufficiently large test set.

Another natural modification to CG-IRLS consists in the introduction of a preconditioner to compensate for the deterioriation of the condition number of $\Phi D_n \Phi^*$ as soon as ε^n becomes too small (when w^n becomes very large). The matrix $\Phi\Phi^*$ is very well conditioned, while the matrix $\Phi D_n \Phi^*$ "sandwiching" D_n becomes more ill-conditioned as n gets larger, and, unfortunately, it is hard to identify additional "sandwiching" preconditioners P_n such that the matrix $P_n \Phi D_n \Phi^* P_n^*$ is suitably well-conditioned. In the numerical experiments standard preconditioners failed to yield any significant improvement in terms of convergence speed. Hence, we refrained from introducing further preconditioners. Instead, as we will show at the end of Subsection 4.1.4.3, a standard (Jacobi) preconditioning of the matrix

$$\left(\Phi^*\Phi + \operatorname{diag}\left[\lambda p w_j^n \right]_{j=1}^N \right),$$

where the source of singularity is added to the product $\Phi^*\Phi$, leads to a dramatic improvement of computational speed.

Empirical test on computational time and failure rate In the following, we define a method to be "successful" if it is computing a solution x for which the relative error $\|x - x^*\|_{\ell_2} / \|x^*\|_{\ell_2} \leq$ 1E-13. The computational time of a method is measured by the time it needs to produce the first iterate which reaches this accuracy. In the following, we present the results of a test which runs the methods CG-IRLS, CG-IRLSm, IHT+CG-IRLSm, and IHT on 100 trials of Setting A, B, and C respectively and $p \in \{1, 0.9, 0.8\}$. For values of $p < 0.8$ the methods become unstable, due to the severe nonconvexity of the problem and it seems that good performance cannot be

reached below this level. Therefore we do not investigate further these cases. Let us stress that IHT does not depend on p.

In each setting we check for each trial which methods succeeds or fails. If all methods succeed, we compare the computational time, determine the fastest method, and count the computational time of each method for the respective mean computational time. The results are shown in Figure 4.2. By analyzing the diagrams, we are able to distill the following observations:

- Especially in Setting A and B, CG-IRLSm and IHT+CG-IRLSm are better or comparable to IHT in terms of mean computational time and provide in most cases the fastest method. CG-IRLS performs much worse. The failure rate of all the methods is negligible here.
- The gap in the computational time between all methods becomes larger when N is larger.
- With increasing dimension of the problem, the advantage of using the modified CG-IRLS methods subsides, in particular in Setting C.
- In the literature [36, 38, 37, 48] superlinear convergence is reported for $p < 1$, and perhaps one of the most surprising outcomes is that the best results for all CG-IRLS methods are instead obtained for $p = 1$. This can probably be explained by observing that superlinear convergence kicks in only in a rather small ball around the solution and hence does not necessarily improve the actual computation time!
- Not only the computational performance, but also the failure rate of the CG-IRLS based methods increases with decreasing p. However, as expected, CG-IRLS succeeds in the convex case of $p = 1$. The failure of CG-IRLS for $p < 1$ can probably be attributed to non-convexity.

We conclude that CG-IRLSm and IHT+CG-IRLSm perform well for $p = 1$ and for the problem dimension N within the range of 1000 – 10000. They are even able to outperform IHT. However, by extrapolation of the numerical results IHT is expected to be faster for $N > 10000$. (This is in compliance with the general folklore that first order methods should be preferred for higher dimension. However, as we will see in Subsection 4.1.4.3, a proper preconditioning of CG-IRLS-λ will win over IHT for dimensions $N \geq 10^5$!) As soon as $N < 1000$, direct methods such as Gaussian elimination are faster than CG, and thus, one should use standard IRLS with $p < 1$.

Choice of β, `maxiter_cg`, and `start_iht` The numerical tests in the previous paragraph were preceded by a careful and systematic investigation of the tuning of the parameters β, `maxiter_cg`, and `start_iht`. While we fixed `start_iht` to 100, 150, and

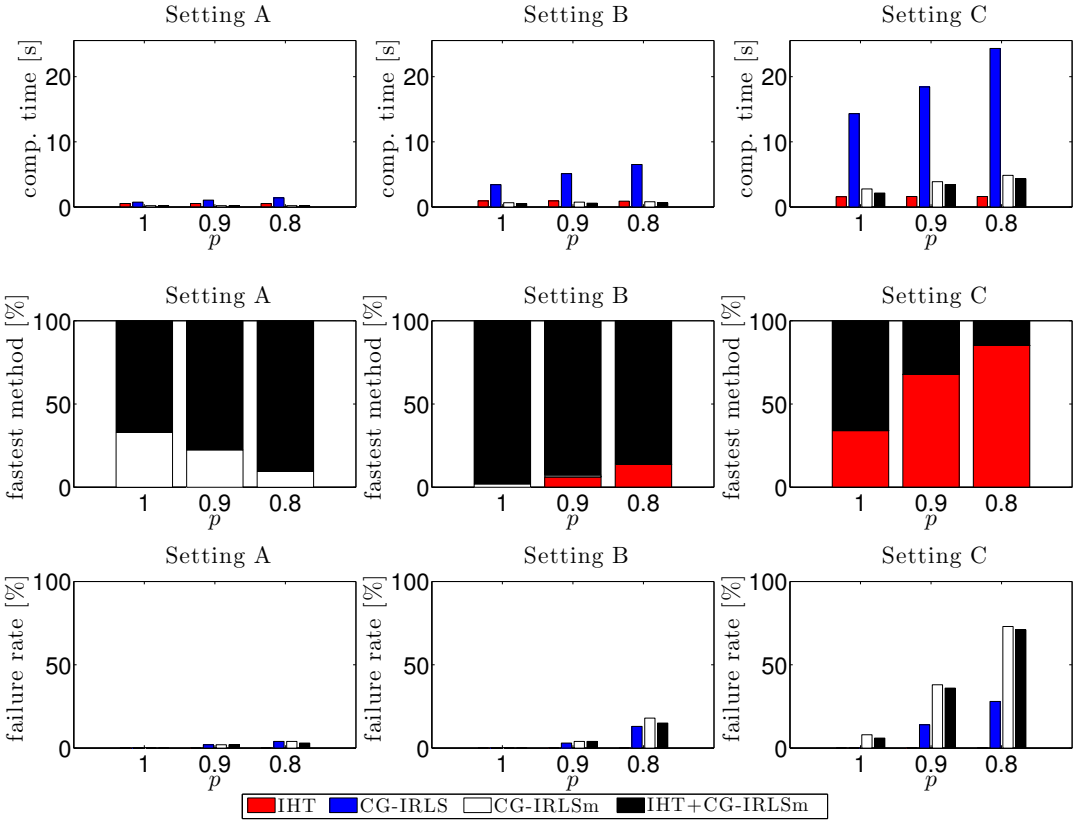


Figure 4.2: Empirical test on Setting A, B, and C for the methods CG-IRLS (blue), CG-IRLSm (white), IHT+CG-IRLSm (black), and IHT (red). Upper: Mean computational time. Center: Fastest method (in %). Lower: Failure rate (in %).

200 for Setting A, B, and C respectively to produce a good starting value, we tried $\beta \in \{1/N, 0.01, 0.1, 0.5, 0.75, 1, 1.5, 2, 5, 10\}$, and `maxiter_cg` $\in \{\lfloor m/8 \rfloor, \lfloor m/12 \rfloor, \lfloor m/16 \rfloor\}$ for each setting. The results of this parameter sensitivity study can be summarized as follows:

- The best computational time is obtained for $\beta \sim 1$. In particular the computational time is not depending substantially on β in this order of magnitude. More precisely, for CG-IRLS the choice of $\beta = 0.5$ and for (IHT+)CG-IRLSm the choice of $\beta = 2$ works best.

- The choice of `maxiter_cg` very much determines the tradeoff between failure and speed of the method. The value $\lfloor m/12 \rfloor$ seems to be the best compromise. For a smaller value the failure rate becomes too high, for a larger value the method is

too slow.

Phase transition diagrams. Besides the empirical analysis of the speed of convergence, we also investigate the robustness of CG-IRLS with respect to the achievable sparsity level for exact recovery of x^*. Therefore, we fix $N = 2000$ and we compute a phase transition diagram for IHT and CG-IRLS on a regular Cartesian 50×40 grid, where one axis represents m/N and the other represents k/m. For each grid point we plot the empirical success recovery rate, which is numerically realized by running both algorithms on 20 random trials. CG-IRLS or IHT is successful if it is able to compute a solution with a relative error of less than 1E-4 within 20 or 500 (outer) iterations respectively. Since we aim at simulating a setting in which the sparsity k is not known exactly, we set the parameter $K = 1.1 \cdot k$ for both IHT and CG-IRLS. The interpolated plot is shown in Figure 4.3. It turns out that CG-IRLS has a significantly higher success recovery rate than IHT for less sparse solutions.

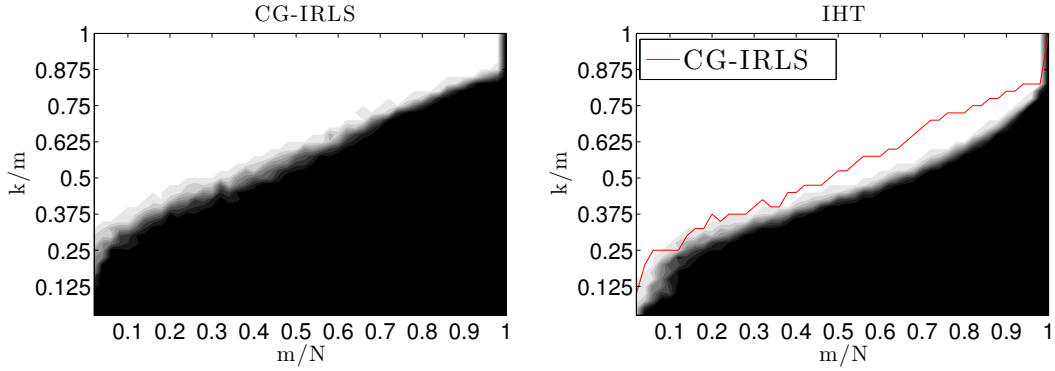


Figure 4.3: Phase transition diagrams of IHT and CG-IRLS for $N = 2000$. The recovery rate is presented in grayscale values from 0% (white) up to 100% (black). As a reference, in the right subfigure, the 90% recovery rate level line of the CG-IRLS phase transition diagram is plotted to show more evidently the improved success rate of the latter algorithm.

4.1.4.3 Algorithm CG-IRLS-λ

Specific settings We restrict the maximal number of outer iterations to 25. Furthermore, we modify (4.33), so that the CG-algorithm also stops as soon as $\|\rho^{n+1,i}\|_{\ell_2} \leq$ 1E-16 $\cdot N^{3/2}m$. As soon as the residual undergoes this particular threshold, we call the CG solution (numerically) "exact". The ε-update rule is extended by imposing the lower bound $\varepsilon^n = \varepsilon^n \vee \varepsilon^{\min}$ where $\varepsilon^{\min} =$ 1E-9. Additionally we propose to choose $\varepsilon^{n+1} \leq 0.8^n \varepsilon^n$, which practically turns out to increase dramatically the speed

of convergence. The summable sequence $(a_n)_{n\in\mathbb{N}}$ in Theorem 4.15 is defined by setting $a_n = \sqrt{Nm} \cdot 10^4 \cdot (1/2)^n$. We split our investigation into a noisy and a noiseless setting.

For the noisy setting we set MSNR = 100. According to [15, 27], we choose $\lambda = c\sigma\sqrt{m \log N}$ as a near-optimal regularization parameter, where we empirically determine $c = 0.48$. Since we work with relatively large values of λ in the regularized problem (2.39), we cannot use the synthesized sparse solution x^* as a reference for the convergence analysis. Instead, we need another reliable method to compute the minimizer of the functional. In the convex case of $p = 1$, this is performed by the well-known and fast algorithm FISTA [13], which shall also serve as a benchmark for the speed analysis. In the non-convex case of $p < 1$, there is no method which guarantees the computation of the global minimizer, thus, we have to omit a detailed speed-test in this case. However, we describe the behavior of Algorithm 14 for p changing.

If the problem is noiseless, i.e., $e = 0$, the solution x^λ of (2.39) converges to the solution of (2.3) for $\lambda \to 0$ (compare Lemma 2.15 for the case $p = 1$). Thus, we choose $\lambda = m \cdot$ 1E-8, and assume the synthesized sparse solution x^* as a good proxy for the minimizer and a reference for the convergence analysis. (Actually, this can also be seen the other way around, i.e., we use the minimizer x^λ of the regularized functional to compute a good approximation to x^*.) It turns out that for $\lambda \approx 0$, as we comment below in more detail, FISTA is basically of no use.

CG-IRLS-λ vs. IRLS-λ As in the previous subsection, we first show that the CG-method within IRLS-λ leads to significant improvements in terms of the computational speed. Therefore we choose a noisy trial of Setting B, and compare the computational time of the methods IRLS-λ, CG-IRLS-λ, and FISTA. The result is presented on the left plot of Figure 4.4. We observe that CG-IRLS-λ computes the first iterations in much less time than IRLS-λ, but due to bad conditioning of the inner CG problems it performs much worse afterwards. Furthermore, as may be expected, the algorithm is not suitable to compute a highly accurate solution. For the computation of a solution with a relative error in the order of 1E-3, CG-IRLS-λ outperforms FISTA. FISTA is able to compute highly accurate solutions, but a solution with a relative error of 1E-3 should be sufficient in most applications because the goal in general is not to compute the minimizer of the Lagrangian functional but an approximation of the sparse signal.

Modifications to CG-IRLS-λ To further decrease the computational time of CG-IRLS-λ, we propose the following modifications:

(i) To overcome the bad conditioning in the CG loop, we precondition the matrix $A_n = \Phi^*\Phi + \operatorname{diag}\left[\lambda p w_j^n\right]_{j=1}^N$ by means of the Jacobi preconditioner, i.e., we pre-multiply the linear system by the inverse of its diagonal, $(\operatorname{diag} A_n)^{-1}$, which is a very efficient operation in practice.

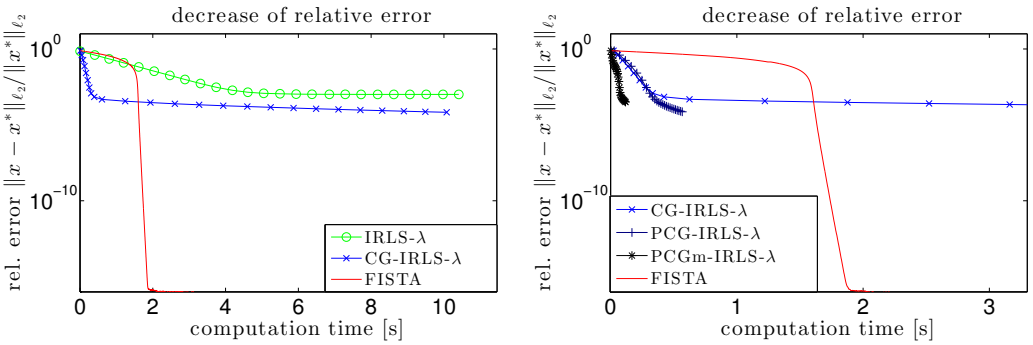


Figure 4.4: Single trial of Setting B. Left: Relative error plotted against the computational time for IRLS-λ (light green, $\circ$), CG-IRLS-λ (blue, $\times$), and FISTA (red, $-$). Right: Relative error plotted against computational time for CG-IRLS-λ (blue, $\times$), PCG-IRLS-λ (dark blue, $+$), PCGm-IRLS-λ (black, $*$), and FISTA (red, $-$).

(ii) We introduce the parameter `maxiter_cg` which defines the maximal number of inner CG iterations and is set to the value `maxiter_cg` $= 4$ in the following.

The algorithm with modification (i) is called PCG-IRLS-λ, and the one with modification (i) and (ii) PCGm-IRLS-λ. We run these algorithms on the same trial of Setting B as in the previous paragraph. The respective result is shown on the right plot of Figure 4.4. This time, preconditioning effectively yields a strong decrease of computational time, especially in the final iterations where A_n is badly conditioned. Furthermore, modification (ii) importantly increases the performance of the proposed algorithm also in the initial iterations. However, again we have to take into consideration that we may violate the assumptions of Theorem 4.15 so that convergence is not guaranteed anymore and failure rates might potentially increase. In the following two paragraphs, we present simulations on noisy and noiseless data, which give a more precise picture of the speed and failure rate of the previously introduced methods in comparison to FISTA and IHT.

Empirical test on computational time and failure rate with noisy data In the previous paragraph, we observed that the CG-IRLS-λ methods are only computing efficiently solutions with a small relative error. Thus we now focus on this setting and compare the three methods PCG-IRLS-λ, PCGm-IRLS-λ, and FISTA with respect to their computational time and failure rate in recovering solutions with a relative error of 1E-1, 1E-2, and 1E-3. We only consider the convex case $p = 1$. Similarly to the procedure in Section 4.1.4.2, we run these algorithms on 100 trials for each setting with the respectively chosen values of λ. In Figure 4.5 the upper bar plot shows the

result for the mean computational time and the lower stacked bar plot shows how often a method was the fastest one. We do not present a plot of the failure rate since none of the methods failed at all. By means of the plots, we demonstrate that both PCG-IRLS-λ, and PCGm-IRLS-λ are faster than FISTA, while PCGm-IRLS-λ always performs best.

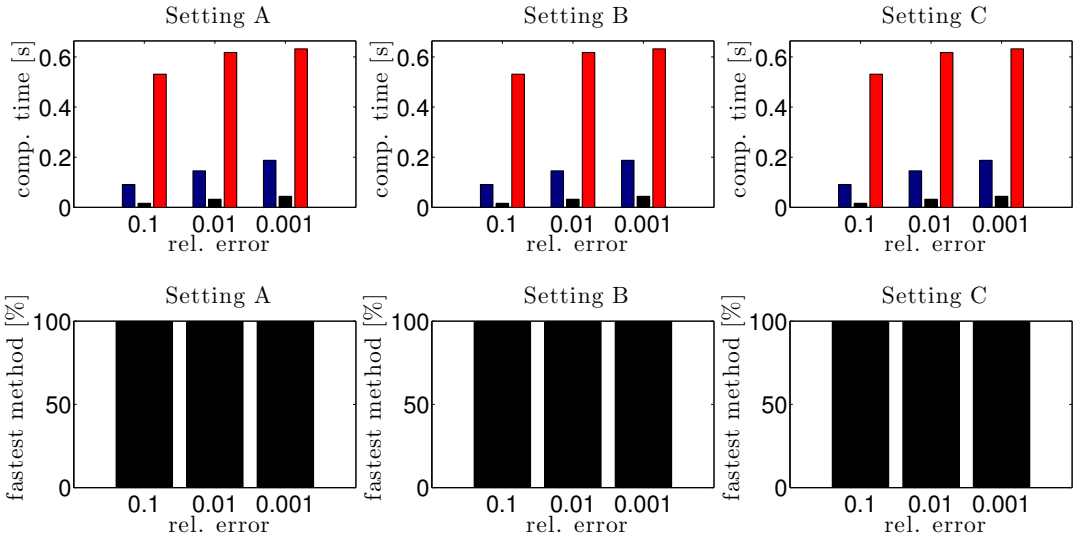


Figure 4.5: Empirical test on Setting A, B, and C for the methods PCG-IRLS-λ (blue), PCGm-IRLS-λ (black), and FISTA (red). Upper: Mean computational time. Lower: Fastest method (in %).

Empirical test on computational time and failure rate with noiseless data In the noiseless case, we compare the computational time of FISTA and PCGm-IRLS-λ to IHT and IHT+CG-IRLSm. We set `maxiter_cg` $= 40$ for PCGm-IRLS-λ. In a first test, we run these algorithms on one trial of Setting A, and C respectively, and plot the results in Figure 4.6.

As already mentioned, FISTA is not suitable for small values of λ on the order of $m \cdot 1\text{E-}8$ and converges then extremely slowly, but PCGm-IRLS-λ can compete with the remaining methods. IHT+CG-IRLSm is in some settings able to outperform IHT, at least when a high accuracy is needed. PCGm-IRLS-λ is always at least as fast as IHT with increasing relative performance gain for increasing dimensions. This observation suggests the conjecture that PCGm-IRLS-λ provides the fastest method also in rather high dimensional problems. To validate this hypothesis numerically, we introduce two new high dimensional settings (to reach higher dimensionalities and retaining low

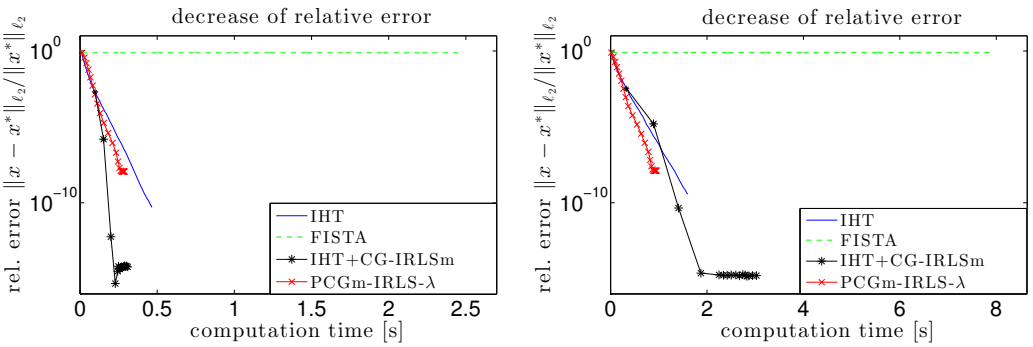


Figure 4.6: Left: Setting A. Right: Setting C. Comparison of IHT (blue, $-$), FISTA (green, $--$), IHT+CG-IRLSm (black, $*$), and PCGm-IRLS-λ (red, $\times$).

computation times for the extensive tests it is again very beneficial to use the real cosine transform as a model for Φ):

	Setting D	**Setting E**
N	100000	1000000
m	40000	400000
k	1500	15000
K	2500	25000

We run the most promising algorithms IHT and PCGm-IRLS-λ on a trial of the large scale settings D and E. The result which is plotted in Figure 4.7 shows that PCGm-IRLS-λ is able to outperform IHT in these settings unless one requires an extremely low relative error ($\leq$ 1E-8), because of the error saturation effect. We confirm this outcome in a test on 100 trials for Setting D and E and present the result in Figure 4.8.

Dependence on p In the last experiment of this section, we are interested in the influence of the parameter p. Of course, changing p also means modifying the problem resulting in a different minimizer. Due to non-convexity also spurious local minimizers may appear. Therefore, we do not compare the speed of the method to FISTA. In Figure 4.9, we show the performance of Algorithm PCGm-IRLS-λ for a single trial of Setting C and the parameters $p \in \{1, 0.9, 0.8, 0.7\}$ for the noisy and noiseless setting. As reference for the error analysis, we choose the sparse synthetic solution x^*, which is actually not the minimizer here.

In both the noisy and noiseless setting, using a parameter $p < 1$ improves the computational time of the algorithm. In the noiseless case, $p = 0.9$ seems to be a good

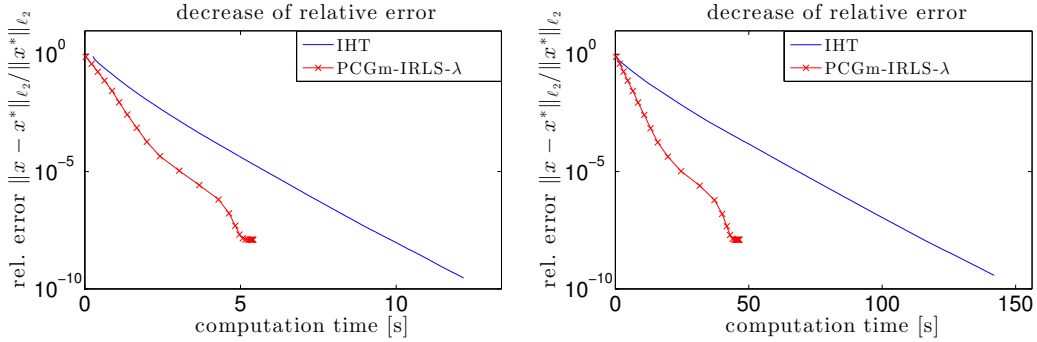


Figure 4.7: Left: Setting D. Right: Setting E. Comparison of IHT (blue, −), and PCGm-IRLS-λ (red, $\times$).

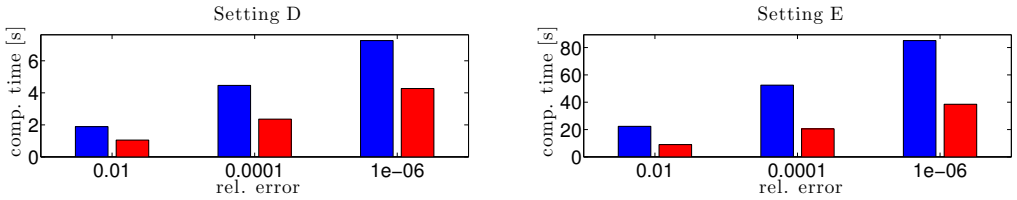


Figure 4.8: Empirical test on the mean computational time of Setting D and E for the methods IHT (blue), and PCGm-IRLS-λ (red).

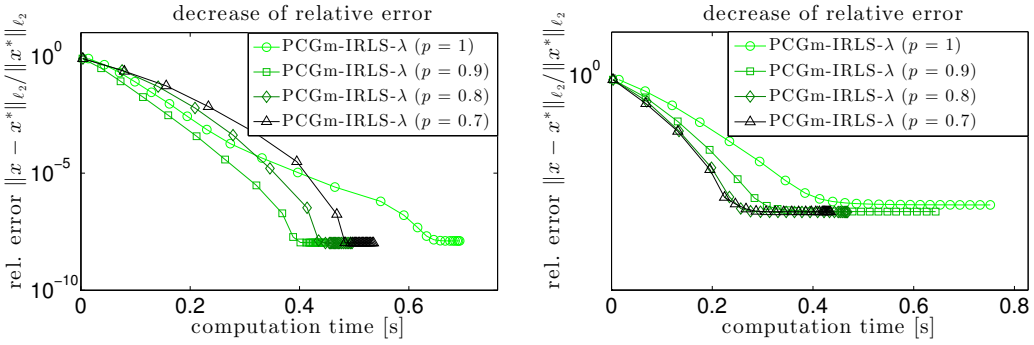


Figure 4.9: Results of Algorithm PCGm-IRLS-λ for a single trial of Setting C for different values of p with noise (right) and without noise (left).

choice, smaller values do not improve the performance. In contrast, in the noisy setting the computational time decreases with decreasing p.

4.2 Parallel Domain Decomposition Based Solutions for the Iterative Soft Thresholding Algorithm

The reason why we are interested in parallel solutions for sparse recovery problems is that we may dispose of an encoder that is only available as a full matrix and may not fit in the memory of one single machine. In another situation, we may have to solve a bunch of similar smaller problems, which fit in the memory, but their parallel treatment dramatically reduces the computational time for the entire dataset. Moreover, also big data problems that fit in the memory of a machine may be considered for parallelization since it decreases in most cases the computation time. In particular for readers, who are not so familiar with general parallelization concepts, we open this section by a respective overview.

We distinguish between *data parallelism* and *task parallelism*. In the first concept, each machine is doing the same calculations on different data, in the second concept, each machine is specialized on a particular task. An example for task parallelism is pipelining, e.g., if one considers a bunch of images where each has to be transformed into a black/white-image and sharpened. This task can be accomplished by processing an image first by a machine that transforms the color into gray level, and afterwards, sharpen it by another machine, while the first machine processes the next image, and so on. In our investigation, we will mainly focus on data parallelism since we essentially have to perform a single task on data of large size.

Parallel implementations are based on either the *shared* or the *distributed memory paradigm*. It is characteristic for the shared memory paradigm that all cores can access the same storage and one has to take care about concurrent writing operations to the same range of memory (e.g., multi-threading/OpenMP). In the distributed memory paradigm, each core can only access its own memory, and one has to synchronize the cores among each other by means of communication through data and information transfer, also called *message passing* (e.g., MPI). Since we talk about solutions for big data, and since we only have at disposal conventional (parallel) systems with very restricted RAM per processing unit, we necessarily have to split the data and focus on the distributed memory paradigm in this section.

In order to design a parallel algorithm, in principle two different approaches can be followed: Either one takes an existing sequential algorithm[1] and tries to distribute

[1] With the term "sequential" we mean that all operations of the algorithm are sequentially executed.

each calculation step and its respective data requirements to multiple cores without changing the essential operating principle of the algorithm, or one creates an algorithm from scratch based on a novel methodology which exploits in particular the data and computation distribution. If the individual operations of a sequential algorithm can be *parallelized* in the first case above, then we call such a parallelization *native*. The advantage of a native parallel algorithm is that one does not need to (re)invent the flow of data processing, and one can concentrate on the efficiency of the parallelization[2], which is defined as follows:

Definition 4.25 (Parallel speedup and efficiency)
Let $\theta(P)$ measure the parallel running time of an algorithm which is parallelized on P cores. In particular $\theta(1)$ is the time of the sequential version (on a single core) of an algorithm. The *parallel speedup* is defined as

$$\text{speedup}(P) := \frac{\theta(1)}{\theta(P)}, \tag{4.64}$$

and the *parallel efficiency* is defined as

$$\text{eff}(P) := \frac{\theta(1)}{P\,\theta(P)}. \tag{4.65}$$

We distinguish between sub-linear ($< P$), linear ($= P$), and super-linear ($> P$) parallel speedup.

A parallelization is considered inefficient if and only if $\text{eff}(P) \leq 1/P$. Then the speedup is less or equal than 1, and the execution of the parallelized version of the algorithm takes more time than the sequential one. The efficiency of an algorithm depends on theoretical (algorithmic concept) and practical (implementation, computer architecture) aspects.

In order to explain positive efficiency effects on a solely theoretical level for native parallelizations, we assume a homogeneous system with P similar cores and exclude any hardware or implementation effects. In a native parallelization of an algorithm, where we only distribute the constant total amount of work to P cores (and thus split the computational cost), it is rather obvious that the ideal speedup can only be linear, and thus the parallel efficiency is bounded by one.

If a non-native algorithm can have a theoretical super-linear speedup—and thus a parallel efficiency above one—is controversially discussed by scientists, and has been proved and disproved by different underlying models, see, e.g., [24, 184, 107, 153,

[2]Let us emphasize that we consider this topic in this section from a theoretical point of view describing an exact precision data processing. A deterministic natively parallelized algorithm, which does not contain any randomness, should produce the same (intermediate) results as its sequential version. However, in practice this is very unlikely to be achieved since truncation errors (and its propagation) are the reason that both versions are not exactly identical.

102]. However, results with super-linear speedup were reported, e.g., for evolutionary algorithms [156, 2, 167], and branch-and-bound algorithms [121]. In those examples, the parallel algorithms use a strategy, in which one core may produce results that affect the amount of work of other cores [135, p.127–128].

From a practical point of view, one possibility to gain efficiency in native and non-native parallelization is the distribution of data into very small portions such that they can be moved from the RAM to the cache of a core. Usually, the cache allows much faster access times, and one may observe (even for native parallelization) a super-linear speedup. Such a technique was exploited for instance in the FFTW [86].

We present in the last paragraph of Section 4.2.1 another non-natively parallelized algorithm for sparse recovery, where super-linear speedup is expected according to the theoretical investigations, and we indeed observe it in the numerical experiments. Nonetheless, since numerical tests are often performed on complicated hardware architectures, perhaps accompanied by background processes and compiler optimization in the software, it is in general difficult to clearly identify in numerical experiments whether a super-linear effect is due to theoretical or practical aspects.

Note that in general any non-parallelized step of an algorithm and any communication (message passing) among the cores are borne by the parallel efficiency. This implies that parallelization always has limitations. In fact any algorithm always consists of one part of the computations that can be parallelized, and one part that cannot be parallelized. The parallel computation time for the non-parallelizable part is constant and not decreasing with an increasing number of used cores P. Thus, if P is large enough, the computation time of the non-parallelizable part dominates and prevents a further speedup.

So far, we implicitly assumed that we have to solve only one single problem. Hereafter we cover the case where we have to solve $Q > 1$ problems independently, i.e., the solution of each problem can be computed without the knowledge of any (intermediate) result of the other problems. An example for such a situation is the simple downsampling of an image, where one splits it into subimages, which are processed independently. Consider the (simplified) setting, where one has $P_{\max}$ cores available and needs to solve $Q \geq P_{\max}$ problems of the same type. Assume further that each problem has approximately the same cost, and $P_{\max}$ being a divisor of Q. If an algorithm that solves one of those problems cannot be parallelized with a super-linear speedup, it is reasonable to run the non-parallelized version of the algorithm independently on each core. If a super-linear speedup is possible, one has to figure out the optimal number of cores P_{opt} with the best parallel efficiency and solve each of the Q problems by the parallelized version of the algorithm on a bundle of P_{opt} cores. We illustrate this abstract description by the following example.

Example 4.26
Set $P_{\max} = 4$, $Q = 8$, and assume that one single problem will have the following parallel computation times depending on the number of cores P (sub-linear speedup):

P	1	2	4
parallel running time $\theta(P)$ in s	1	0.75	0.5
speedup(P)	1	4/3	2
eff(P)	1	2/3	1/2

In order to compute $Q = 8$ problems, and since we have $P_{\max} = 4$ cores available, we can decide whether we either compute 2 problems on each core, or split the number of cores into 2×2 cores and give each group 4 problems, or use the entire group of 4 cores for each problem and let them compute all 8 problems. Then, we would need 2s (2 problems times 1s), 3s (4 problems times 0.75s), or 4s (8 problems times 0.5s) respectively in order to compute the totality of 8 problems. Thus, it is best to prefer the first variant where one does process each problem non-parallelized.

Let us now assume that one problem can be solved by parallel computation times that exhibit a super-linear speedup:

P	1	2	4
parallel running time $\theta(P)$ in s	1	0.3	0.2
speedup(P)	1	10/3	5
eff(P)	1	5/3	5/4

In this case, the best parallel efficiency is obtained for $P_{\text{opt}} = 2$. For the totality of 8 problems one would need 2s, 1.2s, or 1.6s respectively, and thus, one obtains the optimal result in terms of parallel computational time, when splitting up the number of cores into 2 groups of 2 cores, and processes on each group 4 problems.

We draw here a rather rough view on parallelization since the intention is to familiarize the reader with the mainly used concepts and not with technical details. Naturally, in practice one has to consider even more effects like, e.g., reading and writing of data, or more sophisticated strategies, when Q is not divisible by $P_{\max}$ etc. For further reading we refer to relevant textbooks, e.g. [192, 119, 124].

In the following sections, we explore parallel solutions for the ℓ_1-regularized least squares problem (2.15). At first, we recall the state-of-the-art in Section 4.2.1, giving a detailed overview on existing literature and the respective parallel algorithms. Then, we propose an algorithm that is based on the idea of subspace correction methods for sparse recovery, which was presented in the early work [74]. The advantage of the subspace correction technique is that it allows multiple internal iterations on each core without intermediate message passing in each iteration. The findings of the paper [74] were only rudimentarily verified by numerical experiments since it was mainly of conceptual

nature. No follow-up research on the subspace correction methods, introduced in [74], has been reported in the literature, except for [80]. In Section 4.2.2, we thoroughly analyze and test this parallel algorithm to eventually formulate a more efficient version that is able to outperform the state-of-the-art algorithms, as we demonstrate in extensive numerical experiments (Section 4.2.5). Furthermore, we prove the convergence of the newly designed method in Section 4.2.3 and clarify some implementation details in Section 4.2.4.

The research in Sections 4.2.1–4.2.5 is tailored to the situation of large sparse recovery problems, i.e., the encoding matrix $\Phi \in \mathbb{R}^{m\times N}$ has very large dimensions N and m. By contrast, in Section 4.2.6 we consider the scenario of having a large number of smaller sparse recovery problems of similar size. We address the problem of an efficient scheduling of the individual sparse recovery problems on a parallel machine, using the example of a large-scale hyperspectral-multispectral image fusion. Therefore, we choose the best possible solver for an individual problem among the state-of-the-art solutions, as presented in 4.2.1–4.2.5. Then we describe advantages and disadvantages of *fixed* and *work-stealing* scheduling techniques, propose a combination of both, and present empirical tests, applying this strategy.

4.2.1 Parallel Algorithms for the ℓ_1-regularized Least Squares Problem

Let us consider a joint sparsity problem (see Section 2.3) with measurement data, given as a vector in a finite dimensional space, i.e., $y \in \mathcal{H} := \mathbb{R}^{m\times d}$, $d \geq 1$, and let $T\colon \ell_2(\Lambda) \to \mathcal{H}$ be a bounded linear operator, based on a countable set Λ modeling the measurement process. We want to reconstruct a potentially joint sparse signal $u \in \ell_2(\Lambda)$ from given y by the minimization problem (2.15). We remind that we simplified the notation $\|\cdot\|_{\ell_{p,2}(\Lambda)} = \|\cdot\|_{\ell_p(\Lambda)}$ (compare Section 1.2), and thus, the problem also reads

$$\operatorname*{arg\,min}_{u\in\ell_2(\Lambda)} \left(\mathcal{J}_\lambda(u) := \|Tu - y\|_{\mathcal{H}}^2 + \lambda\|u\|_{\ell_{1,2}(\Lambda)}\right),$$

which is the most common formulation. However, we stick to the first version (2.15) for the sake of a simple notation.

A Parallelized Iterative Soft-Thresholding Algorithm In order to solve problem (2.15), a very simple algorithm is the well-known *iterative soft thresholding algorithm* (ISTA, Section 2.4.3.1). Choosing a starting vector $u^{(0)} \in \ell_1(\Lambda)$, it iterates

$$u^{(n+1)} = \mathbb{S}_{t\lambda}(u^{(n)} + tT^*(y - Tu^{(n)})),$$

with positive stepsize $t \in \mathbb{R}$, where the thresholding operator $\mathbb{S}_{t\lambda}$ is defined as in (2.51), but for a countable index set, i.e.,

$$(\mathbb{S}_{t\lambda}(u))_i := \begin{cases} \left(1 - \frac{t\lambda}{2\|u_i\|_{\ell_2}}\right) u_i, & \|u_i\|_{\ell_2} > \frac{t\lambda}{2}, \\ 0, & \text{otherwise}, \end{cases} \quad i \in \Lambda.$$

In order to formulate a parallelized version of this algorithm, let us decompose the countable index set Λ in P disjoint subsets $\Lambda_1, \dots, \Lambda_P$. Furthermore, we define for each $i = 1, \dots, P$ the operator T_i, which is the restriction of T to vectors supported on the index set Λ_i, and respectively u_i the restriction of a vector $u \in \ell_2(\Lambda)$ to Λ_i. Through this data distribution, a parallelization of ISTA is given by Algorithm 15.

Algorithm 15 Parallel ISTA

Set $u_i^{(0)} := 0$, $[Tu^{(0)}] := 0$, $t \in \mathbb{R}$, $0 < t < \|T\|^{-2}$.

1: **while** stopping criterion is not fulfilled **do**
2: **for** $i = 1, \dots, P$ **do in parallel**
3: $u_i^{(n+1)} = \mathbb{S}_{t\lambda}(u_i^{(n)} + tT_i^*(y - [Tu^{(n)}]))$
4: **end for**
5: $[Tu^{(n+1)}] = \sum_{i=1}^{P} T_i u_i^{(n+1)}$ by `ALLREDUCE`
6: **end while**

In particular the readers that are not familiar with parallel computing may stumble over the `ALLREDUCE` command in the description of the algorithm. Let us explain Step 5 more in detail. Actually, this step determines that each core i is calculating in parallel the vector $f_i := T_i u_i^{(n+1)}$, before the sum of all f_i is computed. Since each core only has access to its own memory and therefore to its own f_i, it requires the communication of those results among all cores in order to sum them up. This procedure is called `REDUCE`. The prefix `ALL-` then describes that the result of the sum is communicated back to all cores. This is necessary since the algorithm is symmetric and each core needs this result again in the next iteration in Step 3. We use (here and in the remainder of the text) the notation $[\cdot]$ in order to emphasize that only the result of the embraced term is stored, communicated, and reused. By the usage of the `ALLREDUCE` synchronization we make sure that the parallel version is exactly doing the same as the sequential version of ISTA, and therefore it is a native parallelization. The particular implementation and computational cost of the `ALLREDUCE` operation depends on the used library and underlying hardware architecture. It requires in general $\mathcal{O}(\log P)$ communications [44].

If we consider the finite dimensional problem and equally distribute the index set $\Lambda = \{1, \dots, N\}$ to the subsets $\Lambda_1, \dots, \Lambda_P$, the parallelization in Algorithm 15 requires in principle the storage of the matrix T_i on each core, thus a matrix of the

size $m \times N/P$. Furthermore the complexity of the computation can be essentially reduced to the complexity of the two matrix-matrix multiplications (or matrix-vector multiplications in the case of $d = 1$) by T_i and T_i^*; thus it is $\mathcal{O}(2dmN/P)$. An alternative to the parallelization in Algorithm 15 would be to first `BROADCAST`[3] all $u_i^{(n)}$ to all cores (such that $u^{(n)}$ is available on each core) and then compute $u_i^{(n+1)} = \mathbb{S}_{t\lambda}(u_i^{(n)} + t([T_i^*y] - [T_i^*T]u^{(n)}))$ in parallel, where the matrices $[T_i^*y]$ and $[T_i^*T]$ are precomputed only once before the `while`-loop. In this case the `ALLREDUCE` command could be omitted. This parallelization would essentially require the storage of the matrix $[T_i^*T]$, which is of the size $N/P \times N$. The complexity of the algorithm is $\mathcal{O}(dN^2/P)$. Thus, the alternative parallelization scheme is more efficient with respect to the one presented in Algorithm 15 in terms of storage and complexity if $2m > N$. However, since we consider compressed sensing problems, we assume $m \ll N$ and therefore the alternative parallelization scheme is not of importance for the remainder of this presentation. More details on complexity are given further below.

Sequential acceleration techniques It is well-known that ISTA is converging slowly in general and needs a large number of iterations [127, 22]. The parallel Algorithm 15 is one way to speed it up. However, several sequential methods that decrease the running time of ISTA were proposed. The most important modifications are FISTA (in combination with backtracking) [13], or the acceleration through an adaptive choice of the stepsize $t = t^{(n)}$ in Step 3. A thorough overview on those techniques is given in the recent field guide towards forward-backward splitting methods in [91]. Since, in general, such methods already tremendously reduce the number of iterations of ISTA, it is very likely that an accelerated sequential version of ISTA outperforms the parallel version. In particular the huge amount of iterations reduces the efficiency of the parallelization since the communication in the `ALLREDUCE` command is necessary in each iteration. Thus, in order to further increase the performance of such an accelerated sequential ISTA, we are facing the question whether the above presented sequential acceleration techniques (FISTA and adaptive stepsize) can be also efficiently parallelized.

Parallelization of acceleration techniques The method FISTA, which reached a certain popularity due to its good worst-case performance, is extending ISTA by a componentwise Nesterov-type prediction step of the iterates $u_i^{(n)}$. Therefore, its native parallelization, which we present in Algorithm 16, does not require any additional communication apart from the unavoidable `ALLREDUCE` command. In [154] it was first formulated and shown to be scalable on distributed systems.

If we fix the stepsize t, FISTA is only guaranteed to converge for $t \leq \|T\|^{-2}$. In the original paper [13], the authors proposed an acceleration of ISTA and FISTA by

[3] A `BROADCAST` command distributes the value of a certain variable from one core to all the other cores.

Algorithm 16 Parallel FISTA (P-FISTA)

Set $\chi_i^{(0)} := 0$, $[T\chi^{(0)}] := 0$, $s^{(0)} := 1$, $0 < t < \|T\|^{-2}$.

1: **while** stopping criterion is not fulfilled **do**
2: **for** $i = 1, \dots, P$ **do in parallel**
3: $u_i^{(n+1)} = \mathbb{S}_{t\lambda}(\chi_i^{(n)} + tT_i^*(y - [T\chi^{(n)}]))$
4: $s^{(n)} = \frac{1+\sqrt{1+4(s^{(n-1)})^2}}{2}$
5: $w^{(n)} = 1 + \frac{s^{(n-1)}-1}{s^{(n)}}$
6: $\chi_i^{(n+1)} = w^{(n)} u_i^{(n+1)} + \left(1 - w^{(n)}\right) u_i^{(n)}$
7: **end for**
8: $[T\chi^{(n+1)}] = \sum_{i=1}^{P} T_i \chi_i^{(n+1)}$ by `ALLREDUCE`
9: **end while**

backtracking, and proved its convergence. The idea of backtracking is to choose t several orders of magnitude above $\|T\|^{-2}$, and to iteratively decrease the value of t and to evaluate Step 3, until the backtracking condition,

$$\begin{aligned} &\|Tu^{(n+1)} - y\|_{\mathcal{H}}^2 \\ &\leq \|T\chi^{(n)} - y\|_{\mathcal{H}}^2 + 2\langle u^{(n+1)} - \chi^{(n)}, T^*(T\chi^{(n)} - y)\rangle + \frac{1}{t}\|u^{(n+1)} - \chi^{(n)}\|_{\ell_2(\Lambda)}^2, \end{aligned} \tag{4.66}$$

is fulfilled for some t'. Then the variable stepsize $t = t^{(n)}$ is set to t'. While the iterative evaluation of Step 3, and the decrease of t can be done in parallel, the evaluation of the backtracking condition (4.66) requires again an `ALLREDUCE` operation in each backtracking step since one has to compute and synchronize $Tu^{(n+1)}$. We present this method in Algorithm 17. Note that the last two summands of the right-hand side of (4.66) can be parallelized as well since we can rewrite them as

$$\sum_{i=1}^{P} 2\langle u_i^{(n+1)} - u_i^{(n)}, T_i^*([T\chi^{(n)}] - y)\rangle + \frac{1}{t}\|u_i^{(n+1)} - \chi_i^{(n)}\|_{\ell_2(\Lambda_i)}^2.$$

Because we have to compute the sum and synchronize the results, there is another `ALLREDUCE` command hidden in the evaluation of (4.66). In an implementation, it can be combined with the `ALLREDUCE` command in Step 5. An important change in Algorithm 17 is that the computation of $[T\chi^{(n+1)}]$ is handled without an additional matrix-vector multiplication and `ALLREDUCE` command, as it was still necessary in Step 8 of Algorithm 16. Now, it can be efficiently updated via the already computed $[Tu^{(n+1)}]$. Note that the backtracking stops after a finite number of steps in each iteration. We comment on this issue further below in Section 4.2.2.

Algorithm 17 P-FISTA with backtracking

Set $\chi_i^{(0)} := 0$, $[T\chi^{(0)}] := 0$, $s^{(0)} := 1$, $t_0 = C\|T\|^{-2}$, $C > 1$, $0 < \eta < 1$.

1: **while** stopping criterion is not fulfilled **do**
2: **for** $i = 1, \dots, P$ **do in parallel**
3: $t = t_0$
4: $u_i^{(n+1)} = \mathbb{S}_{t\lambda}(\chi_i^{(n)} + tT_i^*(y - [T\chi^{(n)}]))$
5: $[Tu^{(n+1)}] = \sum_{i=1}^{P} T_i u_i^{(n+1)}$ by `ALLREDUCE`
6: **if** (4.66) is false **then**
7: $t = \eta t$, goto 4
8: **end if**
9: $s^{(n)} = \frac{1+\sqrt{1+4(s^{(n-1)})^2}}{2}$
10: $w^{(n)} = 1 + \frac{s^{(n-1)}-1}{s^{(n)}}$
11: $\chi_i^{(n+1)} = w^{(n)} u_i^{(n+1)} + \left(1 - w^{(n)}\right) \chi_i^{(n)}$
12: $[T\chi^{(n+1)}] = w^{(n)}[Tu^{(n+1)}] + \left(1 - w^{(n)}\right) [T\chi^{(n)}]$
13: **end for**
14: **end while**

In the recent paper [53], the authors propose Algorithm 18 as an acceleration of the parallel ISTA. Instead of using backtracking in order to determine a (common) stepsize t, in this algorithm, a stepsize t_i is chosen independently for each node $i = 1, \dots, P$ by means of a Barzilai-Borwein step (see [53, Section 4.1]). Furthermore, convergence is obtained by using an Armijo step with stepsize $w^{(n)}$, in the direction $u^{(n+1)} - \chi^{(n)}$. The Armijo backtracking condition is satisfied if

$$\begin{aligned} \|w^{(n)}[Tu^{(n+1)}] + \left(1 - w^{(n)}\right) [T\chi^{(n)}] - y\|_{\mathcal{H}}^2 \\ \leq \|[T\chi^{(n)}] - y\|_{\mathcal{H}}^2 + \sigma w^{(n)} \langle u^{(n+1)} - \chi^{(n)}, T^*(T\chi^{(n)} - y)\rangle, \end{aligned} \tag{4.67}$$

for some parameter $0 < \sigma < 1$. Again, the iterative evaluation of condition (4.67) in Step 7 can be parallelized but needs additional communication.

Domain decomposition frameworks for ISTA While in Algorithm 15, 16, and 17, existing sequential algorithms are natively parallelized by a standard distribution of the work on several cores, in Algorithm 18 a *divide and conquer* principle is used since one applies different stepsizes $t_i^{(n)}$ and performs an individual step on every core. This is an essentially different strategy. It is closely related to a so called *subspace correction* approach towards the parallelization of ISTA, which goes back to the idea of *domain decomposition* in the paper [74], where Algorithm 19 was proposed as a

Algorithm 18 Parallel ISTA with Armijo line-search

Set $\chi_i^{(0)} := 0$, $[T\chi^{(0)}] := 0$.

while stopping criterion is not fulfilled **do**
  for $i = 1, \ldots, P$ **do in parallel**
    determine Barzilai-Borwein stepsize $t_i^{(n+1)}$
    $u_i^{(n+1)} = \mathbb{S}_{t_i^{(n+1)}\lambda}(\chi_i^{(n)} + t_i^{(n+1)} T_i^*(y - [T\chi^{(n)}]))$
  end for
  $[Tu^{(n+1)}] = \sum_{i=1}^{P} T_i u_i^{(n+1)}$ by `ALLREDUCE`
  compute stepsize $w^{(n)} > 0$, satisfying the Armijo backtracking condition (4.67)
  for $i = 1, \ldots, P$ **do in parallel**
    $\chi_i^{(n+1)} = w^{(n)} u_i^{(n+1)} + \left(1 - w^{(n)}\right) \chi_i^{(n)}$
    $[T\chi^{(n+1)}] = w^{(n)} [Tu^{(n+1)}] + \left(1 - w^{(n)}\right) [T\chi^{(n)}]$
  end for
end while

parallel acceleration technique of the sequential ISTA. The novelty in this approach is the execution of multiple independent inner iterations on a subspace. Note that the constant prediction stepsize $w = \frac{1}{P+1}$ is necessary for convergence if $L > 1$. However, in the case that $L = 1$, it can be shown that the algorithm also converges with $w = 1$, and thus it is equivalent to the parallel ISTA, as presented in Algorithm 15. From this point of view, Algorithm 19 can be seen as a more general framework and the origin of Algorithm 15 and 16, although it is not mentioned in [154], which appeared several years later.

An accelerated parallel ISTA with multiple independent inner iterations While the idea of domain decomposition with independent stepsizes was taken into consideration in [53], the idea of multiple inner iterations without additional synchronization among the cores was not pursued further. However, from the following rough estimation of the computational complexity of Algorithm 19 we see that the usage of multiple inner steps may be advantageous for large P. We can assume that the computation of the thresholding operator and the Armijo backtracking steps are of low computational cost compared to application of the operator T_i. Furthermore, we consider an operator T that is finite dimensional and can be represented as a full real valued matrix of size $m \times N$. Thus its application is a simple matrix-vector ($d = 1$) or matrix-matrix ($d > 1$) product. In Algorithm 15 to 18 one has to form explicitely the product $T_i \chi_i^{(n+1)}$ or $T_i u_i^{(n+1)}$ since its result is communicated to all other cores. We assume to have the data equally distributed to the P cores, and consider the operators T_i as submatrices of the

Algorithm 19 Domain decomposition ISTA

Set $\chi_i^{(0)} := 0$, $[T\chi^{(0)}] := 0$, $1 \leq L \in \mathbb{N}$, $t \leq \|T\|^{-2}$, $w = \frac{1}{P+1}$.

while stopping criterion is not fulfilled **do**
  for $i = 1, \ldots, P$ **do in parallel**
    $u_i^{(n+1,0)} = \chi_i^{(n)}$
    $y_i^{(n+1)} = y - [T\chi^{(n)}] + T_i\chi_i^{(n)}$
    for $\ell = 0, \ldots, L-1$ **do**
      $u_i^{(n+1,\ell+1)} = \mathbb{S}_{t\lambda}(u_i^{(n+1,\ell)} + tT_i^*(y_i^{(n+1)} - T_i u_i^{(n+1,\ell)}))$
    end for
  end for
  $[Tu^{(n+1)}] = \sum_{i=1}^{P} T_i u_i^{(n+1,L)}$ by `ALLREDUCE`
  for $i = 1, \ldots, P$ **do in parallel**
    $\chi_i^{(n+1)} = wu_i^{(n+1,L)} + (1-w)\chi_i^{(n)}$
    $[T\chi^{(n+1)}] = w[Tu^{(n+1)}] + (1-w)[T\chi^{(n)}]$
  end for
end while

operator T, then this operation has a complexity of $\mathcal{O}(mdN/P)$ on a parallel system. Moreover, one has to multiply with T_i^* in the gradient step, which has again a cost of mdN/P multiplications. Thus, the algorithms that do not exploit inner iterations have a cost of $\mathcal{O}(I_{\text{single}}mdN/P)$, where I_{single} is the number of iterations until a stopping criterion is fulfilled.

In an inner step of Algorithm 19, we have to compute $T_i^*(T_i u_i^{(n+1,\ell)})$. At first sight, this seems to be the same operation as in the description above, but since one is not required to communicate the result of $T_i u_i^{(n+1,\ell)}$, one has the option to precompute $\bar{T}_i := T_i^* T_i$, and replace the above operation by $\bar{T}_i u_i^{(n+1,\ell)}$ in each inner iteration. This multiplication has a complexity of $\mathcal{O}((N/P)^2 d)$. The reader recognizes that this is not necessarily the better choice since $(N/P)^2 d < 2mdN/P$ if and only if $P > N/(2m)$. However, in high performance computing we usually have hundreds to thousands of cores available and thus, this condition is not too restrictive. For instance, if $m = N/10$, P has to be greater than 5 if we want the multiplication with $\bar{T}_i$ to be more efficient than multiplying subsequently with T_i and T_i^*. Furthermore, this comparison reveals that the cost of the inner iterations shrinks quadratically with the number of used cores. In this respect, the method that we present here is expected to perform better, the more it is parallelized. Despite of this observation, also in Algorithm 19 one has to communicate the result of $T_i u_i^{(n+1,L)}$ once at the end of each series of inner iterations. By means of code optimization, and reuse of results[4], one passage of

[4]Since one already computes $[T_i u_i^{(n+1,L)}] = [T_i u_i^{(n+1)}]$ before the communication step, one can

the inner `while` loop with L inner iterations requires $2mdN/P + (L-1)(N/P)^2 d$ multiplications. If the algorithm needs I_{multiple} iterations, then it has a total cost of $\mathcal{O}(I_{\text{multiple}}(2mdN/p+(L-1)(N/P)^2 d))$. The reasonable hope is that the inner iterations replace outer iterations (with respect to Algorithms 15–18), such that $I_{\text{multiple}} L \approx I_{\text{single}}$, leading to a performance increase due to the usage of the inner iterations. This hope turns out to be the crucial point, since the parallelization very likely leads to a "loss of information" due to the reduced communication. This can be illustrated by the following thought experiment: Let χ^* be the minimizer of the functional (2.15) and let us assume that on cores $2, \dots, P$ this minimizer is achieved. If core 1 is provided from the other cores with $y_1^{(n+1)} = y - \sum_{\tau \neq 1} T_\tau \chi_\tau^*$ (compare Step 4), then of course y_1 contains the best information core 1 can get and the approximation error $\|\chi_i^{(n)} - \chi^*\|_{\ell_2}$ is only produced from the iterations on the domain of core 1. But χ^* is not known and thus the approximation error on each core is propagated as a prior to each other core and has larger impact the larger L is chosen. Thus, it would not be a surprise if eventually $I_{\text{multiple}} L > I_{\text{single}}$.

To summarize these two issues, we note that the increase of the parameter L is on the one hand responsible for a gain in computational time due to efficiently performed inner iterations, and on the other hand it will imply a larger number of outer iterations because one needs more corrective communication in order to reach a certain stopping criterion. One of the main investigations in this section is to answer the question, whether there is an optimal $L > 1$ such that one can outperform Algorithms 15–18. In particular in Section 4.2.5, we will return to this question in the scope of numerical simulations.

Since in Algorithm 19 we only consider so far fixed stepsizes (as in the plain ISTA), it is very likely that the number of iterations I_{multiple} would be relatively high in comparison to the accelerated Algorithms 16–18, i.e., $I_{\text{multiple}} \gg I_{\text{single}}$. In order to tremendously decrease I_{multiple} by several techniques like, e.g., backtracking on subdomains, we propose a modified framework for an accelerated parallel domain decomposition ISTA in the following sections. We prove convergence of the algorithm in Section 4.2.3 and show by numerical results in Section 4.2.5 that the use of this algorithm is advantageous in terms of the parallel runtime for a large number of cores P.

4.2.2 An Accelerated Domain Decomposition ISTA

Algorithm 19 was a powerful acceleration of ISTA by the time of its publication. However, since it needs a large number of iterations, it is definitively outperformed by the methods that are presented in the previous section. One reason for the huge

update $[T_i \chi_i^{(n+1)}] = w[T_i u_i^{(n+1)}] + (1-w)[T_i \chi_i^{(n)}]$ without any additional matrix multiplication. This saves one matrix multiplication in the gradient step of the succeeding first inner iteration.

number of iterations is the fixed stpesize $t < \|T\|^{-2}$. The other even more severe drawback is the fixed prediction stepsize $w = \frac{1}{P+1}$. The more one parallelizes, the smaller w gets, neutralizing the advantages from the parallelization. Nevertheless, a thorough analysis of the convergence proof of Algorithm 19 reveals a certain flexibility towards diverse acceleration strategies, namely

1. a backtracking as in Algorithm 17, but independently performed on each subdomain, such that different stepsizes t are possible on different cores and no communication via `ALLREDUCE` is necessary;

2. a line-search for an optimal adaptive prediction parameter $w^{(n)}$, such that the rather conservative value $w = \frac{1}{P+1}$ is released and larger prediction steps are possible in Step 11 and 12;

3. an adaptive update of a newly introduced stepsize $t^{(n)}$ (depending on the current iterate) which serves as a pre-estimate of the backtracking stepsize t (point 1), allowing for a reduced number of backtracking steps.

Hereafter, we want to introduce these techniques in detail. Since we use an adaptively chosen stepsize (point 3), the purpose of the backtracking (point 1) is the correction of a stepsize which was over-estimated. In the proof of convergence point 1 guarantees the global convergence of the algorithm. In order to properly formulate the backtracking on a subdomain (Λ_i), we generalize the definition of the backtracking condition (4.66):

Definition 4.27
For arbitrary $i \in \{1, \dots, P\}$, let $a, b \in \ell_2(\Lambda_i)$, and $c \in \mathcal{H}$. The *backtracking condition* $\mathcal{B}_t(a, b, c)$ is true for $t > 0$ if and only if

$$\|T_i a - c\|_{\mathcal{H}}^2 \leq \|T_i b - c\|_{\mathcal{H}}^2 + 2\langle a - b, T_i^*(T_i b - c)\rangle + \frac{1}{t}\|a - b\|_{\ell_2(\Lambda_i)}^2.$$

4.2.2.1 Domain Decomposition ISTA with Backtracking, Adaptive Stepsize, and Prediction Step

By Definition 4.27 and according to the points 1,2, and 3, we formulate the domain decomposition ISTA with backtracking, adaptive stepsize, and prediction step in Algorithm 20. It consists of an outer loop (Step 1 to 21) and an inner loop (Step 6 to 12). The outer loop stops if a respective stopping criterion is fulfilled. We comment on such criteria in Section 4.2.4.1. In the inner loop, the actual work is done, thus, independently and in parallel we perform on the subdomains a predefined number $L^{(n)}$ of Bregman steps with possible backtracking (Steps 8 to 10). Note that the backtracking is a loop with an unknown number of iterations. However this loop stops after a finite number of steps as we will show in the next paragraph. After the parallel work, back in the outer loop the P results are synchronized in Step 14. Furthermore

Algorithm 20 Domain decomposition ISTA with backtracking, adaptive stepsize, and prediction step

Set $\chi_i^{(0)} := 0$, $[T\chi^{(n)}] := 0$, $0 < \eta < 1$, $w_{\max} > w_c := \frac{1}{P+1}$, $1 \leq L_{\max} \in \mathbb{N}$, $t^{(0)} = \|T\|^{-2}$.

1: **while** stopping criterion is not fulfilled **do**
2: choose the number of inner iterations $L^{(n)} \in \mathbb{N}$, $1 \leq L^{(n)} \leq L_{\max}$
3: **for** $i = 1, \ldots, P$ **do in parallel**
4: $u_i^{(n+1,0)} = \chi_i^{(n)}$
5: $y_i^{(n+1)} = y - [T\chi^{(n)}] + T_i\chi_i^{(n)}$
6: **for** $\ell = 0, \ldots, L^{(n)} - 1$ **do**
7: $t_i^{(n,\ell+1)} = t^{(n)}$
8: $u_i^{(n+1,\ell+1)} = \mathbb{S}_{t_i^{(n,\ell+1)}\lambda}(u_i^{(n+1,\ell)} + t_i^{(n,\ell+1)}T_i^*(y_i^{(n+1)} - T_iu_i^{(n+1,\ell)}))$
9: **if** $\mathcal{B}_{t_i^{(n,\ell+1)}}(u_i^{(n+1,\ell+1)}, u_i^{(n+1,\ell)}, y_i^{(n+1)})$ is false **then**
10: $t_i^{(n,\ell+1)} = \eta t_i^{(n,\ell+1)}$, goto 8
11: **end if**
12: **end for**
13: **end for**
14: $[Tu^{(n+1)}] = \sum_{i=1}^{P} T_iu_i^{(n+1,L^{(n)})}$ by `ALLREDUCE`
15: choose prediction stepsize $w^{(n+1)} \in \operatorname{arg\,min}_{0<w\leq w_{\max}} \mathcal{J}_\lambda(wu^{(n+1)} + (1-w)\chi^{(n)})$
16: **for** $i = 1, \ldots, P$ **do in parallel**
17: $\chi_i^{(n+1)} = w^{(n+1)}u_i^{(n+1,L^{(n)})} + (1 - w^{(n+1)})\chi_i^{(n)}$
18: $[T\chi^{(n+1)}] = w^{(n+1)}[Tu^{(n+1)}] + (1 - w^{(n+1)})[T\chi^{(n)}]$
19: **end for**
20: update $t^{(n+1)}$
21: **end while**

a prediction step is performed (Step 15). The prediction is not only supposed to accelerate the algorithm by finding along the computed direction an iterate such that the functional value is decreased, but it also ensures convergence (see the proof of Theorem 4.30). Eventually the initial backtracking stepsize $t^{(n)}$ is (adaptively) updated.

The convergence of the algorithm can be shown without further specifying in detail, how $t^{(n)}$ and $L^{(n)}$ are chosen. On the one hand, the fact that Steps 2, 15, and 20 are flexibly tunable is an advantage of Algorithm 20. On the other hand, such flexibility can be a curse for a user with low experience. Therefore, we present in Section 4.2.4 a practical guide on how to implement these steps.

Also the independent backtracking cuts both ways. It allows on the one hand to determine an optimal stepsize for each subproblem, and thus for larger steps on that domain, on the other hand one backtracking step needs basically the cost of one inner iteration. Since the number of backtracking steps can differ from core to core, this may lead to the idling of some of the cores. Therefore a reduced overall performance is expected due to the loss of synchronization.

4.2.2.2 Backtracking with a Finite Number of Steps

As already mentioned further above, the backtracking loop (Steps 8 to 10) stops after a finite number of steps, which implies that the backtracking condition is always fulfilled at the end of an inner iteration. This assertion is verified by a well known result in the field of soft thresholding and forward-backward splitting algorithms: The backtracking condition $\mathcal{B}_t(a,b,c)$ is always fulfilled if $t \leq \|T\|^{-2}$. It becomes obvious by the equivalent transformation

$$\begin{aligned}
&\mathcal{B}_t(a,b,c) \text{ is true} \\
\iff &\frac{1}{t}\|a-b\|_{\ell_2(\Lambda_i)}^2 \geq \|T_i a - c\|_{\mathcal{H}}^2 - \|T_i b - c\|_{\mathcal{H}}^2 - 2\langle a-b, T_i^*(T_i b - c)\rangle \\
\iff &\frac{1}{t}\|a-b\|_{\ell_2(\Lambda_i)}^2 \geq \|T_i a - c\|_{\mathcal{H}}^2 - \|T_i b - c + (T_i a - T_i b)\|_{\mathcal{H}}^2 + \|T_i a - T_i b\|_{\mathcal{H}}^2 \\
\iff &\frac{1}{t}\|a-b\|_{\ell_2(\Lambda_i)}^2 \geq \|T_i a - T_i b\|_{\mathcal{H}}^2,
\end{aligned}$$

where the last statement definitely holds for $t \leq \|T\|^{-2}$ since

$$\frac{1}{t}\|a-b\|_{\ell_2(\Lambda_i)}^2 \geq \|T\|^2\|a-b\|_{\ell_2(\Lambda_i)}^2 \geq \|T_i\|^2\|a-b\|_{\ell_2(\Lambda_i)}^2 \geq \|T_i a - T_i b\|_{\mathcal{H}}^2.$$

4.2.2.3 Surrogate Function and Thresholding Operator

It is a standard result (see, e.g., [47]) that the thresholding operator $\mathbb{S}_{t\lambda}$ is equivalent to the minimization of a surrogate function. This is in general a helpful tool in order to prove convergence of such thresholding algorithms. Therefore, we define

$$\mathcal{J}_t^S(u,a) := t\|y - Tu\|_{\mathcal{H}}^2 + t\lambda\|u\|_{\ell_1(\Lambda)} + \|u-a\|_{\ell_2(\Lambda)}^2 - t\|Tu - Ta\|_{\mathcal{H}}^2.$$

Transforming the right-hand side, we obtain

$$\mathcal{J}_t^S(u,a) = \|(a + tT^*(y - Ta)) - u\|_{\ell_2(\Lambda)}^2 + t\lambda\|u\|_{\ell_1(\Lambda)} + \Xi,$$

where Ξ is independent of u. Thus, $\mathcal{J}_t^S$ is strictly convex with respect to the first component u and it has a unique minimizer, once the second component a is fixed.

The subdifferential (see Section 2.2.2) of $\mathcal{J}_t^S$ with respect to the first component is given by

$$\begin{aligned}\partial_u \mathcal{J}_t^S(u,a) &= -2(a - tT^*(y - Ta)) + 2u + t\lambda\partial\|\cdot\|_{\ell_1(\Lambda)}(u) \\ &= \{\xi \in \ell_\infty(\Lambda) | \xi_j \in (-2(a - tT^*(y - Ta)) + 2u)_j + t\lambda\partial\|\cdot\|_{\ell_2}(u_j),\, j \in \Lambda\}\,.\end{aligned}$$

Then, e.g., [47, Proposition 2.1], yields

$$\operatorname*{arg\,min}_{u\in\ell_2(\Lambda)} \mathcal{J}_t^S(u,a) = \mathbb{S}_{t\lambda}(a + tT^*(y - Ta)).$$

One obtains the same relations and definitions for each subdomain by replacing $T \to T_i$, $y \to y_i$, and $\Lambda \to \Lambda_i$.

4.2.3 Convergence Results

In order to facilitate the cross-reading, we want to redefine some quantities close to the notation in [74]. We define for each $i = 1, \ldots, P$ the *extension operator* $E_i \colon \ell_2(\Lambda_i) \to \ell_2(\Lambda)$, $(E_i\nu)_j = \nu_j$, if $j \in \Lambda_i$, $(E_i\nu)_j = 0$, otherwise. The *restriction operator*, which is the adjoint operator of the extension operator, is denoted by $R_i := E_i^*$. By these definitions, we have $T_i = TE_i$ and $T_i^* = R_iT^*$.

In the following, we prove the convergence of Algorithm 20. Since it is a generalization of [74, Algorithm (29)], we refer at some points in the proof to arguments in that paper. We reuse those results and calculations, in order to focus here on the necessary modifications that we made. We begin with an auxiliary lemma. It shows an elementary relation of two successive inner iterates. The result is equivalent to a special case of [13, Lemma 2.3]. We reformulate it in our particular notation and repeat the necessary steps from its proof for the sake of completeness and consistency.

Lemma 4.28
Let $u_i^{(n+1,\ell+1)}$, $u_i^{(n+1,\ell)}$, and $y_i^{(n+1)}$ be defined as in Algorithm 20, and $s_i^{(n+1)} := \chi^{(n)} - \chi_i^{(n)} = \sum_{\tau\neq i} E_\tau R_\tau \chi^{(n)}$ for arbitrary $i \in \{1, \ldots, P\}$. If the backtracking condition $\mathcal{B}_t(u_i^{(n+1,\ell+1)}, u_i^{(n+1,\ell)}, y_i^{(n+1)})$ is true for some $t > 0$, then

$$\begin{aligned}\mathcal{J}_\lambda(E_iu_i^{(n+1,\ell)} + s_i^{(n+1)}) - \mathcal{J}_\lambda(E_iu_i^{(n+1,\ell+1)} + s_i^{(n+1)}) \\ \geq \frac{1}{t}\|u_i^{(n+1,\ell+1)} - u_i^{(n+1,\ell)}\|_{\ell_2(\Lambda_i)}.\end{aligned} \tag{4.68}$$

Proof. By definition we have $y_i^{(n+1)} = y - Ts_i^{(n+1)}$. We use the surrogate functional of Section 4.2.2.3 to rewrite Step 8 as

$$\begin{aligned}
u_i^{(n+1,\ell+1)} &= \mathbb{S}_{t\lambda}(u_i^{(n+1,\ell)} + tT_i^*(y_i^{(n+1)} - T_i u_i^{(n+1,\ell)})) \\
&= R_i \mathbb{S}_{t\lambda}(E_i u_i^{(n+1,\ell)} + tT^*(y - T(s_i^{(n+1)} - E_i u_i^{(n+1,\ell)}))) \\
&= R_i \mathbb{S}_{t\lambda}(E_i u_i^{(n+1,\ell)} + s_i^{(n+1)} + tT^*(y - T(s_i^{(n+1)} - E_i u_i^{(n+1,\ell)}))) \\
&= \operatorname*{arg\,min}_{u_i \in \ell_2(\Lambda_i)} \mathcal{J}_t^S(E_i u_i, E_i u_i^{(n+1,\ell)} + s_i^{(n+1)}) \\
&= \operatorname*{arg\,min}_{u_i \in \ell_2(\Lambda_i)} \mathcal{J}_t^S(E_i u_i + s_i^{(n+1)}, E_i u_i^{(n+1,\ell)} + s_i^{(n+1)}),
\end{aligned}$$

where we added s_i^{n+1} in the second and fifth identity since its components on Λ_i are 0. Moreover, by the definition of $\mathcal{J}_\lambda$ and $y_i^{(n+1)}$, and the observation that $\|E_i u + s_i^{(n+1)}\|_{\ell_1(\Lambda)} = \|u\|_{\ell_1(\Lambda_i)} + \|s_i^{(n+1)}\|_{\ell_1(\Lambda)}$ for $u \in \ell_1(\Lambda_i)$, we obtain

$$\begin{aligned}
&\mathcal{J}(E_i u_i^{(n+1,\ell)} + s_i^{(n+1)}) - \mathcal{J}_\lambda(E_i u_i^{(n+1,\ell+1)} + s_i^{(n+1)}) \\
=&\|TE_i u_i^{(n+1,\ell)} - y_i^{(n+1)}\|_{\mathcal{H}}^2 - \|TE_i u_i^{(n+1,\ell+1)} - y_i^{(n+1)}\|_{\mathcal{H}}^2 \\
+&\lambda\|u_i^{(n+1,\ell)}\|_{\ell_1(\Lambda_i)} - \lambda\|u_i^{(n+1,\ell+1)}\|_{\ell_1(\Lambda_i)}.
\end{aligned} \tag{4.69}$$

According to the optimality condition

$$0 \in \partial_u \mathcal{J}_t^S(E_i u_i^{(n+1,\ell+1)} + s_i^{(n+1)}, E_i u_i^{(n+1,\ell)} + s_i^{(n+1)}), \tag{4.70}$$

there exists $\xi \in \partial\|\cdot\|_{\ell_1(\Lambda_i)}(u_i^{(n+1,\ell+1)})$, such that

$$2R_i T^*(TE_i u_i^{(n+1,\ell)} - y_i^{(n+1)}) + \lambda\xi = \frac{2}{t}(u_i^{(n+1,\ell)} - u_i^{(n+1,\ell+1)}). \tag{4.71}$$

Furthermore, the definition of the subdifferential (2.16) yields

$$\|u_i^{(n+1,\ell)}\|_{\ell_1(\Lambda_i)} - \|u_i^{(n+1,\ell+1)}\|_{\ell_1(\Lambda_i)} \geq \langle u_i^{(n+1,\ell)} - u_i^{(n+1,\ell+1)}, \xi\rangle. \tag{4.72}$$

We estimate the first two summands of the right-hand side in (4.69) by the backtracking condition, and the last two summands by (4.72), and use (4.71) in order to replace ξ.

This yields

$$\begin{aligned}
&\mathcal{J}_\lambda(E_i u_i^{(n+1,\ell)} + s_i^{(n+1)}) - \mathcal{J}_\lambda(E_i u_i^{(n+1,\ell+1)} + s_i^{(n+1)}) \\
\geq& - \langle u_i^{(n+1,\ell+1)} - u_i^{(n+1,\ell)}, 2R_i T^*(TE_i u_i^{(n+1,\ell)} - y_i^{(n+1)})\rangle \\
&\quad - \frac{1}{t}\|u_i^{(n+1,\ell+1)} - u_i^{(n+1,\ell)}\|^2_{\ell_2(\Lambda_i)} + \lambda\langle u_i^{(n+1,\ell)} - u_i^{(n+1,\ell+1)}, \xi\rangle \\
=& - \langle u_i^{(n+1,\ell+1)} - u_i^{(n+1,\ell)}, 2R_i T^*(TE_i u_i^{(n+1,\ell)} - y_i^{(n+1)}) + \lambda\xi\rangle \\
&\quad - \frac{1}{t}\|u_i^{(n+1,\ell+1)} - u_i^{(n+1,\ell)}\|^2_{\ell_2(\Lambda_i)} \\
=& - \langle u_i^{(n+1,\ell+1)} - u_i^{(n+1,\ell)}, \frac{2}{t}(u_i^{(n+1,\ell)} - u_i^{(n+1,\ell+1)})\rangle - \frac{1}{t}\|u_i^{(n+1,\ell+1)} - u_i^{(n+1,\ell)}\|^2_{\ell_2(\Lambda_i)} \\
=& \frac{1}{t}\|u_i^{(n+1,\ell+1)} - u_i^{(n+1,\ell)}\|^2_{\ell_2(\Lambda_i)}.
\end{aligned}$$

□

Remark 4.29
An immediate consequence from Lemma 4.28 is the monotonicity property for the inner iterations,

$$\mathcal{J}_\lambda(E_i u_i^{(n+1,\ell)} + s_i^{(n+1)}) \geq \mathcal{J}_\lambda(E_i u_i^{(n+1,\ell+1)} + s_i^{(n+1)}), \quad \ell = 0, \dots, L^{(n)} - 1.$$

Theorem 4.30 (Weak Convergence)
Algorithm 20 produces a sequence $(\chi^{(n)})_{n\in\mathbb{N}}$ in $\ell_2(\Lambda)$ whose weak accumulation points are minimizers of the functional $\mathcal{J}_\lambda$. In particular, the set of weak accumulation points is non-empty.

Proof. We first observe that in each inner iteration $\ell = 0, \dots, L^{(n)}-1$, the backtracking condition is fulfilled by construction since $t_i^{(n,\ell+1)}$ is decreased until $\mathcal{B}_{t_i^{(n,\ell+1)}}, (u_i^{(n+1,\ell+1)} u_i^{(n+1,\ell)}, y_i^{(n+1)})$ is true, and therefore Lemma 4.28 applies. In the presentation of this proof, we want to denote by $t_i^{(n,\ell+1)}$ exclusively the final valid backtracking stepsize in the ℓ-th inner iteration. In particular, we have $t_i^{(n,\ell+1)} \leq t^{(n)}$. Summing up the inequalities (4.68) for the inner iterations $\ell = 0, \dots, L^{(n)} - 1$, we obtain a telescopic sum on the left-hand side and a sum on the right-hand side, and thus, we have

$$\begin{aligned}
&\mathcal{J}_\lambda(E_i u_i^{(n+1,0)} + s_i^{(n+1)}) - \mathcal{J}_\lambda(E_i u_i^{(n+1,L^{(n)})} + s_i^{(n+1)}) \\
&\qquad \geq \frac{1}{t^{(n)}} \sum_{\ell=0}^{L^{(n)}-1} \|u_i^{(n+1,\ell+1)} - u_i^{(n+1,\ell)}\|^2_{\ell_2(\Lambda_i)},
\end{aligned} \tag{4.73}$$

for all $i = 1, \dots, P$. Note that by definition $\chi^{(n)} = E_i u_i^{(n+1,0)} + s_i^{(n+1)}$ the first summand on the left-hand side of (4.73) is equal to $\mathcal{J}_\lambda(\chi^{(n)})$. Then, the monotonicity

$$\mathcal{J}_\lambda(\chi^{(n)}) \geq \mathcal{J}_\lambda(E_i u_i^{(n+1,L^{(n)})} + s_i^{(n+1)}) \tag{4.74}$$

follows immediately.
Next, we need to show that the set of minimizers in Step 15 is not empty. By a convexity argument, which was already used in [74], we show below that

$$\mathcal{J}_\lambda(w_c u^{(n+1)} + (1 - w_c)\chi^{(n)}) \leq \mathcal{J}_\lambda(\chi^{(n)}) \tag{4.75}$$

is always true for $0 < w_c = 1/(P+1)$. Since $w_c < w_{\max}$ by definition, one concludes again by the convexity of $\mathcal{J}_\lambda$ that there is at least one strictly positive minimizer in the interval $]0, w_{\max}]$. This implies that Step 15 is well-defined.

Let us recall the convexity argument that shows (4.75): Since $\mathcal{J}_\lambda$ is a convex functional, we have

$$\frac{\left(\mathcal{J}_\lambda(\chi^{(n)}) + \sum_{i=1}^{P} \mathcal{J}_\lambda(E_i u_i^{(n+1,L^{(n)})} + s_i^{(n+1)})\right)}{P+1} \geq \mathcal{J}_\lambda\left(\frac{\chi^{(n)} + \sum_{i=1}^{P}(E_i u_i^{(n+1,L^{(n)})} + s_i^{(n+1)})}{P+1}\right).$$

By the above relation, the definitions $u^{(n+1)} = \sum_{i=1}^{P} E_i u_i^{(n+1,L^{(n)})}$ and $s_i^{(n+1)} = \chi^{(n)} - \chi_i^{(n)}$, and the monotonicity property (4.74) we obtain

$$\begin{aligned}
\mathcal{J}_\lambda(\chi^{(n)}) &= \frac{1}{P+1}\left(\mathcal{J}_\lambda(\chi^{(n)}) + P\mathcal{J}_\lambda(\chi^{(n)})\right) \\
&\geq \frac{1}{P+1}\left(\mathcal{J}_\lambda(\chi^{(n)}) + \sum_{i=1}^{P} \mathcal{J}_\lambda(E_i u_i^{(n+1,L^{(n)})} + s_i^{(n+1)})\right) \\
&\geq \mathcal{J}_\lambda\left(\frac{\chi^{(n)} + \sum_{i=1}^{P}(E_i u_i^{(n+1,L^{(n)})} + s_i^{(n+1)})}{P+1}\right) \\
&= \mathcal{J}_\lambda\left(\frac{1}{P+1}\left(P\chi^{(n)} + u^{(n+1)}\right)\right) \\
&= \mathcal{J}_\lambda\left(w_c u^{(n+1)} + (1 - w_c)\chi^{(n)}\right),
\end{aligned} \tag{4.76}$$

where we used in the last equality the definition $w_c := \frac{1}{P+1}$. Thus, we showed inequality (4.75).
Recall that we obtain $w^{(n+1)}$ from Step 15. By its minimum property and $\chi^{(n+1)} = w^{(n+1)}u^{(n+1)} + (1 - w^{(n+1)})\chi^{(n)}$ (compare Step 17), we have

$$\mathcal{J}_\lambda\left(w_c u^{(n+1)} + (1 - w_c)\chi^{(n)}\right) \geq \mathcal{J}_\lambda(\chi^{(n+1)}) \tag{4.77}$$

Combining (4.76) and (4.77), and plugging in (4.73), we obtain

$$\mathcal{J}_\lambda(\chi^{(n+1)}) \leq \frac{\left((P+1)\mathcal{J}_\lambda(\chi^{(n)}) - \sum_{i=1}^{P} \frac{1}{t^{(n)}} \sum_{\ell=0}^{L^{(n)}-1} \|u_i^{(n+1,\ell+1)} - u_i^{(n+1,\ell)}\|_{\ell_2(\Lambda_i)}^2\right)}{P+1},$$

and thus

$$\mathcal{J}_\lambda(\chi^{(n)}) - \mathcal{J}_\lambda(\chi^{(n+1)}) \geq \frac{1}{t^{(n)}(P+1)} \sum_{i=1}^{P} \sum_{\ell=0}^{L^{(n)}-1} \|u_i^{(n+1,\ell+1)} - u_i^{(n+1,\ell)}\|_{\ell_2(\Lambda_i)}^2. \quad (4.78)$$

The sequence $(\mathcal{J}_\lambda(\chi^{(n)}))_{n\in\mathbb{N}}$ is convergent, since it is monotonically decreasing and bounded from below by 0. Moreover, we assume that $t^{(n)}$ is bounded above by a positive bound $t_{\max} < \infty$. Thus, we have by (4.78) that

$$\sum_{i=1}^{P} \sum_{\ell=0}^{L^{(n)}-1} \|u_i^{(n+1,\ell+1)} - u_i^{(n+1,\ell)}\|_{\ell_2(\Lambda_i)}^2 \to 0, \quad n \to \infty. \quad (4.79)$$

Since $\lambda\|\chi^{(n)}\|_{\ell_2(\Lambda)} \leq \lambda\|\chi^{(n)}\|_{\ell_1(\Lambda)} \leq \mathcal{J}_\lambda(\chi^{(n)}) \leq \mathcal{J}_\lambda(\chi^{(0)})$, the sequence $(\chi^{(n)})_{n\in\mathbb{N}}$ is uniformly bounded in $\ell_2(\Lambda)$, and thus, there exists a weakly convergent subsequence $(\chi^{(n_k)})_{k\in\mathbb{N}}$, whose weak limit is denoted by $\chi^{(\infty)}$. For simplicity, we rename this subsequence again $(\chi^{(n)})_{n\in\mathbb{N}}$. Moreover, using the well-known root-mean-square arithmetic-mean inequality $\sqrt{\frac{1}{r}\left(\sum_{k=1}^{r} z_k^2\right)} \geq \frac{1}{r}\left(\sum_{k=1}^{r} z_k\right)$, $z \in \mathbb{R}_+^r$, the triangle inequality, as well as the convexity of $\|\cdot\|_{\ell_2(\Lambda)}^2$, we obtain

$$\begin{aligned}
\sum_{i=1}^{P} \sum_{\ell=0}^{L^{(n)}-1} \|u_i^{(n+1,\ell+1)} - u_i^{(n+1,\ell)}\|_{\ell_2(\Lambda_i)}^2 &\geq \sum_{i=1}^{P} \frac{1}{L^{(n)}} \left(\sum_{\ell=0}^{L^{(n)}-1} \|u_i^{(n+1,\ell+1)} - u_i^{(n+1,\ell)}\|_{\ell_2(\Lambda_i)}\right)^2 \\
&\geq \sum_{i=1}^{P} \frac{1}{L^{(n)}} \|E_i u_i^{(n+1,L^{(n)})} - E_i u_i^{(n+1,0)}\|_{\ell_2(\Lambda)}^2 \\
&= \frac{1}{PL^{(n)}} \|\sum_{i=1}^{P} E_i u_i^{(n+1,L^{(n)})} - E_i \chi_i^{(n)}\|_{\ell_2(\Lambda)}^2 \\
&= \frac{1}{PL^{(n)}} \|u^{(n+1)} - \chi^{(n)}\|_{\ell_2(\Lambda)}^2 \\
&= \frac{1}{PL^{(n)} w^{(n+1)}} \|\chi^{(n+1)} - \chi^{(n)}\|_{\ell_2(\Lambda)}^2 \\
&\geq \frac{1}{PL_{\max} w_{\max}} \|\chi^{(n+1)} - \chi^{(n)}\|_{\ell_2(\Lambda)}^2,
\end{aligned}$$

where the third identity follows from $\chi^{(n+1)} - \chi^{(n)} = w^{(n+1)}(u^{(n+1)} - \chi^{(n)})$ (Step 17). Consequently,

$$\|\chi^{(n+1)} - \chi^{(n)}\|_{\ell_2(\Lambda)} \to 0, \quad \|u^{(n+1)} - \chi^{(n)}\|_{\ell_2(\Lambda)} \to 0, \quad n \to \infty, \tag{4.80}$$

and the sequence $(u^{(n+1)})_{n\in\mathbb{N}}$ is also weakly converging with weak limit $u^{(\infty)} = \chi^{(\infty)}$. Recall that weak convergence implies componentwise convergence, also for $T^*Tu^{(n)}$ and $T^*T\chi^{(n)}$. From the optimality condition (4.70) for $\ell = L^{(n)} - 1$, we obtain the componentwise relation

$$\begin{aligned} 0 \in &\left(-2(u_i^{(n+1,L^{(n)}-1)} + t_i^{(n,L^{(n)})} R_i T^*((y - T\sum_{\tau\neq i} E_\tau \chi_\tau^{(n)}) - TE_i u_i^{(n+1,L^{(n)}-1)})) \right)_j \\ &+ \left(2u_i^{(n+1,L^{(n)})}\right)_j + t_i^{(n,L^{(n)})} \lambda \partial \|\cdot\|_{\ell_2}((u_i^{(n+1,L^{(n)})})_j), \quad j \in \Lambda_i, \quad i = 1,\dots,P. \end{aligned}$$

Since the backtracking condition is fulfilled definitely for $t^{(n)} \leq \|T\|^{-2}$ (compare Section 4.2.2.2), there exists $t_{\min} := \eta\|T\|^{-2} > 0$ such that $0 < t_{\min} \leq t^{(n)} \leq t_{\max} < \infty$. We conclude that also $t_i^{(n,L^{(n)})}$ is bounded below and above by positive constants. Therefore, we can divide by $t_i^{(n,L^{(n)})}$ and obtain

$$\begin{aligned} 0 \in &\left(-2R_i T^*((y - T\sum_{\tau\neq i} E_\tau \chi_\tau^{(n)}) - TE_i u_i^{(n+1,L^{(n)}-1)}) \right)_j \\ &+ \frac{\left(2u_i^{(n+1,L^{(n)})} - 2u_i^{(n+1,L^{(n)}-1)}\right)_j}{t_i^{(n,L^{(n)})}} + \lambda\partial\|\cdot\|_{\ell_2}((u_i^{(n+1,L^{(n)})})_j). \end{aligned}$$

Eventually, we let $n \to \infty$, and conclude by the boundedness of $t_i^{(n,L^{(n)})}$, (4.80), (4.79), and the fact that the (component-wise) subdifferential $\partial\|\cdot\|_{\ell_2}$ is an outer semicontinuous set-valued function (compare [74, Lemma 3.2 and Definition 1]) that

$$\begin{aligned} 0 \in &\left(2R_i T^*((y - T\sum_{\tau\neq i} E_\tau R_\tau \chi^{(\infty)}) - TE_i R_i u^{(\infty)})) \right)_j + \lambda\partial\|\cdot\|_{\ell_2}((E_i R_i u^{(\infty)})_j), \\ &j \in \Lambda_i, \quad i = 1,\dots,P, \end{aligned}$$

and thus

$$0 \in \partial_u \mathcal{J}_1^S(\chi^{(\infty)}, \chi^{(\infty)})$$

since $u^{(\infty)} = \chi^{(\infty)}$. Note that this result does not depend on the stepsize any more. From [47, Proposition 2.1] it follows that

$$\chi^{(\infty)} = \mathbb{S}_\lambda(\chi^{(\infty)} + T^*(y - T\chi^{(\infty)})).$$

The minimality of $\chi^{(\infty)}$ for the functional J_λ then follows by [47, Proposition 3.10]. The above argumentation is valid for any subsequence and any weak accumulation point of the original sequence $(\chi^{(n)})_{n\in\mathbb{N}}$. □

In a practical setting, we will always have a finite dimensional problem, since we are working on a computer with limited memory. In this case, the weak convergence of the algorithm also implies strong convergence. Nevertheless, it may be of theoretical interest, to also show strong convergence in the infinite dimensional setting. For the case that one chooses a fixed stepsize $t = t^{(n)} \leq \|T\|^{-2}$, the proof of the strong convergence is conducted by the same arguments as in the proof of [74, Theorem 4.2].

Theorem 4.31 (Strong Convergence)
If a fixed stepsize $t = t^{(n)} < \|T\|^{-2}$ is chosen, Algorithm 20 produces a sequence $(\chi^{(n)})_{n\in\mathbb{N}}$ in $\ell_2(\Lambda)$ whose strong accumulation points are minimizers of the functional $\mathcal{J}_\lambda$. In particular, the set of strong accumulation points is non-empty.

Any additional investigation concerning the strong convergence for arbitrary stepsizes, which are determined by backtracking, are not considered here, and left as an open problem. The proof in [74, Theorem 4.2] does not work since one does not have $\|I - tT^*T\| \leq 1$ any more. Instead, the use of the backtracking condition is required. Together with the usage of an arbitrary stepsize, such a proof is supposed to be very technical.

4.2.4 Implementation Details

In the formulation of Algorithm 20 we left open how $t^{(n)}$, $L^{(n)}$, and $w^{(n)}$ are updated. Therefore, we catch up on a detailed explanation on how we implement the Steps 2, 15, 20, and we clarify which stopping criterion is used in the algorithm.

4.2.4.1 A Fair Stopping Criterion

Since we want to compare Algorithm 20 towards other state-of-the-art algorithms, we need a stopping criterion in order to provide a fair comparison. According to our remarks in the end of Section 4.2.1, this concerns in particular the number of iterations that an algorithm needs. Furthermore, a stopping criterion should also work in practice. In [154, 53], the authors use the criterion

$$\frac{\|u^{(n)} - u^*\|_{\ell_2(\Lambda)}}{\|u^*\|_{\ell_2(\Lambda)}},$$

where u^* is the minimizer of $\mathcal{J}_\lambda$. While this is a fair criterion for a comparison since the iterate of any method is compared to the same minimizer, it is impracticable since one does certainly not dispose of u^* in practice because its computation is the actual

task of the algorithm. Another frequently used stopping criterion, which was also used in [74], is the relative difference of the functional in two successive iterations, i.e.,

$$\frac{|\mathcal{J}_\lambda(u^{(n+1)}) - \mathcal{J}_\lambda(u^{(n)})|}{\mathcal{J}_\lambda(u^{(n)})}.$$

While it does not need any information about the minimizer, it is not necessarily fair since it can lead to wrong conclusions about the performance of an algorithm. The reason is that its value is highly depending on the actual size of the step that the algorithm takes in the current iteration. Due to the continuity of the functional $\mathcal{J}_\lambda$, the smaller the difference $\|u^{(n+1)} - u^{(n)}\|_{\ell_2(\Lambda)}$, the smaller is the value of this stopping criterion. In this sense, an algorithm which may be of poor performance and progresses in very small steps, may stop much earlier than an efficient algorithm with a large stepsize.

In our numerical investigations, we use the penalization of the first order optimality conditions since it provides a strong statement on the quality of an iterate, and it does neither depend on the previous iterate, nor on the minimizer χ^*. Thus, we define the stopping criterion residual as

$$\begin{aligned}
&\tilde{r}^{(n)} := \|\xi^{(n)}\|_{\ell_2(\Lambda)}, \\
&\xi_j^{(n)} := \begin{cases} \left(2T^*(Tu^{(n)} - y)\right)_j + \lambda \frac{u_j^{(n)}}{\|u_j^{(n)}\|_{\ell_2}} & \|u_j^{(n)}\|_{\ell_2} > 0, \\ \max\left(0, \left\|\left(2T^*(Tu^{(n)} - y)\right)_j\right\|_{\ell_2} - \lambda\right) & \text{otherwise,} \end{cases} \qquad j \in \Lambda.
\end{aligned}$$

We stop the algorithm as soon as the normalized stopping criterion

$$r^{(n)} := \frac{\tilde{r}^{(n)}}{\|2T^*y\|_{\ell_2(\Lambda)}} < \mathrm{tol}_r \tag{4.81}$$

is fulfilled for a predefined tolerance tol_r. Note that the above formulation is suited to be parallelized. In particular, its computational cost is marginal, since the multiplication $[T^*Tu^{(n)}]$ has to be computed anyway and can be reused. Only the composition of $\|\xi^{(n)}\|_{\ell_2(\Lambda)}$ needs another `ALLREDUCE` command.

4.2.4.2 An Adaptive Choice of the Number of Inner Iterations $L^{(n)}$

The number of inner iterations $L^{(n)}$ is the parameter which is mainly responsible for the tradeoff between the gain of computational time due to efficiently performed inner iterations, and the loss due to the error propagation. This issue is extensively described in the paragraph *An accelerated parallel ISTA with multiple independent inner iterations* in Section 4.2.1. In our algorithmic scheme, the user is free to play

with this variable. However, it makes sense to start with a small value of $L^{(n)}$, and increase it with the iteration number n. The simple reason is that one is far away from the solution in the beginning and the iterates are a very erroneous estimate of the minimizer. This error is distributed to the other cores and the error is added up in multiple independent inner iterations. Thus, it is better to synchronize the parallel steps more often in order to correct the miscomputed components. As soon as the algorithm computes iterates close to the solution, the error propagation has a lower impact and synchronization is required less often. Then one can make use of multiple efficient inner iterations. In our implementation, we set $L^{(1)} := 1$, and use the following logarithmic update rule,

$$L^{(n+1)} := \begin{cases} 1 & r^{(n)} > \gamma \operatorname{tol}_r, \\ \max\left(L^{(n)}, \min\left(L_{\max}, \lfloor 2 + \log_{10}(\frac{\gamma \operatorname{tol}_r}{r^{(n)}}) \frac{L_{\max}-1}{\log_{10}(\gamma)} \rfloor\right)\right) & \text{otherwise,} \end{cases}$$

where tol_r is the given tolerance for the stopping criterion, $r^{(n)}$ is the stopping criterion residual (see (4.81)), and $\gamma > 0$ is the parameter which determines how early the algorithm switches from a single inner iteration to multiple inner iterations. The above update rule makes sure that the sequence $(L^{(n)})_{n\in\mathbb{N}}$ is increasing, and $1 \leq L^{(n)} \leq L_{\max}$.

4.2.4.3 Update Strategies for the Stepsize $t^{(n)}$

An important feature of the algorithm is an estimation of the stepsize $t^{(n+1)}$. Again, it is left to the user to try different strategies. Whatever strategy one chooses, by the backtracking in Steps 8–10 it is made sure that the stepsize is small enough in order to guarantee convergence. A non-adaptive strategy would be to set $t^{(n)} := C\|A\|^{-2}$, with $C > 1$ chosen such that $t^{(n)}$ is several orders of magnitude above the stepsize $\|A\|^{-2}$ for which the backtracking condition is guaranteed to hold (compare Section 4.2.2). Such a static rule has the advantage of no additional computational effort in Step 20. However, depending on the choice of C, it is very likely that a lot of backtracking steps (and therefore additional substantial work) are necessary in order to correct such a non-adaptive stepsize. A good adaptive stepsize depends on the current iterate and the particular shape of the objective functional in its environment. Therefore, we use the adaptive Barzilai-Borwein (BB) update rule [201]

$$t^{(n+1)} := \begin{cases} t_m & 2t_m > t_s, \\ t_s - \frac{1}{2}t_m & \text{otherwise,} \end{cases}$$

in our implementation. This rule was shown to be very efficient in the recent investigation [91]. It uses the *steepest descent* stepsize

$$t_s := \frac{\|\chi^{(n+1)} - \chi^{(n)}\|^2_{\ell_2(\Lambda)}}{\langle \chi^{(n+1)} - \chi^{(n)}, T^*T(\chi^{(n+1)} - \chi^{(n)})\rangle},$$

and the *minimum residual* stepsize

$$t_m := \frac{\langle \chi^{(n+1)} - \chi^{(n)}, T^*T(\chi^{(n+1)} - \chi^{(n)}) \rangle}{\|T^*T(\chi^{(n+1)} - \chi^{(n)})\|^2_{\ell_2(\Lambda)}},$$

which result from a quadratic approximation of $\|T\chi - y\|^2_{\ell_2(\Lambda)}$ in the current iterate. A more detailed explanation can be found in [201, 91, 73]. If the updated stepsize is not positive it is reset to the previous value. Again, the computation of this update rule does not require any additional multiplications since it reuses already computed quantities.

4.2.4.4 Choice of the Prediction Stepsize $w^{(n+1)}$

We remind that the problem

$$\operatorname*{arg\,min}_{0<w\leq w_{\max}} \mathcal{J}_\lambda(wu^{(n+1)} + (1-w)\chi^{(n)}) \tag{4.82}$$

is well defined (within the scope of Algorithm 20), and the set of minimizers is non-empty. This was shown in the proof of Theorem 4.30. A solution to problem (4.82) allows a maximum decrease of the objective functional in the given direction. However, giving a closer look to the proof of Theorem 4.30, we also notice that it is not necessary to find the minimizer but to simply replace Step 15 by "choose prediction stepsize $w^{(n+1)} \in \{0 < w \leq w_{\max} | \mathcal{J}_\lambda(wu^{(n+1)} + (1-w)\chi^{(n)}) \leq \mathcal{J}_\lambda(\frac{1}{P+1}u^{(n+1)} + (1-\frac{1}{P+1})\chi^{(n)})\}$". Thus, this step allows a lot of flexibility in the implementation. In particular, an approximate solution of (4.82) is sufficient for convergence (as long as it fulfills the above condition). Together with the fact that the minimization is only done in one dimension, the effort to perform this step is small compared to the overall complexity of the algorithm, which is dominated by the complexity of the application of the operator T (matrix-matrix multiplication, $\mathcal{O}(mNd)$). We suggest two possible means for the approximate computation of the minimizer. Define

$$\mathcal{J}_\lambda(w) := \theta_1 w^2 + \theta_2 w + \lambda \sum_{j\in\Lambda} \|w\theta_{3j} + \theta_{4j}\|_{\ell_2}, \tag{4.83}$$

with $\theta_1 := \|T(u^{(n+1)} - \chi^{(n)})\|^2_{\ell_2(\Lambda)}$, $\theta_2 := 2\langle T(u^{(n+1)} - \chi^{(n)}), T\chi^{(n)} \rangle$, $\theta_3 := u^{(n+1)} - \chi^{(n)}$, and $\theta_4 = \chi^{(n)}$. Note that the calculation of θ_1, and θ_2 is very cheap since the products of T and the iterates are a byproduct of the algorithm. A function evaluation of (4.83) needs $\mathcal{O}(Nd)$ multiplications. By omitting constant terms, a simple calculation shows that (4.82) is equivalent to

$$\operatorname*{arg\,min}_{0<w\leq w_{\max}} \mathcal{J}_\lambda(w). \tag{4.84}$$

Since problem (4.84) is continuous but not smooth, it can be solved by means of the Nelder-Mead Simplex Algorithm [143], which needs one function evaluation per iteration and therefore $\mathcal{O}([\text{iterations of Nelder-Mead}] \cdot Nd)$ multiplications. Since we are considering large-scale problems, we will have in general

$$[\text{iterations of Nelder-Mead}] \ll m,$$

and thus a comparably small amount of computational effort. However, it is well-known that this algorithm is in general not efficient. To further reduce the computational cost of this step, we present the following alternative method to compute a minimizer of (4.84): First, sort the set $\{w_j | \theta_{3j} \neq 0 \text{ and } \exists w_j \colon w_j\theta_{3j} + \theta_{4j} = 0, \text{ and } w_{\max} \geq w_j > 0\}$ and obtain $w_{j_1}, \ldots, w_{j_{\tilde{N}}}$, with $\tilde{N} \leq N$ and $w_{j_\tau} \leq w_{j_{\tau+1}}$. With this, we computed the relevant positions where the function $\mathcal{J}_\lambda(w)$ is non-smooth. The effort of this operation is $\mathcal{O}(N \log N)$. Then, we compute an element $\tau_{\min}$ of

$$\operatorname*{arg\,min}_{\tau=1,\ldots,\tilde{N}} \mathcal{J}_\lambda(w_{j_\tau})$$

within $\mathcal{O}(Nd \log N)$ since we have to evaluate the function $\log N$ times for a search on a sorted array. Then we define $j_{\min} = j_{\tau_{\min}}$. Since we know that $\mathcal{J}_\lambda(w)$ is differentiable on the intervals $]w_{j_{\min}-1}, w_{j_{\min}}[$, and $]w_{j_{\min}}, w_{j_{\min}+1}[$, we can determine a minimizer on this interval from a gradient step method, which is known to converge in very few iterations. Each iteration of such a gradient step will require again the evaluation of the function and its derivative. Thus, we have a total cost of $\mathcal{O}(Nd(\log N + [\text{iterations of Gradient-Descent}]))$. Since in general

$$\log N + [\text{iterations of Gradient-Descent}] \ll [\text{iterations of Nelder-Mead}] \ll m,$$

this method is further decreasing the computational effort for the determination of the prediction stepsize. Note that the above description is only a sketch. In the implementation one has to consider some special cases, e.g., if there are no points in $]0, w_{\max}]$, where the function is non-smooth.

Since the evaluation of the functional $\mathcal{J}_\lambda(w)$ and its derivative can be parallelized easily, the cost of both methods shrinks with the number of cores.

4.2.5 Simulations

Recall that the formulation of Algorithm 20 allows a creative implementation of Steps 2, 15, and 20 independent from our suggestions in Section 4.2.4, as long as it stays in certain bounds as, e.g., $w < w_{\max}$ with $w_{\max} > w_c$. We make use of this flexibility in order to define the following two instances of the algorithm:

- **DDLISTA:** Do not choose an adaptive stepsize, but set $t^{(n)} := \|T\|^{-2}$. Moreover, do not perform backtracking. The algorithm nevertheless converges, since $t^{(n)}$ always fulfills the backtracking condition and thus the inner iterations decrease the functional. Perform a line search in Step 15.
- **DDLBISTA:** Choose an adaptive BB stepsize $t^{(n)}$ (compare Section 4.2.4), turn backtracking on, and perform a line search in Step 15.

The algorithm DDLISTA does not use backtracking steps, and thus, it has the advantage that each core is doing approximately the same work, and the disadvantage that the stepsize is fixed. The algorithm DDLBISTA uses an adaptive stepsize and thus does individual steps, however due to the backtracking correction, efficiency may be lost due to idling.

After defining the test setting in Section 4.2.5.1, we investigate in Section 4.2.5.2 in the question if the introduction of multiple inner iterations leads to a shorter running time of DDLISTA and DDLBISTA respectively. In this context we clarify for which maximum number of inner iterations $L_{\max}$, we obtain the best performance. We then fix this parameter for each algorithm respectively and compare its performance to the state-of-the-art solvers P-FISTA and PSCL in Section 4.2.5.3.

4.2.5.1 Test Setting

We define a "small", a medium, and a large test setting:

	Setting A	**Setting B**	**Setting C**
N	2048	16384	131072
m	256	2048	16384
k	32	256	2048

Furthermore, we fix the number of channels $d = 10$. The matrices $T \in \mathbb{R}^{m\times N}$ are randomly generated normalized Gaussian-type matrices and we generate the measurement data $y \in \mathbb{R}^{m\times d}$ from a random sparse multi-channel signal $x \in \mathbb{R}^{N\times d}$ with entries $x_i \sim \mathcal{N}(0,1)$ and $\#\operatorname{supp}(x) = k$, via $y = Tx + e$, where the entries of the noise $e \in \mathbb{R}^{m\times d}$ are randomly chosen such that $e_i \sim \mathcal{N}(0, 0.01)$. The algorithm parameters are set to $\eta = 0.5$, and $w_{\max} = 100$.

All tests are performed for $P = 1, 2, 4, 8, 16, 32, 64, 128$ cores and we create a trivial decomposition of our index sets $\Lambda = \{1, \dots, N\}$ into P subsets $\Lambda_i = \{(i-1)N/P + 1, \dots, iN/P\}$.

In order to make the problem free of dimension, we choose the regularization parameter $\lambda = 0.01\|T^*y\|_{\ell_\infty(\Lambda)}$. We use the normalized stopping criterion (4.81) with

$\text{tol}_r = 5\text{E-}6$ for a fair comparison among all methods.

Moreover, we produced optimized code for the special case of $P = 1$. In particular one does not need communication via the `ALLREDUCE` command. This optimized single core version allows a fair comparison among the parallelized and non-parallelized implementation. Parallelization in general requires a computational and memory overhead, which can lead to the practical observation that a parallelization with a very small number of cores (e.g. $P = 2$) is less efficient than its non-parallelized counterpart.

All algorithms are implemented by `C++` in combination with the linear algebra library `Eigen` [65], and compiled with the `Intel MPI library 5.0`. The tests were executed on the Haswell Nodes of the SuperMUC Petascale System Phase 2, situated in Garching (Munich), Germany, with the support of the Leibniz Rechenzentrum (LRZ). The x86-based SuperMUC was ranking 24th (Phase 2) in the Top500 List as of November 2015 [174]. It is built out of 18432 Intel Xeon E5-2680 8C processors running at up to 2.7 GHz, which sums up to a total of 147,456 cores [128]. One compute node consists of two sockets, each equipped with eight cores. A single node can access 32 GiB of main memory, resulting in 2 GiB per core and 288 TiB for the entire machine.

4.2.5.2 Comparison for Different Values of $L_{\max}$

The answer to the question, whether it makes sense or not to use more than one inner iteration, and thus $L_{\max} > 1$, is twofold. This is the outcome of the numerical results that we present in this section. We performed for DDLISTA and DDLBISTA ten tests with randomly generated datasets of Setting A, and with different levels of parallelization, according to the description in Section 4.2.5.1. For each value of P and $L_{\max}$, we computed the mean of the number of iterations and the total parallel computation time, and present the results in Figure 4.10. We comment the four rows of subfigures from the top to the bottom.

- **iterations vs cores:** Note that the number of iterations is not the number of the used outer iterations, but the sum of the used inner iterations until the stopping of the algorithm, thus $\sum_n L^{(n)}$. For fixed P, the number of iterations increases with increasing $L_{\max}$ for both algorithms, as we predicted it in Section (4.2.1). Moreover, in DDLISTA for $P \geq 8$, the number of iterations stays nearly constant. In particular the number of iterations is nearly constant for all P, if $L_{\max} \in \{1, 2\}$. In DDLBISTA, the number of iterations jumps when P changes from 1 to 2, but it stays constant afterwards. Having a constant number of iterations with increasing P is advantageous since then the effect of the parallelization is not neutralized by a higher number of iterations.
- **time vs cores:** In DDLBISTA, additional inner iterations have a negative

impact on the performance. With increasing $L_{\max}$ the times get worse for fixed P, whereas in DDLISTA, the algorithm benefits from $L_{\max} > 1$, the best results are obtained for the parameter $L_{\max} = 2$. Due to the code optimization and the fact that no communication is necessary, in general, the results are better in the case of $P = 1$ than for $P = 2$.

- **time vs cores (perc.):** In order to see, how much one gains with respect to the parameter setting $L_{\max} = 1$, we express on a percentage basis the results of the previous subfigure in terms of the quotient time/time($L_{\max} = 1$). In both algorithms, we observe that the larger we choose P the more the algorithms profit from a parameter $L_{\max} > 1$. This confirms our expectation that efficiently performed multiple inner iterations increase the performance, as formulated at the end of Section 4.2.1.

- **comm. time vs cores (perc.):** There is no vital difference in the communication time for different $L_{\max}$. Thus, we can conclude that the performance increase for $L_{\max} = 2$ is mainly caused by the efficient evaluation of the matrix-vector product in the inner iterations, and not by a more economic use of communication.

Summarizing the above observations, one can speedup DDLISTA by inner iterations, while the use of additional inner iterations in DDLBISTA only has a negative effect on the performance. Obviously a different number of backtracking steps on different cores produces a non equilibrated work load among the cores. We performed the same experiment on Setting B and C, observing the same behavior. Thus, in the following tests, we fix $L_{\max} = 2$ for DDLISTA and $L_{\max} = 1$ for DDLBISTA.

4.2.5.3 Comparison to State-of-the-Art Solvers

We compare our results to the state-of-the-art solvers P-FISTA and PSCL that we described in Section 4.2.1. Since, we do not consider overlapping subdomains, we use the non-overlapping version PSCLN (Algorithm 18). In our tests, P-FISTA worked much better with backtracking (Algorithm 17) than without (Algorithm 16). It was observed as well in the numerical investigations of [53]. Therefore, we only consider P-FISTA with backtracking. We do not consider G-Rock [154] since it is in general not ensured to converge and we are only interested in algorithms that are fully reliable. According to the results in Section 4.2.5.2, we compare those algorithms to DDLISTA (with $L_{\max} = 2$) and DDLBISTA (with $L_{\max} = 1$). Again, we performed ten tests with randomly generated datasets of Setting A, B, and C, and with different levels of parallelization, according to the description in Section 4.2.5.1. We present the results for each setting respectively in the columns of Figure 4.11 and comment hereafter each row of subfigures from the top to the bottom. For Setting C, we were not able to compute a result for $P \in \{1, 2\}$ since the data was too large to fit in the memory of less than $P = 4$ cores.

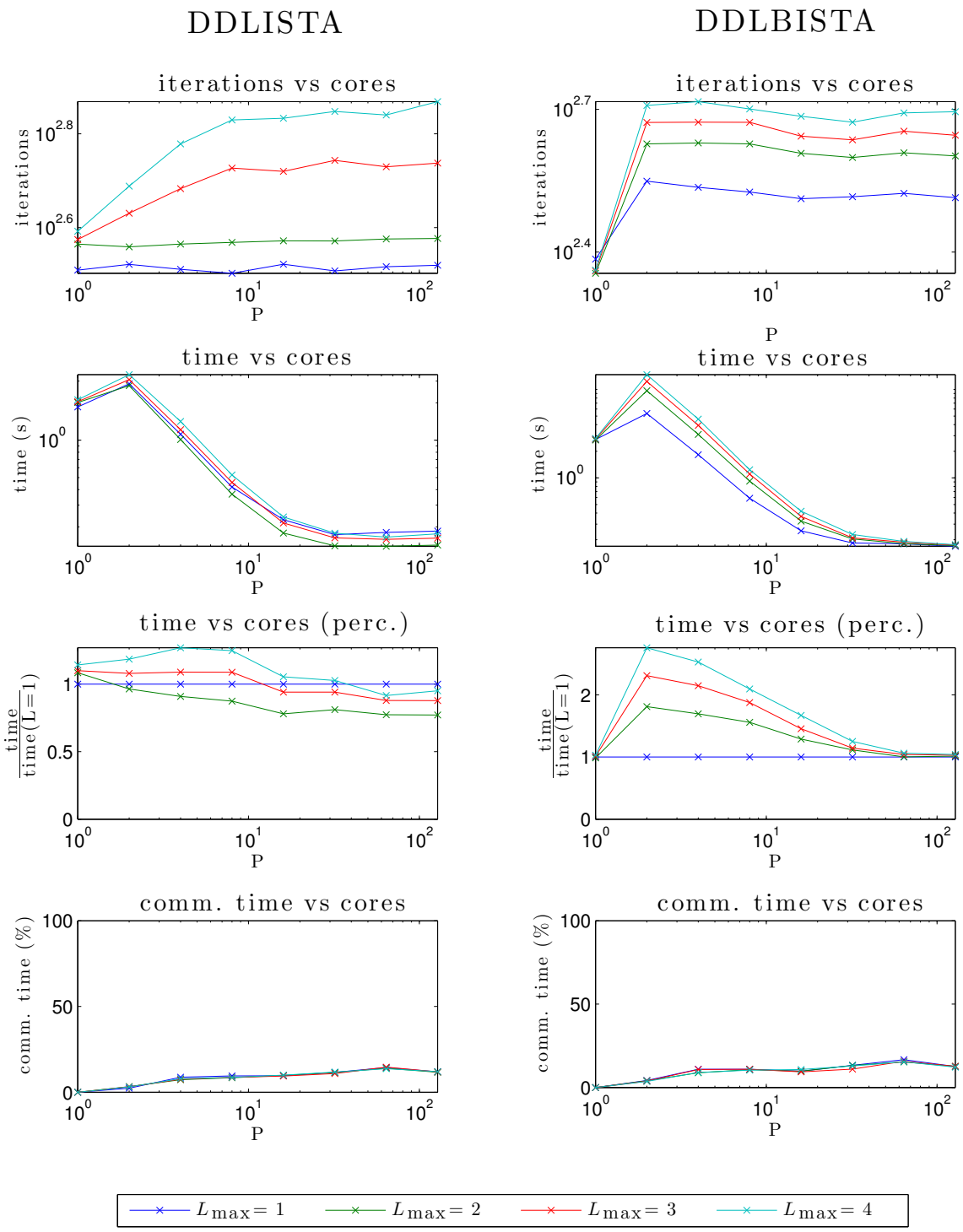


Figure 4.10: Parallelization results for DDLISTA (left column) and DDLBISTA (right column) in Setting A with parameters $L_{\max} = 1, \ldots, 4$.

- **iterations vs cores:** Since P-FISTA is a native parallelization, it has naturally a constant number of iterations. The number of iterations for PSCLN is slightly increasing with P. DDLBISTA needs less iterations than DDLISTA.

- **time vs cores:** Again, we observe that the result for $P = 1$ is better than for $P = 2$ (except for P-FISTA) due to the efficient implementation of all algorithms. We furthermore observe that all methods outperform P-FISTA in the range of $P > 4$. One exception is Setting A, where FISTA overtakes all methods for $P = 128$. The simple reason is that the dimensions in this problem are relatively small such that the computation of the prediction steps in PSCLN, DDLISTA, and DDLBISTA becomes an important overhead when P gets too large. In general, although it uses more iterations, DDLISTA is faster than DDLBISTA. While PSCLN is the fastest method for smaller P, DDLBISTA and DDLISTA are the fastest methods for larger P.

- **time vs cores (perc.):** Here, we express on a percentage basis the results of the previous subfigure in terms of the quotient time/time(P-FISTA). Dependent on the cores that one has at disposal, one can get to more than 50% decrease of the computational time.

- **parallel efficiency vs cores (perc.):** As explained in the introduction of Section 4.2, the parallel efficiency (4.65) indicates whether or not one should use a parallelized algorithm in the case that one has to solve many problems of the same type. Obviously, in Setting A and B, where the matrix still fits in the memory of a single core, the parallel efficiency stays below 1. This means that one solves one problem per core, if one has to solve a lot of them. However, at the level of very large dimensions (Setting C), where we are forced to parallelize due to memory shortage, we have to redefine the parallel efficiency via the smallest possible P, in this setting
$$\text{eff}(P) := \frac{4\theta(4)}{P\,\theta(P)}.$$
Then all methods reach a parallel efficiency greater than one, and it is even increasing with P. This means a super-linear speedup. If one wants to solve many of those problems, it is appropriate to solve them on a large number of cores P.

- **comm. time vs cores (perc.):** In percentage, the communication time of all methods is approximately equal, except for the Setting A. The proportion of communication in P-FISTA and also PSCLN is higher than for DDLISTA and DDLBISTA for large P.

The main outcome of those experiments is that DDLISTA and DDLBISTA exploit their full potential for larger values of P. In particular in the very large-scale Setting

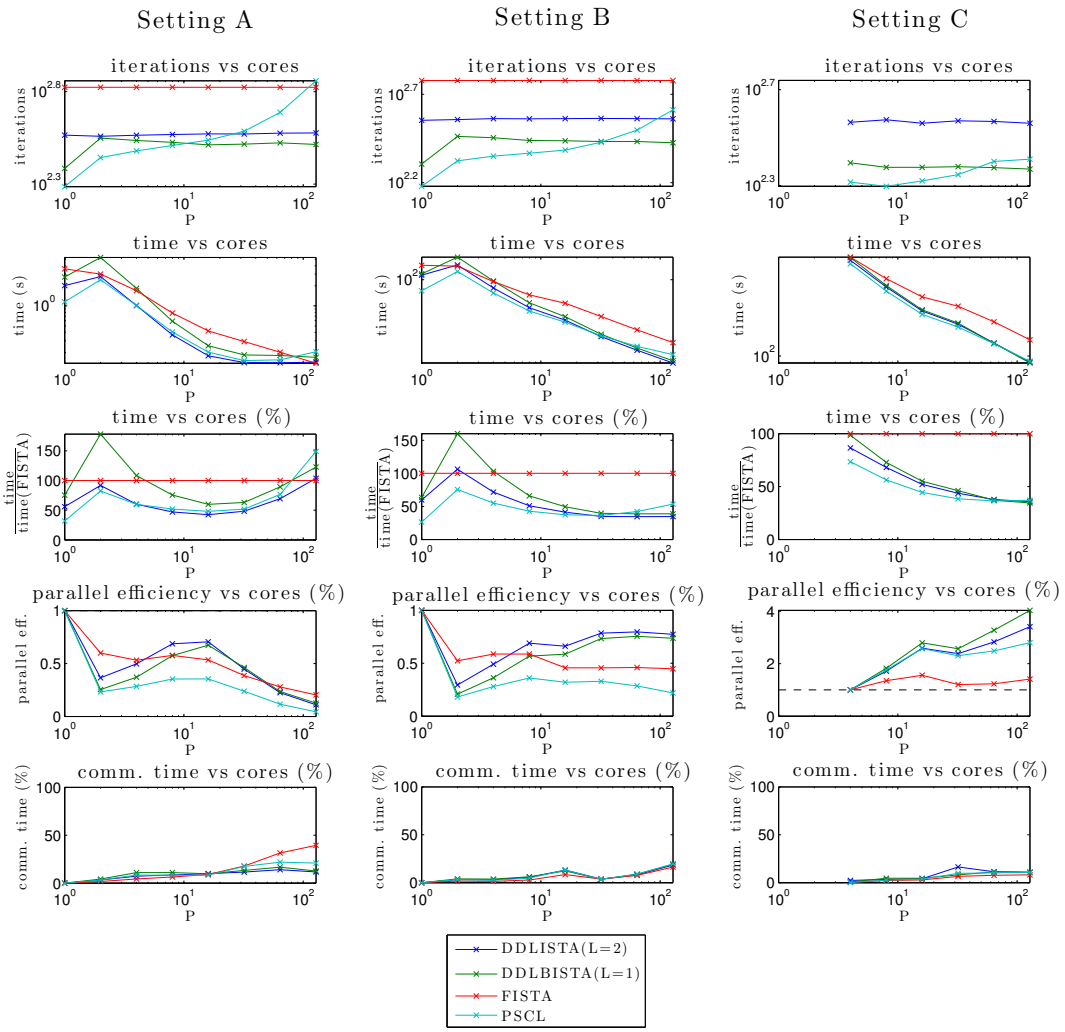


Figure 4.11: Parallelization results for DDLISTA($L_{\max} = 2$), DDLBISTA($L_{\max} = 1$), P-FISTA, PSCLN for 10 random trials of Setting A (left column), Setting B (center column), and Setting C (right column).

C the subspace methods DDLISTA, DDLBISTA, and PSCLN outperform the native parallelization P-FISTA.

4.2.6 A Solver for Large-Scale Hyper- and Multispectral Image Sharpening

In this section, we present an application of sparse recovery for the fusion of hyper- and multispectral images, which is the outcome of a joint work with the German Aerospace

Center (DLR) and presented in [97]. We introduce the corresponding SparseFI project in Section 4.2.6.1, which incorporates the fusion of air- and spaceborne image data being of extremely large size such that the need for high performance computing (HPC) is raised. In its core the image fusion algorithm is based on a large amount of sparse recovery problems for which we determine the best suitable solver in Section 4.2.6.2. In order to solve this set of sparse recovery problems on multi-core systems, one has to apply the efficient work scheduling which is explained in Section 4.2.6.3.

4.2.6.1 The SparseFI Project and High Performance Computing

Image fusion aims at combining two or more images into a single image that features valuable information from all of the input images. Typically, we acquire those input images in the field of remote sensing by air- and spaceborne sensors. One of the most prominent problems is the fusion of two optical images of different spatial and spectral resolution. Many topographic earth observation satellites such as IKONOS, GeoEye, Pleiades, WorldView-2, and WorldView-3 are equipped with both a panchromatic, i.e., single-channel, sensor of very high spatial resolution and a medium to high spatial resolution multispectral instrument. In the literature, the fusion of such an image pair is referred to as *pan-sharpening*. It aims at creating an image which has the spectral resolution of the multispectral image and the spatial resolution of the panchromatic image. Solutions to the pan-sharpening problem have been proposed in [180, 164, 200, 152, 136, 5, 25]. An approach, based on sparse representation based methods was introduced in [202, 209] and resulted in the *Sparse Fusion of Images (SparseFI)* method, which we already introduced briefly in Section 1.1.2. Based on this work, a further development is the use of joint sparse representations in [207, 208, 205], leading to the modified *Joint Sparse Fusion of Images (J-SparseFI)* method.

Hyperspectral instruments, such as AVIRIS, Hyperion, HYDICE, HySpex [10], HISUI [115], and the German next-generation Environmental Mapping and Analysis Program (EnMAP) sensor [175], or the DLR Earth Sensing Imaging Spectrometer (DESIS) acquire electromagnetic energy in docents to hundreds of contiguous wavelength ranges. By this higher spectral resolution we are enabled to identify different materials within the observed scene, each possessing a characteristic spectral signature. However, the high number of bands comes at the expense of degradation in spatial resolution due to the narrowness of each spectral band. In order to allow for applications such as terrain classification, mineral detection, and exploration, multiple materials inside single pixels need to be discriminated, i.e., unmixed [18, 113, 16]. The level of detail and the diversity of materials in hyperspectral data acquired over urban areas is greater compared to rural scenes. This makes applications with such kind of data particularly demanding. We can apply data fusion techniques if corresponding high spatial resolution multispectral data is additionally available. This allows for the

enhancement of the spatial resolution of the hyperspectral image and, thus, for the identification and localization of contributing sources at the resolution scale of the high resolution image. While some pan-sharpening methods would be principally applicable to the fusion of hyperspectral and panchromatic data, the replacement of the panchromatic image by a multispectral image in the fusion problem introduces new methodical challenges. Moreover, in order to apply sophisticated signal processing algorithms to large scale hyperspectral imagery, we require computational resources that have become available only in recent years thanks to the rapid development in computer technology. Hyperspectral-multispectral data fusion methods were proposed in [92, 198, 103, 197, 194, 199, 17].

Based on the precursor algorithm J-SparseFI, we elaborated in the paper [97] a new method for the fusion of hyperspectral and multispectral imagery called *Jointly Sparse Fusion of Hyperspectral and Multispectral Imagery (J-SparseFI-HM)*. It creates a high spatial resolution hyperspectral image patch-wise by exploiting the jointly sparse representation of hyperspectral image patches in dictionaries that are built up from the multispectral image. Based on the earlier attempts [100, 101], it jointly estimates bundles of an adjustable number of high resolution hyperspectral bands by fusing the corresponding low resolution hyperspectral channels with possibly multiple multispectral bands. This approach takes into account the spectral response functions (SRF) of the hyperspectral and multispectral sensor. At the time of the finalization of this thesis, the paper [97] was still under review. Since the research field is highly competitive, we kindly ask the reader to understand that we can only give a rather rough description of the modeling and statement of the J-SparseFI-HM algorithm in this thesis. For details, we refer to the prospectively published version of the paper and doctoral thesis of Claas Grohnfeldt. Fortunately this restriction only applies to the engineering methodology of the project but not to the mathematical and computer scientific details.

Let us take for granted that there exists the J-SparseFI-HM software, which mainly solves a stack of joint sparsity problems of the type (2.12). We refer to them as the *group LASSO* problems (compare Section 2.3). They are roughly derived as follows: In each problem, we have given a low spatial resolution hyperspectral image patch $y_{\text{low}} = (y_1, \ldots, y_d) \in \mathbb{R}^{m_l \times d}$ where the $y_i \in \mathbb{R}^{m_l}$, $i = 1, \ldots, d$, represent a subset of d (vectorized) hyperspectral bands of the image with a size of m_l pixels each. Moreover, we have a low resolution dictionary $D_{\text{low}} \in \mathbb{R}^{m_l \times N}$, with N atoms of a size of m_l pixels, which is obtained by spatial downsampling and filtering from a high resolution dictionary $D_{\text{high}} \in \mathbb{R}^{m_h \times N}$, which contains detail patches (roofs, meadows, streets, etc.) from the high resolution multispectral image. By means of the low resolution dictionary D_{low}, and the low resolution hyperspectral image patch bands y_i, $i = 1 \ldots d$, we want to obtain for each band a sparse representation $x_i \in \mathbb{R}^N$, $i = 1, \ldots, d$, such that $y_i \approx D_{\text{low}} x_i$. Since all bands depict the same scene, we assume to have the same active dictionary elements for all bands, i.e, the x_i, $i = 1, \ldots, d$, are supposed to be

jointly sparse. Thus, we compute by the group LASSO,

$$\arg\min_{z\in\mathbb{R}^{N\times d}} \frac{1}{2}\|D_{\text{low}}z - y_{\text{low}}\|_{\ell_{2,2}} + \lambda\|z\|_{\ell_{1,2}}, \tag{4.85}$$

a joint sparse representation $x = (x_1, \ldots, x_d) \in \mathbb{R}^{N\times d}$, which is composed of the respective sparse representations x_i, $i = 1, \ldots, d$. Then, a high resolution hyperspectral image patch y_{high} is obtained from the sparse representation x and the high resolution dictionary D_{high} via

$$y_{\text{high}} = D_{\text{high}}x.$$

Since this procedure has to be performed for a predefined set of patches, which results from a decomposition of a large image, we potentially have to solve a large number of those problems. Note that the dimensions N, d, m_l, m_h may vary from problem to problem. Nevertheless, those variations are small enough such that we can assume that all problems roughly fit into the same class of problem sizes. Also the properties of the dictionaries only vary slightly, in particular its condition number. Thus, we are in the fortunate situation that the computation of solutions to (4.85) takes approximately the same time for each problem.

The high potential of sparse representation based data fusion methods with respect to image quality was demonstrated in the above mentioned work towards SparseFI, and J-SparseFI. However, the main drawback of such methods is the high computational cost, which is due to the necessity of solving a large number of optimization problems of the type (4.85). Moreover, the number of problems linearly increases with the number of image patches, i.e., with the spatial size of the input images. In Section 4.2.6.2, we determine the best suitable solver for the group LASSO problems of the respective problem size of our application. We comment in Section 4.2.6.3 on the scheduling of such an amount of (to the greatest extent independent) problems on multi-core systems.

In the experiments that are presented in the following sections, we used airborne VNIR HySpex data acquired over Munich, Germany, in 2012. The HySpex sensor is characterized by 160 spectral channels spanning from 415 to 992 nm. Synthetic high resolution multispectral data has been simulated by filtering the reference HySpex image pixel-wise, using the spectral response functions of the WorldView-2 imager in the relevant spectral range. We effectively use the first five WorldView-2 channels ranging from 350 to 700 nm. The high resolution reference image has a ground sampling distance of 2 m and a size of 480 by 480 pixels.

In order to process large scale Earth observation data, the J-SparseFI-HM image fusion software is optimized for memory exploitation on the SuperMUC (see Section 4.2.5.1 for system specifications). Internode MPI communication is kept low in order to maximize parallel efficiency. The J-SparseFI-HM application is compiled with the Intel MPI compiler using the system's wrapper mpiCC.

4.2.6.2 Determining a Suitable Solver

The J-SparseFI-HM algorithm has been developed in order to sharpen multiple individual patches in parallel. The parallelization reduces the overall computational time and allows for a distribution of the data. In order to solve the group LASSO problem (4.85), we have several relevant solvers at disposal, which we presented in Section 4.2.1 and 4.2.2. Out of those we choose the one which is best suited for the hyperspectral-multispectral image fusion. Therefore, we first extracted randomly a set of 10 test problems, which were created by the J-SparseFI-HM software. All test problems have the dimensions $N = 699$, $m_l = 404$, $d = 3$. We notice that the problem size is relatively small since it undercuts the one of Setting A, as defined in Section 4.2.5.1. On this testset, we ran algorithm DDLISTA, DDLBISTA, P-FISTA, and PSCLN for $P = 1, 2, 4, 8, 16, 32, 64, 128$ cores and we created a trivial decomposition of our index sets $\Lambda = \{1, \ldots, N\}$ into P subsets $\Lambda_i = \{(i-1)N/P + 1, \ldots, iN/P\}$, similarly as described in Section 4.2.5. In contrast to our observations in Section 4.2.5.2, we figured out that DDLISTA and DDLBISTA performed best with the parameter $L_{\max} = 1$ for the given class of problems. The results of the simulations are presented in Figure 4.12. Without doubt, the analysis of the plots yields that P-FISTA performs best for such small problems. Furthermore, we observe that the parallel efficiency is always below one. This means in particular that it is more efficient to assign only one core to the solution of each problem (compare also the explanations on the interpretation of parallel efficiency in Example 4.26 and the respective paragraph).

4.2.6.3 Parallel Work Scheduling and Idling

Algorithm 21 J-SparseFI-HM (rough description)

1: **for** patch $i = 1, \ldots, N_p$ **do**
2: create the dictionary D_{low} for the patch i
3: **for** each independent group LASSO problem $j = 1, \ldots, N_t$ **do**
4: solve the j-th problem of the form (4.85)
5: **end for**
6: merge the N_t results of the group LASSO problems together
7: **end for**

We sketch the description of J-SparseFI-HM in Algorithm 21 in order to see where parallelization is possible. According to this description, we need to solve $N_p \cdot N_t$ problems of the type (4.85). However it is not possible to solve them completely independent from each other, since all problems which belong to the same patch i will need the same dictionary and their results have to be merged together. Therefore, synchronization in Step 2 and Step 6 is necessary. In order to explain the parallelization

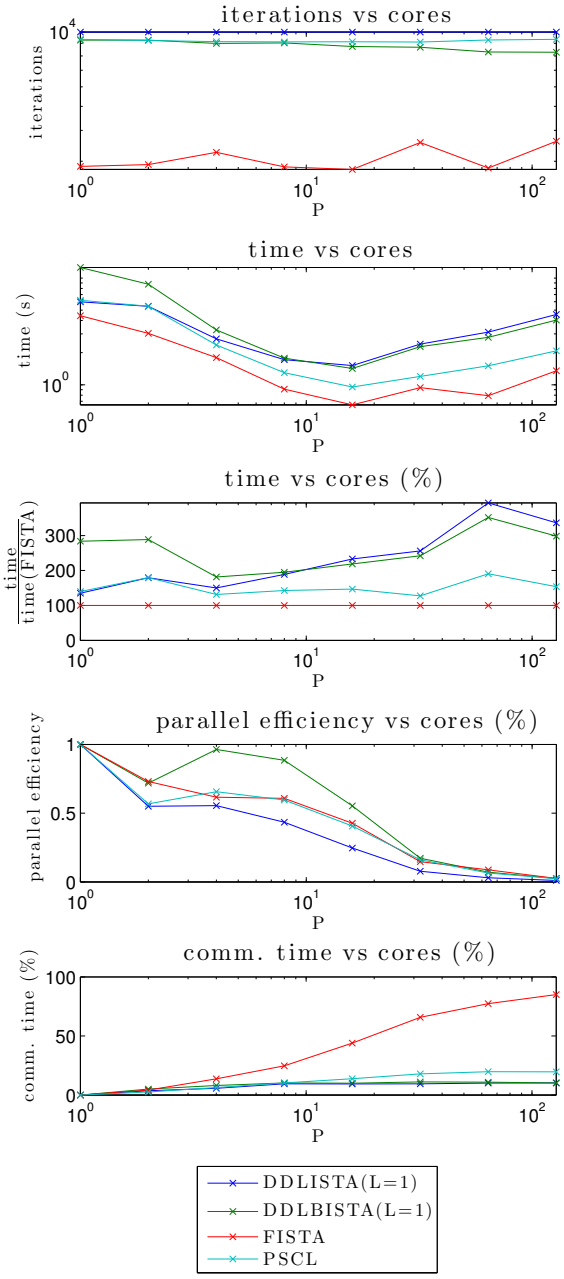


Figure 4.12: Parallelization results for DDLISTA($L_{\max} = 1$), DDLBISTA($L_{\max} = 1$), P-FISTA, and PSCLN for 10 random trials of a typical J-SparseFI-HM setting with $N = 699$, $m_l = 404$, $d = 3$.

strategies, we assume that we have Q cores available and that we split them into G groups of $S = Q/G$ cores, presuming for simplicity that G is a divisor of Q. This allows to assign to each patch a group of S cores.

Recall from the previous section that each problem of the type (4.85) has to be solved by only a single core. Thus, it is clear that $S \leq N_t$. If $S = N_t$, then each core has to solve exactly one problem. However, if $S < N_t$ and S is a divisor of N_t then each cores solves exactly N_t/S problems. Since we assumed that we need approximately the same time in order to solve each group LASSO problem, this means that the workload is well balanced among the S cores. If S is not a divisor of N_t, then the work is unevenly distributed. The part of the group of S cores which has less work, will have to wait for the cores with more workload, since in Step 6 all S cores are synchronized in order to communicate their results within the group. We call this behavior *inner idling*.

When a group of cores finished the computation of one patch, it proceeds to the next patch, etc. In general $N_p \gg G$. If G is not a divisor of N_p, the work is unevenly distributed among the individual groups. Although each group of cores can independently process its portion of the total work, some groups will definitely idle in the end and wait for the other groups. This behavior is called *outer idling*.

Outer and inner idling is at the expense of the parallel efficiency of the algorithm J-SparseFI-HM. Therefore, an effort is necessary to optimally use the resources and reduce the idling. Both, the inner and the outer processing can be modeled as a set of W work units, which has to be processed by V processing units. In this scenario we consider the following two approaches for the scheduling of the work units:

- The first option is to pre-assign the work units to the processing units right at the beginning. Then, processing unit $i \in \{1, \ldots, V\}$ processes the set of work units $\{i+jV \,|\, j = 0, \ldots, \lfloor W/V \rfloor\}$ if $i \leq (W \bmod V)$, and $\{i+jV \,|\, j = 0, \ldots, \lfloor W/V \rfloor - 1\}$ otherwise. If V is a divisor of W, there is no case differentiation and the work is evenly distributed. This concept is called a *fixed schedule*.

- The second option would be to consider the W work units as a global stack. Then, as soon as a processing unit is done with its previous work, it gets assigned at runtime to the top work unit of the global stack. The work unit is removed from the stack and the next unit is free for treatment. Like this, processing units are only idling when the stack is empty. This concept is called *work stealing schedule*.

The advantage of the work stealing concept is the flexible handling of the work units. Nevertheless, it requires the maintenance of a global counter variable with exclusive write access, which stores the index of the next free work unit. In particular if V is large and the work units are small, i.e., they can be computed in short times, the counter variable needs to be accessed frequently by many processing units. The effect

is idling in many processing units since only one of them can read from and write to the counter. In contrast, the fixed schedule does not require synchronization among the processes at runtime. However, the static distribution of work may lead to unbalanced work load among the different processing units. Therefore, we propose to take the best of both worlds, and split the work units into one portion, which is initially done by the fixed schedule, and the remainder being processed by a work stealing schedule. Concretely, we define the parameter $s_{\text{ws}} \in \mathbb{Z}$, $s_{\text{ws}} \geq -1$ which splits the total work into $W_{\text{fs}} := \max(\min(V(\lfloor W/V \rfloor - s_{\text{ws}}), W), 0)$ and $W_{\text{ws}} := W - W_{\text{fs}}$ work units that are handled by the fixed schedule and the work stealing schedule respectively. This means in particular that no work stealing ($W_{\text{ws}} = 0$) is done for $s_{\text{ws}} = -1$, and $W_{\text{ws}} = (W \bmod V)$ for $s_{\text{ws}} = 0$.

By means of those scheduling paradigms, we want to schedule the inner and the outer `for`-loop of Algorithm 21. Since in general $N_t \ll N_p$, the maintenance of a work stealing counter in the inner loop would be too inefficient. From a practical point of view it is best to choose S as a divisor of N_t in order to have the best work load distribution within the inner loop. In the outer loop we apply the hybrid fixed/work stealing schedule, where we set $W = N_p$ and $V = G$. Thus, assuming that the total number of cores Q, as well as N_p and N_t are given, it remains the question which choice of the parameters S and s_{ws} is best. Since it is very hard to predict the running time of the individual group LASSO problems, we need to determine those parameters experimentally.

In a generic experiment we solve a test problem with $N_p = 4096$ patches. For each patch we need to solve $N_t = 32$ group LASSO problems. According to our results in 4.2.6.2, we solve each of the problems with a non-parallelized FISTA. We test each combination of the parameters $s_{\text{ws}} \in \{0, 1, 2, 3\}$ and $S \in \{1, 2, 4, 8, 16\}$ for a total number of $Q = 32$ cores. We present the outcome of the respective computational time in Table 4.1. The effect of the inner idling is visible by the increasing values for increasing S. For $S = 1$, one can see the effect of outer idling. This effect is partly absorbed through work stealing ($s_{\text{ws}} = 1$). However, too much work stealing leads to concurrent access to the common global counter variable and reduces the performance ($s_{\text{ws}} > 1$). In particular for $S = 1$, work stealing only has a negative effect since too many groups ($G = Q/S = 32$) block each other. Thus, the algorithm runs fastest for $s_{\text{ws}} = 1$ and $S = 2$.

We conclude that the hybrid scheduling leads to a slight acceleration of the algorithm J-SparseFI-HM.

$s_{\mathbf{ws}}$ \ S	**1**	**2**	**4**	**8**	**16**
0	509.07	500.10	529.86	595.89	719.33
1	670.51	**472.40**	507.63	579.12	696.01
2	721.42	510.49	514.80	569.27	691.16
3	793.48	510.89	522.67	556.60	681.64

Table 4.1: Parallel computation time (in s) of J-SparseFI-HM for different combinations of the work stealing parameter s_{ws} and number of processors S dedicated to the inner loop.

Chapter 5

Conclusion and Outlook

In this thesis we improved the robustness and efficiency of selected sparse recovery methods with the main outcome that the support identification properties of convex methods can be tremendously enhanced by non-convex techniques, and that properly tuned second order methods such as iteratively re-weighted least squares can outperform first order methods such as fast iterative soft thresholding and iterative hard thresholding.

In terms of robustness, we proposed and successfully evaluated non-convex methods for the particular case when a sparse signal is perturbed by strong noise and the noise folding phenomenon makes it difficult to determine reliable results from linear measurements of this perturbed signal. In particular, we could undoubtedly reveal that the popular convex methods, such as ℓ_1-minimization, BPDN, LASSO, etc. quickly reach their limitations, when it comes to the need of an exact, robust, and reliable support identification. In such a setting, it is advantageous to use selective decoders such as SLP, IHT, or a multi-penalty optimization. Those methods take into account the different statistical properties of the relevant part of the signal and the noise. Local minimizers, which are an intrinsic problem that one has to face sooner or later when dealing with non-convex methods, can be avoided to the greatest extent when executing the methods with reasonable starting values, as, e.g., the output of the above mentioned convex decoders. We conclude that the simple application of standard ℓ_1-minimization based decoders to problems where the measurement process is perturbed by strong signal noise, is in general not sufficient to obtain a reliable support identification.

Regarding the computational efficiency of sparse recovery methods, we worked on the one hand on a setting, were fast matrix-vector multiplications can be applied. We proposed a CG acceleration of IRLS. The surprising conclusion of the respective results and comparison to state-of-the-art algorithms is that first order methods can be outperformed by second order methods.

In the case that one is forced to distribute large data on multiple machines and find parallel solutions for sparse recovery, we proposed a flexible algorithm, which is based on a domain decomposition and multiple parallel thresholding operations. We showed that such an algorithm scales in practice very well if a large number of

machines is involved. The consideration of multiple independent steps on each core without intermediate communication turns out to be advantageous in terms of the parallel efficiency.

By the example of a fusion method for hyperspectral-multispectral data of the Earth's surface, we presented eventually an application of sparse recovery techniques on real data, proposing an efficient treatment of such large data sets by means of the best solver choice and an optimal work scheduling of the stack of sparse recovery problems, which is produced by the method. From this example we learned that the proper choice of an algorithm can depend on the problem size and that a non-trivial scheduling has a positive effect on the runtime.

The particular results that were presented in this thesis lead to the following additional research questions by which we intend to stimulate master students and doctoral students but also advanced researchers for further thinking and activity on the topics that were addressed:

- First insights in the particular (numerical) dependence of IRLS on the limit of the sequence $(\varepsilon^n)_{n\in\mathbb{N}}$ were given in Section 2.4.1.2. We also showed a theoretical upper bound for the approximation error of two successive iterates, which confirms the numerical tests. The challenge of finding a sharp theoretical lower bound is open. Also numerical tests with different encoding matrices may be interesting in order to obtain further knowledge on the dependencies of the involved constants.

- The newly introduced methods in Chapter 3 can be tested and analyzed for different encoding matrices, which may not have such advantageous spectral properties as the RIP/NSP.

- Although we showed that the method CG-IRLS-λ and its derivatives exhibit a fast convergence, we did not give theoretical guarantees on the rate of convergence. Regarding Figure 4.7, this seems to be a complicated task since visually one cannot identify a constant rate. Furthermore, the role of the parameter p is still not thoroughly investigated (compare Figure 4.9) and promises—at least in a setting with low noise—a further gain in the performance of the algorithm.

- It is an open problem to show the strong convergence of Algorithm 20 when the stepsize $t = t^{(n)}$ is not fixed. Using the backtracking condition as well as an adaptation of the techniques from [49] may lead to a proper proof.

Appendix A

Proofs

Proof (Proof of Lemma 2.15). Since $\lambda \to 0$, one can choose a null sequence $(\lambda_n)_{n\in\mathbb{N}}$. Furthermore, let u^* be a solution to problem (2.4). Then the inequality $\|u_{\lambda_n}\|_{\ell_1(\Lambda)} \leq \frac{1}{\lambda_n}\mathcal{J}_{\lambda_n}(u_{\lambda_n}) \leq \frac{1}{\lambda_n}\mathcal{J}_{\lambda_n}(u^*) = \|u^*\|_{\ell_1(\Lambda)}$ is valid for all n. Thus, there is a subsequence $(\lambda_{n_k})_{k\in\mathbb{N}}$ such that $u_{\lambda_{n_k}} \to u_0$ and $\|u_0\|_{\ell_1(\Lambda)} \leq \|u^*\|_{\ell_1(\Lambda)}$. Furthermore, from the inequality $\lambda_{n_k}\|u^*\|_{\ell_1(\Lambda)} \geq \mathcal{J}_{\lambda_{n_k}}(u_{\lambda_{n_k}}) = \|Tu_{\lambda_{n_k}} - y\|^2_{\ell_2(\Lambda)} + \lambda_{n_k}\|u_{\lambda_{n_k}}\|_{\ell_1(\Lambda)} \geq 0$, the boundedness of $\|u_{\lambda_{n_k}}\|_{\ell_1(\Lambda)}$, the fact that $\lambda_{n_k} \to 0$, and the continuity of the norm one obtains $0 = \lim_{k\to\infty} \|Tu_{\lambda_{n_k}} - y\|^2_{\ell_2(\Lambda)} = \|Tu_0 - y\|^2_{\ell_2(\Lambda)}$. Since $\|u_0\|_{\ell_1(\Lambda)} \leq \|u^*\|_{\ell_1(\Lambda)}$ and $Tu_0 = y$, the limit u_0 is a solution of problem (2.4). □

Proof (Proof of Lemma 2.16). The proof is divided into two parts. First, we show that 0 is a solution of problem (2.15), and in the second part the uniqueness of this solution is established.

As derived in Section 2.2.2 the optimality conditions for problem (2.15) are given by

$$\begin{aligned} -2(T^*(Tu-y))_i &= \lambda \frac{u_i}{\|u_i\|_{\ell_2}} && \text{if } u_i \neq 0, \\ 2\|(T^*(Tu-y))_i\|_{\ell_2} &\leq \lambda && \text{if } u_i = 0, \end{aligned} \qquad i \in \Lambda.$$

The substitution of $u = 0$ in the optimality conditions yields $2\|(T^*y)_i\|_{\ell_2} \leq \lambda$, $i \in \Lambda$. This holds true since $\lambda > 2\|T^*y\|_{\ell_\infty(\Lambda)}$ is given.

To prove the uniqueness of the solution, we show that the assumption of any second solution $\tilde{u} \neq 0$ leads to a contradiction. Assume there is $\tilde{u} \neq 0$ which solves problem (2.15). Then

$$\mathcal{J}_\lambda(\tilde{u}) - \mathcal{J}_\lambda(0) \leq 0. \tag{A.1}$$

On the other hand we have

$$\begin{aligned} \mathcal{J}_\lambda(\tilde{u}) - \mathcal{J}_\lambda(0) &= \lambda\|\tilde{u}\|_{\ell_1(\Lambda)} + \|T\tilde{u}\|^2_{\ell_2(\Lambda)} - 2\langle T\tilde{u}, y\rangle \\ &\geq \lambda\|\tilde{u}\|_{\ell_1(\Lambda)} + \|T\tilde{u}\|^2_{\ell_2(\Lambda)} - 2\|\tilde{u}\|_{\ell_1(\Lambda)}\|T^*y\|_{\ell_\infty(\Lambda)} \qquad \text{(A.2)} \\ &= \|T\tilde{u}\|^2_{\ell_2(\Lambda)} + (\lambda - 2\|T^*y\|_{\ell_\infty(\Lambda)})\|\tilde{u}\|_{\ell_1(\Lambda)} > 0, \end{aligned}$$

where Hölder's inequality is used in (A.2). This contradicts to (A.1), completing the proof. □

List of Figures

List of Tables

Bibliography

[1] M. V. Afonso, J. M. Bioucas-Dias, and M. A. Figueiredo. “Fast image recovery using variable splitting and constrained optimization”. In: *IEEE Transactions on Image Processing* 19.9 (2010), pp. 2345–2356.

[2] E. Alba. “Parallel evolutionary algorithms can achieve super-linear performance”. In: *Information Processing Letters*. Evolutionary Computation 82.1 (2002), pp. 7–13. ISSN: 0020-0190.

[3] G. Alessandrini and S. Vessella. “Lipschitz stability for the inverse conductivity problem”. In: *Advances in Applied Mathematics* 35.2 (2005), pp. 207–241.

[4] B. Alexeev and R. Ward. “On the complexity of Mumford-Shah-type regularization, viewed as a relaxed sparsity constraint”. In: *IEEE Transactions on Image Processing* 19.10 (2010), pp. 2787–2789.

[5] K. Amolins, Y. Zhang, and P. Dare. “Wavelet based image fusion techniques—An introduction, review and comparison”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 62.4 (2007), pp. 249–263.

[6] E. Arias-Castro and Y. C. Eldar. “Noise folding in compressed sensing”. In: *IEEE Signal Processing Letters* (2011), pp. 478–481.

[7] M. Artina, M. Fornasier, and S. Peter. *Damping Noise-Folding and Enhanced Support Recovery in Compressed Sensing - Extended Technical Report*. Tech. rep. Nov. 2014.

[8] M. Artina. “Lagrangian Methods for Constrained Non-Convex Minimizations and Applications in Fracture Mechanics”. Dissertation. Munich: Technical University of Munich, 2015.

[9] M. Artina, M. Fornasier, and F. Solombrino. “Linearly constrained nonsmooth and nonconvex minimization”. In: *SIAM Journal on Optimization* 23.3 (2013), pp. 1904–1937.

[10] R. Baissa, K. Labbassi, P. Launeau, A. Gaudin, and B. Ouajhain. “Using HySpex SWIR-320m hyperspectral data for the identification and mapping of minerals in hand specimens of carbonate rocks from the Ankloute Formation (Agadir Basin, Western Morocco)”. In: *Journal of African Earth Sciences* 61.1 (Aug. 2011), pp. 1–9. ISSN: 1464-343X.

[11] R. G. Baraniuk. “Compressive sensing”. In: *IEEE Signal Processing Magazine* 24.4 (2007), pp. 118–121.

[12] R. G. Baraniuk, M. Davenport, R. DeVore, and M. Wakin. “A simple proof of the restricted isometry property for random matrices”. In: *Constructive Approximation* 28.3 (2008), pp. 253–263.

[13] A. Beck and M. Teboulle. “A fast iterative shrinkage-thresholding algorithm for linear inverse problems”. In: *SIAM Journal on Imaging Sciences* 2.1 (2009), pp. 183–202. ISSN: 1936-4954.

[14] C. R. Berger, S. Zhou, J. C. Preisig, and P. Willett. “Sparse channel estimation for multicarrier underwater acoustic communication: From subspace methods to compressed sensing”. In: *IEEE Transactions on Signal Processing* 58.3 (2010), pp. 1708–1721.

[15] P. Bickel, Y. Ritov, and A. Tsybakov. “Simultaneous analysis of lasso and Dantzig selector”. In: *Annals of Statistics* 37.4 (2009), pp. 1705–1732.

[16] J. Bieniarz, E. Aguilera, X. X. Zhu, R. Müller, and P. Reinartz. “Joint Sparsity Model for Multilook Hyperspectral Image Unmixing”. In: *IEEE Geoscience and Remote Sensing Letters* 12.4 (Apr. 2015), pp. 696–700. ISSN: 1545-598X.

[17] J. Bieniarz, R. Muller, X. X. Zhu, and P. Reinartz. “Hyperspectral image resolution enhancement based on joint sparsity spectral unmixing”. In: *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. July 2014, pp. 2645–2648.

[18] J. M. Bioucas-Dias, A. Plaza, S. Member, N. Dobigeon, M. Parente, Q. Du, S. Member, P. Gader, and J. Chanussot. “Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches”. In: *IEEE Journal on Selected Topics in Applied Earth Observations and Remote Sensing* (2012), pp. 354–379.

[19] T. Blumensath and M. E. Davies. “Iterative thresholding for sparse approximations”. In: *The Journal of Fourier Analysis and Applications* 14.5 (2008), pp. 629–654. ISSN: 1069-5869.

[20] T. Blumensath and M. E. Davies. “Iterative hard thresholding for compressed sensing.” In: *Applied and Computational Harmonic Analysis* 27.3 (2009), pp. 265–274.

[21] K. Bredies and M. Holler. “Regularization of linear inverse problems with total generalized variation”. In: *Journal of Inverse and Ill-posed Problems* 22.6 (2014), pp. 871–913.

[22] K. Bredies and D. A. Lorenz. “Linear convergence of iterative soft-thresholding”. In: *Journal of Fourier Analysis and Applications* 14.5 (2008), pp. 813–837.

[23] K. Bredies and D. A. Lorenz. "Minimization of non-smooth, non-convex functionals by iterative thresholding". In: *Journal of Optimization Theory and Applications* 165.1 (2015), pp. 78–112.

[24] R. P. Brent. "The parallel evaluation of general arithmetic expressions". In: *Journal of the ACM (JACM)* 21.2 (1974), pp. 201–206.

[25] B. Brower and C. Laben. *Process for enhancing the spatial resolution of multispectral imagery using pan-sharpening*. US Patent 6,011,875. Google Patents, Jan. 2000.

[26] E. J. Candès et al. "Compressive sampling". In: *Proceedings of the international congress of mathematicians*. Vol. 3. Madrid, Spain, 2006, pp. 1433–1452.

[27] E. J. Candès and Y. Plan. "Near-ideal model selection by l_1 minimization". In: *The Annals of Statistics* 37.5A (2009), pp. 2145–2177.

[28] E. J. Candès, J. Romberg, and T. Tao. "Stable signal recovery from incomplete and inaccurate measurements". In: *Communications on pure and applied mathematics* 59.8 (2006), pp. 1207–1223.

[29] E. J. Candès and J. Romberg. "Quantitative robust uncertainty principles and optimally sparse decompositions". In: *Foundations of Computational Mathematics* 6.2 (2006), pp. 227–254.

[30] E. J. Candès and T. Tao. "Near optimal signal recovery from random projections: universal encoding strategies?" In: *IEEE Transactions on Information Theory* 52.12 (2006), pp. 5406–5425.

[31] E. Candès and T. Tao. "The Dantzig selector: Statistical estimation when p is much larger than n". In: *The Annals of Statistics* 35.6 (2007), pp. 2313–2351.

[32] E. Candès and M. Wakin. "An introduction to compressive sampling". In: *IEEE Signal Processing Magazine* 25.2 (2008), pp. 21–30.

[33] E. Candès, M. Wakin, and S. Boyd. "Enhancing Sparsity by Reweighted l_1 Minimization". English. In: *Journal of Fourier Analysis and Applications* 14.5-6 (2008), pp. 877–905. ISSN: 1069-5869.

[34] I. Carron. *Compressive Sensing: The Big Picture*. `https://sites.google.com/site/igorcarron2/cs`. 2015.

[35] A. Chambolle and T. Pock. "A first-order primal-dual algorithm for convex problems with applications to imaging". In: *Journal of Mathematical Imaging and Vision* 40.1 (2011), pp. 120–145.

[36] R. Chartrand. "Exact Reconstruction of Sparse Signals via Nonconvex Minimization". In: *Signal Processing Letters, IEEE* 14.10 (Oct. 2007), pp. 707–710. ISSN: 1070-9908.

[37] R. Chartrand and W. Yin. "Iteratively reweighted algorithms for compressive sensing". In: *IEEE International Conference on Acoustics, Speech and Signal Processing*. Mar. 2008, pp. 3869–3872.

[38] R. Chartrand and V. Staneva. "Restricted isometry properties and nonconvex compressive sensing". In: *Inverse Problems* 24.3 (2008), p. 035020. ISSN: 0266-5611.

[39] A. K. Cline. "Rate of convergence of Lawson's algorithm". In: *Mathematics of Computation* 26 (1972), pp. 167–176. ISSN: 0025-5718.

[40] A. Cohen, W. Dahmen, and R. DeVore. "Adaptive wavelet schemes for nonlinear variational problems". In: *SIAM Journal on Numerical Analysis* 41.5 (2003), pp. 1785–1823. ISSN: 0036-1429.

[41] A. Cohen, W. Dahmen, and R. DeVore. "Compressed sensing and best k-term approximation". In: *Journal of the American Mathematical Society* 22.1 (2009), pp. 211–231.

[42] P. L. Combettes and J.-C. Pesquet. "Proximal splitting methods in signal processing". In: *Fixed-point algorithms for inverse problems in science and engineering*. Springer, 2011, pp. 185–212.

[43] P. L. Combettes and V. R. Wajs. "Signal recovery by proximal forward-backward splitting". In: *Multiscale Modeling & Simulation* 4.4 (2005), pp. 1168–1200.

[44] D. E. Culler, R. M. Karp, D. Patterson, A. Sahay, E. E. Santos, K. E. Schauser, R. Subramonian, and T. von Eicken. "LogP: A practical model of parallel computation". In: *Communications of the ACM* 39.11 (1996), pp. 78–85.

[45] S. Dahlke, M. Fornasier, and T. Raasch. "Multilevel preconditioning and adaptive sparse solution of inverse problems". In: *Mathematics of Computation* 81.277 (2012), pp. 419–446.

[46] I. Daubechies and G. Teschke. "Variational image restoration by means of wavelets: simultaneous decomposition, deblurring, and denoising". In: *Applied and Computational Harmonic Analysis* 19.1 (2005), pp. 1–16. ISSN: 1063-5203.

[47] I. Daubechies, M. Defrise, and C. De Mol. "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint". In: *Communications on Pure and Applied Mathematics* 57.11 (2004), pp. 1413–1457.

[48] I. Daubechies, R. DeVore, M. Fornasier, and C. Güntürk. "Iteratively re-weighted least squares minimization for sparse recovery". In: *Communications on Pure and Applied Mathematics* 63.1 (2010), pp. 1–38.

[49] I. Daubechies, M. Fornasier, and I. Loris. "Accelerated projected gradient methods for linear inverse problems with sparsity constraints". In: *Journal of Fourier Analysis and Applications* 14.5-6 (2008), pp. 764–792.

[50] M. A. Davenport. *The RIP and the NSP*. `http://cnx.org/contents/l7a37n1E@5/The-RIP-and-the-NSP`. Apr. 14, 2011.

[51] M. A. Davenport, J. N. Laska, J. R. Treichler, and R. G. Baraniuk. "The pros and cons of compressive sensing for wideband signal acquisition: noise folding versus dynamic range". In: *IEEE Transactions on Signal Processing* 60.9 (2012), pp. 4628–4642.

[52] DLR (EOC). *EnMap*. `http://www.enmap.org`. 2015.

[53] Q. Dong, X. Liu, Z.-W. Wen, and Y.-X. Yuan. "A Parallel Line Search Subspace Correction Method for Composite Convex Optimization". en. In: *Journal of the Operations Research Society of China* 3.2 (May 2015), pp. 163–187. ISSN: 2194-668X, 2194-6698.

[54] D. L. Donoho and X. Huo. "Uncertainty principles and ideal atomic decomposition". In: *IEEE Transactions on Information Theory* 47.7 (2001), pp. 2845–2862. ISSN: 0018-9448.

[55] D. L. Donoho and G. Kutyniok. "Microlocal analysis of the geometric separation problem". In: *Communications on Pure and Applied Mathematics* 66.1 (2013), pp. 1–47. ISSN: 0010-3640.

[56] D. L. Donoho and P. Stark. "Uncertainty principles and signal recovery". In: *SIAM Journal on Applied Mathematics* 49.3 (1989), pp. 906–931. ISSN: 0036-1399.

[57] D. L. Donoho. "Compressed sensing". In: *IEEE Transactions on Information Theory* 52.4 (2006), pp. 1289–1306.

[58] D. L. Donoho and B. F. Logan. "Signal recovery and the large sieve". In: *SIAM Journal on Applied Mathematics* 52.2 (1992), pp. 577–591.

[59] D. L. Donoho and Y. Tsaig. "Fast solution of l1-norm minimization problems when the solution may be sparse". In: *IEEE Transactions on Information Theory* 54.11 (2008), pp. 4789–4812.

[60] D. L. Donoho, M. Vetterli, R. A. DeVore, and I. Daubechies. "Data compression and harmonic analysis". In: *Information Theory, IEEE Transactions on* 44.6 (1998), pp. 2435–2476.

[61] D. Dorsch and H. Rauhut. "Sparse recovery in MIMO radar - dependence on the support structure". In: *2015 3rd International Workshop on Compressed Sensing Theory and its Applications to Radar, Sonar and Remote Sensing (CoSeRa)*. June 2015, pp. 56–60.

[62] DSP at Rice University. *Compressive Sensing Resources*. `http://dsp.rice.edu/cs`. 2015.

[63] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. "Least angle regression". In: *Annals of Statistics* 32.2 (2004), pp. 407–499.

[64] M. Ehler, M. Fornasier, and J. Sigl. "Quasi-linear compressed sensing". In: *Multiscale Modeling & Simulation* 12.2 (2014), pp. 725–754.

[65] *Eigen Library*. `http://eigen.tuxfamily.org`.

[66] I. Ekeland and R. Témam. *Convex Analysis and Variational Problems*. SIAM, Dec. 1999. ISBN: 978-0-89871-450-0.

[67] Y. C. Eldar, P. Kuppinger, and H. Bölcskei. "Block-sparse signals: Uncertainty relations and efficient recovery". In: *IEEE Transactions on Signal Processing* 58.6 (2010), pp. 3042–3054.

[68] Y. C. Eldar and G. Kutyniok. *Compressed sensing: theory and applications*. Cambridge University Press, 2012.

[69] H. Engl, M. Hanke, and A. Neubauer. *Regularization of Inverse Problems*. Vol. 375. Mathematics and Its Applications. Dordrecht, Boston, London: Kluwer Academic Publishers, 1996.

[70] E. Esser. "Applications of Lagrangian-based alternating direction methods and connections to split Bregman". In: *CAM report* 9 (2009), p. 31.

[71] M. A. Figueiredo and R. D. Nowak. "An EM algorithm for wavelet-based image restoration". In: *IEEE Transactions on Image Processing* 12.8 (2003), pp. 906–916.

[72] M. Figueiredo, R. Nowak, and S. Wright. "Gradient Projection for Sparse Reconstruction: Application to Compressed Sensing and Other Inverse Problems". In: *IEEE Journal of Selected Topics in Signal Processing* 1.4 (Dec. 2007), pp. 586–597. ISSN: 1932-4553.

[73] R. Fletcher. "On the Barzilai-Borwein Method". In: *Optimization and Control with Applications*. Ed. by L. Qi, K. Teo, and X. Yang. Applied Optimization 96. Springer US, 2005, pp. 235–256. ISBN: 978-0-387-24254-5 978-0-387-24255-2.

[74] M. Fornasier. "Domain decomposition methods for linear inverse problems with sparsity constraints". In: *Inverse Problems* 23.6 (2007), pp. 2505–2526. ISSN: 0266-5611.

[75] M. Fornasier. "Numerical Methods for Sparse Recovery". In: *Theoretical Foundations and Numerical Methods for Sparse Recovery*. Radon Series on Computational and Applied Mathematics. De Gruyter, 2010, pp. 93–200. ISBN: 978-3-11-022614-0.

[76] M. Fornasier, V. Naumova, and S. Pereverzyev. "Parameter choice strategies for multi-penalty regularization". In: *SIAM Journal on Numerical Analysis* 52.4 (2014), pp. 1770–1794.

[77] M. Fornasier and H. Rauhut. "Compressive Sensing". In: *Handbook of Mathematical Methods in Imaging*. Ed. by O. Scherzer. Springer, 2011, pp. 187–228.

[78] M. Fornasier and H. Rauhut. "Recovery algorithms for vector-valued data with joint sparsity constraints". In: *SIAM Journal on Numerical Analysis* 46.2 (2008), pp. 577–613. ISSN: 0036-1429.

[79] M. Fornasier and R. Ward. "Iterative thresholding meets free-discontinuity problems". In: *Found. Comput. Math.* 10.5 (2010), pp. 527–567.

[80] M. Fornasier, A. Langer, and C.-B. Schönlieb. "A convergent overlapping domain decomposition method for total variation minimization". In: *Numerische Mathematik* 116.4 (2010), pp. 645–685.

[81] M. Fornasier and S. Peter. "An Overview on Algorithms for Sparse Recovery". In: *Sparse Reconstruction and Compressive Sensing in Remote Sensing*. Ed. by R. Bamler and X. X. Zhu. Earth Observation and Image Processing. to appear. Springer, 2016.

[82] M. Fornasier, S. Peter, H. Rauhut, and S. Worm. "Conjugate gradient acceleration of iteratively re-weighted least squares methods". In: *Computational Optimization and Applications* (Mar. 2016). to appear.

[83] M. Fornasier, H. Rauhut, and R. Ward. "Low-rank matrix recovery via iteratively reweighted least squares minimization". In: *SIAM Journal on Optimization* 21.4 (2011), pp. 1614–1640. ISSN: 1052-6234.

[84] S. Foucart and H. Rauhut. *A Mathematical Introduction to Compressive Sensing*. Birkhäuser Basel, 2013. ISBN: 978-0-8176-4947-0 978-0-8176-4948-7.

[85] J. Friedman, T. Hastie, and R. Tibshirani. "A note on the group lasso and a sparse group lasso". In: *arXiv preprint arXiv:1001.0736* (2010).

[86] M. Frigo and S. G. Johnson. "The design and implementation of FFTW3". In: *Proceedings of the IEEE* 93.2 (2005), pp. 216–231.

[87] D. Gabay and B. Mercier. "A dual algorithm for the solution of nonlinear variational problems via finite element approximation". In: *Computers & Mathematics with Applications* 2.1 (1976), pp. 17–40.

[88] A. Garnaev and E. Gluskin. "On widths of the Euclidean ball". In: *Soviet Mathematics Doklady* 30 (1984), pp. 200–204.

[89] R. Glowinski and P. Le Tallec. *Augmented Lagrangian and operator-splitting methods in nonlinear mechanics*. Vol. 9. SIAM, 1989.

[90] E. D. Gluskin. "Norms of random matrices and widths of finite-dimensional sets". In: *Mathematics of the USSR-Sbornik* 48.1 (1984), p. 173.

[91] T. Goldstein, C. Studer, and R. G. Baraniuk. "A Field Guide to Forward-Backward Splitting with a FASTA Implementation". In: *arXiv:1411.3406* (Nov. 2014). arXiv: 1411.3406.

[92] R. B. Gomez, A. Jazaeri, and M. Kafatos. "Wavelet-based hyperspectral and multispectral image fusion". In: *Proc. SPIE*. Vol. 4383. 2001, pp. 36–42.

[93] M. Grant and S. Boyd. *CVX: Matlab Software for Disciplined Convex Programming, version 2.1*. http://cvxr.com/cvx. Mar. 2014.

[94] M. Grant and S. Boyd. "Graph implementations for nonsmooth convex programs". In: *Recent Advances in Learning and Control*. Ed. by V. Blondel, S. Boyd, and H. Kimura. Lecture Notes in Control and Information Sciences. http://stanford.edu/simboyd/graph_dcp.html. Springer-Verlag Limited, 2008, pp. 95–110.

[95] C. Grohnfeldt, T. M. Burns, and X. X. Zhu. "Dictionary Learning Strategies for Sparse Representation Based Hyperspectral Image Enhancement". In: *Proceedings of Whispers 2014*. Tokyo, Japan: IEEE Xplore, 2015, pp. 1–4.

[96] C. Grohnfeldt, T. M. Burns, and X. X. Zhu. "Dynamic Dictionariy Learning Methods for Sparse Representation Based Multiresolution Image Fusion". In: *IEEE Transactions on Image Processing* (2015). submitted.

[97] C. Grohnfeldt, S. Peter, and X. X. Zhu. "Jointly Sparse Fusion of Hyperspectral and Multispectral Imagery - The J-SparseFI-HM Algorithm". In: *IEEE Transactions on Geoscience and Remote Sensing* (2015). submitted.

[98] C. Grohnfeldt and X. X. Zhu. "Splitting the Hyperspectral-Multispectral Image Fusion Problem into Weighted Pan-sharpening Problems - The Spectral Grouping Concept". In: *Proceedings of Whispers 2014*. Tokyo, Japan: IEEE Xplore, 2015, pp. 1–4.

[99] C. Grohnfeldt and X. X. Zhu. "Towards a Combined Sparse Representation and Unmixing Based Hyperspectral Resolution Enhancement Method". In: *IEEE International Geoscience and Remote Sensing Symposium*. Milan, Italy, 2015, pp. 1–4.

[100] C. Grohnfeldt, X. X. Zhu, and R. Bamler. "Jointly Sparse Fusion of Hyperspectral and Multispectral Imagery". In: *IEEE International Geoscience and Remote Sensing Symposium*. IEEE, 2013.

[101] C. Grohnfeldt, X. X. Zhu, and R. Bamler. "The J-SparseFI-HM Hyperspectral Resolution Enhancement Method - Now Fully Automated." In: *Proceedings of Whispers 2014*. Lausanne, Switzerland: IEEE Xplore, 2014, pp. 1–4.

[102] J. L. Gustafson. "Fixed time, tiered memory, and superlinear speedup". In: *Proceedings of the Fifth Distributed Memory Computing Conference (DMCC5)*. 1990, pp. 1255–1260.

[103] R. Hardie, M. Eismann, and G. Wilson. "MAP estimation for hyperspectral image resolution enhancement using an auxiliary sensor". In: *Image Processing, IEEE Transactions on* 13.9 (Aug. 2004), pp. 1174–1184. ISSN: 1057-7149.

[104] J. Haupt, R. G. Baraniuk, R. Castro, and R. Nowak. "Compressive distilled sensing: Sparse recovery using adaptivity in compressive measurements". In: *Proceedings of the 43rd Asilomar conference on Signals, systems and computers*. Asilomar'09. Piscataway, NJ, USA: IEEE Press, 2009.

[105] J. Haupt, R. G. Baraniuk, R. Castro, and R. Nowak. "Sequentially designed compressed sensing". In: *Proc. IEEE/SP Workshop on Statistical Signal Processing*. 2012.

[106] J. Haupt, R. Castro, and R. Nowak. "Distilled sensing: Adaptive sampling for sparse detection and estimation". In: *IEEE Transactions on Information Theory* 57.9 (2011), pp. 6222–6235.

[107] D. P. Helmbold and C. E. McDowell. "Modeling speedup (n) greater than n". In: *IEEE Transactions on Parallel & Distributed Systems* 2 (1990), pp. 250–256.

[108] M. R. Hestenes and E. Stiefel. "Methods of Conjugate Gradients for Solving Linear Systems". In: *Journal of Research of the National Bureau of Standards* 49.6 (Dec. 1952), pp. 409–436.

[109] M. Hintermüller and T. Wu. "Nonconvex TV^q-Models in Image Restoration: Analysis and a Trust-Region Regularization–Based Superlinearly Convergent Solver". In: *SIAM Journal on Imaging Sciences* 6.3 (2013), pp. 1385–1415.

[110] P. W. Hollanda and R. E. Welsch. "Robust regression using iteratively reweighted least-squares". In: *Communications in Statistics - Theory and Methods* 6.9 (1977), pp. 813–827.

[111] M. Holler and K. Kunisch. "On infimal convolution of total variation type functionals and applications". In: *SIAM Journal on Imaging Sciences* 7.4 (2014), pp. 2258–2300.

[112] M. Hügel, H. Rauhut, and T. Strohmer. "Remote sensing via l_1-minimization". In: *Foundations of Computational Mathematics* 14.1 (2014), pp. 115–150.

[113] M. D. Iordache, J. M. Bioucas-Dias, and A. Plaza. "Collaborative Sparse Regression for Hyperspectral Unmixing". In: *IEEE Transactions on Geoscience and Remote Sensing* 52.1 (Jan. 2014), pp. 341–354. ISSN: 0196-2892.

[114] K. Ito and K. Kunisch. "A variational approach to sparsity optimization based on Lagrange multiplier theory". In: *Inverse problems* 30.1 (2014), p. 015001.

[115] A. Iwasaki, N. Ohgi, J. Tanii, T. Kawashima, and H. Inada. "Hyperspectral Imager Suite (HISUI) -Japanese hyper-multi spectral radiometer". In: *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. July 2011, pp. 1025–1028.

[116] B. S. Kashin. "Diameters of some finite-dimensional sets and classes of smooth functions". In: *Izvestiya Rossiiskoi Akademii Nauk. Seriya Matematicheskaya* 41.2 (1977), pp. 334–351.

[117] J. T. King. "A minimal error conjugate gradient method for ill-posed problems". In: *Journal of Optimization Theory and Applications* 60.2 (1989), pp. 297–304. ISSN: 0022-3239.

[118] F. Krahmer, S. Mendelson, and H. Rauhut. "Suprema of chaos processes and the restricted isometry property". In: *Comm. Pure Appl. Math.* 67.11 (2014), pp. 1877–1904.

[119] V. Kumar, A. Grama, A. Gupta, and G. Karypis. *Introduction to parallel computing: design and analysis of algorithms*. Vol. 400. Benjamin/Cummings Redwood City, CA, 1994.

[120] M.-J. Lai, Y. Xu, and W. Yin. "Improved Iteratively Reweighted Least Squares for Unconstrained Smoothed l_q Minimization". In: *SIAM Journal on Numerical Analysis* 51.2 (2013), pp. 927–257.

[121] T.-H. Lai and S. Sahni. "Anomalies in parallel branch-and-bound algorithms". In: *Communications of the ACM* 27.6 (1984), pp. 594–602.

[122] L. Landweber. "An iteration formula for Fredholm integral equations of the first kind". In: *American journal of mathematics* (1951), pp. 615–624.

[123] C. L. Lawson. *Contributions to the Theory of Linear Least Maximum Approximation*. Ph.D. thesis. University of California, Los Angeles, 1961.

[124] F. T. Leighton. *Introduction to parallel algorithms and architectures: Arrays· trees· hypercubes*. Elsevier, 2014.

[125] L.J.P. van der Maaten, E.O. Postma, and H. J. van den Herik. *Dimensionality Reduction: A Comparative Review*. Tech. rep. TiCC-TR 2009-005. Tilburg University Technical Report, 2009.

[126] B. Logan. "Properties of High-Pass Signals". PhD thesis. New York: Columbia University, 1965.

[127] I. Loris. "On the performance of algorithms for the minimization of l1-penalized functionals". In: *Inverse Problems* 25.3 (2009), p. 035008.

[128] LRZ. *SuperMuc Petascale System*. `https://www.lrz.de/services/compute/supermuc/systemdescription/`. 2015.

[129] S. Lu and S. V. Pereverzev. "Multi-parameter regularization and its numerical realization". In: *Numerische Mathematik* 118.1 (2011), pp. 1–31.

[130] M. Lustig, D. L. Donoho, J. M. Santos, and J. M. Pauly. "Compressed sensing MRI". In: *Signal Processing Magazine, IEEE* 25.2 (2008), pp. 72–82.

[131] M. Lustig, D. Donoho, and J. M. Pauly. "Sparse MRI: The application of compressed sensing for rapid MR imaging". In: *Magnetic resonance in medicine* 58.6 (2007), pp. 1182–1195.

[132] S. G. Mallat. *A Wavelet Tour of Signal Processing: The Sparse Way*. 3rd. Academic Press, 2008.

[133] S. G. Mallat and Z. Zhang. "Matching pursuits with time-frequency dictionaries". In: *IEEE Transactions on Signal Processing* 41.12 (1993), pp. 3397–3415.

[134] Y. Meyer. "Oscillating patterns in image processing and nonlinear evolution equations". In: *AMS University Lecture Series* 22 (2002).

[135] R. Miller and L. Boxer. *Algorithms Sequential & Parallel: A Unified Approach*. Cengage Learning, Dec. 2012. ISBN: 1-133-36680-5.

[136] M. Moeller, T. Wittman, and A. L. Bertozzi. "Variational wavelet pan-sharpening". In: *CAM Report* (2008), pp. 08–81.

[137] G. Moore. "Cramming More Components onto Integrated Circuits". In: *Electronics* 38.8 (Apr. 1965), pp. 114–117. ISSN: 0018-9219.

[138] B. K. Natarajan. "Sparse approximate solutions to linear systems." In: *SIAM Journal on Computing* 24 (1995), pp. 227–234.

[139] V. Naumova and S. V. Pereverzyev. "Multi-penalty regularization with a component-wise penalization". In: *Inverse Problems* 29.7 (2013), p. 075002.

[140] V. Naumova and S. Peter. "Minimization of multi-penalty functionals by alternating iterative thresholding and optimal parameter choices". In: *Inverse Problems* 30.12 (2014), p. 125003.

[141] D. Needell. "Noisy signal recovery via iterative reweighted L1-minimization". In: *Proceedings of the 43rd Asilomar conference on Signals, systems and computers*. Asilomar'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 113–117. ISBN: 978-1-4244-5825-7.

[142] D. Needell and J. A. Tropp. "CoSaMP: iterative signal recovery from incomplete and inaccurate samples". In: *Communications of the ACM* 53.12 (Dec. 2010), pp. 93–100. ISSN: 0001-0782.

[143] J. A. Nelder and R. Mead. "A Simplex Method for Function Minimization". en. In: *The Computer Journal* 7.4 (Jan. 1965), pp. 308–313. ISSN: 0010-4620, 1460-2067.

[144] Y. Nesterov. "Smooth minimization of non-smooth functions". In: *Mathematical programming* 103.1 (2005), pp. 127–152.

[145] Y. Nesterov, A. Nemirovskii, and Y. Ye. *Interior-point polynomial algorithms in convex programming*. Vol. 13. SIAM, 1994.

[146] Netflix, Inc. *Netflix Prize*. `http://www.netflixprize.com`. 2009.

[147] J. Nocedal and S. J. Wright. *Numerical optimization*. Second. Springer Series in Operations Research and Financial Engineering. New York: Springer, 2006.

[148] P. Ochs, A. Dosovitskiy, T. Brox, and T. Pock. "On iteratively reweighted algorithms for nonsmooth nonconvex optimization in computer vision". In: *SIAM Journal on Imaging Sciences* 8.1 (2015), pp. 331–372.

[149] M. R. Osborne. *Finite algorithms in optimization and data analysis*. Wiley Series in Probability and Mathematical Statistics: Applied Probability and Statistics. John Wiley & Sons, Ltd., Chichester, 1985. ISBN: 0-471-90539-9.

[150] M. R. Osborne, B. Presnell, and B. A. Turlach. "A new approach to variable selection in least squares problems". In: *IMA Journal of Numerical Analysis-Institute of Mathematics and its Applications* 20.3 (2000), pp. 389–404.

[151] M. R. Osborne, B. Presnell, and B. A. Turlach. "On the lasso and its dual". In: *Journal of Computational and Graphical statistics* 9.2 (2000), pp. 319–337.

[152] X. Otazu, M. González-Audicana, O. Fors, and J. Núñez. "Introduction of sensor spectral response into image fusion methods. Application to wavelet-based methods". In: *IEEE Transactions on Geoscience and Remote Sensing* 43.10 (2005), pp. 2376–2385.

[153] D. Parkinson. "Parallel efficiency can be greater than unity". In: *Parallel Computing* 3.3 (1986), pp. 261–262.

[154] Z. Peng, M. Yan, and W. Yin. "Parallel and distributed sparse optimization". In: *2013 Asilomar Conference on Signals, Systems and Computers*. Nov. 2013, pp. 659–646.

[155] S. Peter, M. Artina, and M. Fornasier. "Damping noise-folding and enhanced support recovery in compressed sensing". In: *IEEE Transactions on Signal Processing* 63.22 (Nov. 2015), pp. 5990–6002.

[156] K. Puljic and R. Manger. "A distributed evolutionary algorithm with a super-linear speedup for solving the vehicle routing problem". In: *Computing and Informatics* 31.3 (2012), p. 675.

[157] A. Quarteroni, R. Sacco, and F. Saleri. *Numerical Mathematics*. Texts in Applied Mathematics Series. Springer-Verlag GmbH, 2000. ISBN: 978-0-387-98959-4.

[158] R. Ramlau and C. A. Zarzer. "On the minimization of a Tikhonov functional with a non-convex sparsity constraint". In: *Electronic Transactions on Numerical Analysis* 39 (2012), pp. 476–507. ISSN: 1068-9613.

[159] H. Rauhut. "Compressive sensing and structured random matrices". In: *Theoretical foundations and numerical methods for sparse recovery*. Ed. by M. Fornasier. Vol. 9. Radon Series Comp. Appl. Math. deGruyter, 2010, pp. 1–92.

[160] M. Rudelson and R. Vershynin. "On sparse reconstruction from Fourier and Gaussian measurements". In: *Comm. Pure Appl. Math.* 61 (2008), pp. 1025–1045.

[161] F. Santosa and W. Symes. "Linear Inversion of Band-Limited Reflection Seismograms". In: *SIAM Journal on Scientific and Statistical Computing* 7.4 (Oct. 1986), pp. 1307–1330. ISSN: 0196-5204.

[162] K. Schnass. "Dictionary Identification-Sparse Matrix-Factorisation via l_1-Minimisation". In: *IEEE Transactions on Information Theory* 56.7 (2010), pp. 3523–3539.

[163] K. Schnass and P. Vandergheynst. "Dictionary preconditioning for greedy algorithms". In: *Signal Processing, IEEE Transactions on* 56.5 (2008), pp. 1994–2002.

[164] V. Shah, N. Younan, and R. King. "An Efficient Pan-Sharpening Method via a Combined Adaptive PCA Approach and Contourlets". In: *IEEE Transactions on Geoscience and Remote Sensing* 46.5 (May 2008), pp. 1323–1335. ISSN: 0196-2892.

[165] C. E. Shannon. "A mathematical theory of communication". In: *Bell System Technical Journal* 27 (1948), pp. 379–423, 623–656.

[166] C. E. Shannon. "Communication in the presence of noise". In: *Proceedings of the IRE* 37.1 (1949), pp. 10–21.

[167] R. Shonkwiler. "Parallel Genetic Algorithms." In: *ICGA*. Citeseer, 1993, pp. 199–205.

[168] J. Sigl. "Quasi-linear compressed sensing". Masterthesis. Munich: Technical University of Munich, 2013.

[169] J.-L. Starck, D. L. Donoho, and E. J. Candès. "Astronomical image representation by the curvelet transform". In: *Astronomy & Astrophysics* 398.2 (2003), pp. 785–800.

[170] J.-L. Starck, M. Elad, and D. Donoho. "Redundant multiscale transforms and their application for morphological component separation". In: *Advances in Imaging and Electron Physics* 132.82 (2004), pp. 287–348.

[171] J.-L. Starck, M. Elad, and D. L. Donoho. "Image decomposition via the combination of sparse representations and a variational approach". In: *Image Processing, IEEE Transactions on* 14.10 (2005), pp. 1570–1582.

[172] J.-L. Starck, M. K. Nguyen, and F. Murtagh. "Wavelets and curvelets for image deconvolution: a combined approach". In: *Signal Processing* 83.10 (2003), pp. 2279–2283.

[173] M. Stojnic, W. Xu, A. S. Avestimehr, and B. Hassibi. "Compressed sensing of approximately sparse signals". In: *ISIT*. 2008, pp. 2182–2186.

[174] E. Strohmaier, J. Dongarra, H. Simon, and M. Meurer. *Top 500 Supercomputing Sites*. `http://www.top500.org`. Nov. 2015.

[175] T. Stuffler, K. Förster, S. Hofer, M. Leipold, B. Sang, H. Kaufmann, B. Penné, A. Mueller, and C. Chlebek. "Hyperspectral imagingtext–An advanced instrument concept for the EnMAP mission (Environmental Mapping and Analysis Programme)". In: *Acta Astronautica* 65.7–8 (Oct. 2009), pp. 1107–1112. ISSN: 0094-5765.

[176] H. L. Taylor, S. C. Banks, and J. F. McCoy. "Deconvolution with the l1-norm". In: *Geophysics* 44.1 (1979), pp. 39–52. ISSN: 0016-8033, 1942-2156.

[177] R. Tibshirani. "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1996), pp. 267–288.

[178] J. Treichler, M. A. Davenport, and R. G. Baraniuk. "Application of compressive sensing to the design of wideband signal acquisition receivers". In: *6th U.S. / Australia Joint Workshop on Defense Applications of Signal Processing (DASP)*. Lihue, Hawaii, Sept. 2009.

[179] J. A. Tropp. "Greed is good: Algorithmic results for sparse approximation". In: *IEEE Transactions on Information Theory* 50.10 (2004), pp. 2231–2242.

[180] T.-M. Tu, S.-C. Su, H.-C. Shyu, and P. S. Huang. "A new look at IHS-like image fusion methods". In: *Information Fusion* 2.3 (Sept. 2001), pp. 177–186. ISSN: 1566-2535.

[181] E. Van Den Berg and M. P. Friedlander. "Probing the Pareto frontier for basis pursuit solutions". In: *SIAM Journal on Scientific Computing* 31.2 (2008), pp. 890–912.

[182] L. J. P. van der Maaten. *Matlab Toolbox for Dimensionality Reduction*. `https://lvdmaaten.github.io/drtoolbox/`. 2016.

[183] L. J. P. van der Maaten and G. E. Hinton. "Visualizing High-Dimensional Data Using t-SNE". In: *Journal of Machine Learning Research* 9.TiCC-TR 2009-005 (Nov. 2008), pp. 2579–2605.

[184] O. M. L. Vance Faber. "Superlinear Speedup of an Efficient Algorithm Is Not Possible". In: *Parallel Computing* 3.3 (1986), pp. 259–260. ISSN: 0167-8191.

[185] R. Vershynin. "Introduction to the non-asymptotic analysis of random matrices". In: *arXiv preprint arXiv:1011.3027* (2010).

[186] L. Vese and S. Osher. "Image denoising and decomposition with Total Variation minimization and oscillatory functions". In: *Journal of Mathematical Imaging and Vision* 20 (2004), pp. 7–18.

[187] L. Vese and S. Osher. "Modeling textures with Total Variation minimization and oscillating patterns in image processing". In: *Journal of Scientific Computing* 19 (2003), pp. 553–572.

[188] C. R. Vogel and M. E. Oman. "Fast, robust total variation-based reconstruction of noisy, blurred images." English. In: *IEEE Transactions on Image Processing* 7.6 (1998), pp. 813–824. ISSN: 1057-7149.

[189] S. Voronin. "Regularization of Linear Systems with Sparsity Constraints with Applications to Large Scale Inverse Problems". PhD thesis. Applied and Computational Mathematics Department, Princeton University, 2012.

[190] S. Voronin and I. Daubechies. "An Iteratively Reweighted Least Squares Algorithm for Sparse Regularization". In: *arXiv:1511.08970 [math]* (Nov. 2015). arXiv: `1511.08970 [math]`.

[191] W. Wang, S. Lu, H. Mao, and J. Cheng. "Multi-parameter Tikhonov regularization with L0 sparsity constraint". In: *Inverse Problems* 29 (2013), p. 065018.

[192] B. Wilkinson and M. Allen. *Parallel programming*. Prentice hall Upper Saddle River, NJ, 1999.

[193] Z. Xu, X. Chang, F. Xu, and H. Zhang. "L1/2 Regularization: A Thresholding Representation Theory and a Fast Solver". In: *IEEE Transactions on Neural Networks and Learning Systems* 23.7 (2012), pp. 1013–1027.

[194] N. Yokoya, T. Yairi, and A. Iwasaki. "Coupled Nonnegative Matrix Factorization Unmixing for Hyperspectral and Multispectral Data Fusion". In: *IEEE Transactions on Geoscience and Remote Sensing* 50.2 (2012), pp. 528–537.

[195] C. A. Zarzer. "On Tikhonov regularization with non-convex sparsity constraints". In: *Inverse Problems* 25.2 (2009), pp. 025006, 13. ISSN: 0266-5611.

[196] J. Zeng, S. Lin, Y. Wang, and Z. Xu. "Regularization: Convergence of Iterative Half Thresholding Algorithm". In: *IEEE Transactions on Signal Processing* 62.9 (May 2014), pp. 2317–2329. ISSN: 1053-587X.

[197] Y. Zhang, S. D. Backer, and P. Scheunders. "Noise-Resistant Wavelet-Based Bayesian Fusion of Multispectral and Hyperspectral Images". In: *IEEE Transactions on Geoscience and Remote Sensing* 47.11 (Nov. 2009), pp. 3834–3843.

[198] Y. Zhang and M. He. "Multi-Spectral and Hyperspectral Image Fusion Using 3-D Wavelet Transform". In: *Journal of Electronics (China)* 24.2 (Mar. 2007), pp. 218–224.

[199] Y. Zhang, Y. Cui, and B. He. "Gif-Based Least Square Method for Hyperspectral and Multispectral Data Fusion". In: *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE, 2013.

[200] Y. Zhang. "Problems in the Fusion of Commercial High-Resolution Satellites Images as well as LANDSAT 7 Images and Initial Solutions". In: *Proceedings of the ISPRS, CIG, and SDH Joint International Symposium on Geospatial Theory, Processing and Applications*. 2002, pp. 9–12.

[201] B. Zhou, L. Gao, and Y.-H. Dai. "Gradient Methods with Adaptive Step-Sizes". en. In: *Computational Optimization and Applications* 35.1 (Mar. 2006), pp. 69–86. ISSN: 0926-6003, 1573-2894.

[202] X. X. Zhu and R. Bamler. "A Sparse Image Fusion Algorithm With Application to Pan-Sharpening". In: *IEEE Transactions on Geoscience and Remote Sensing* 51.5 (2013), pp. 2827–2836. ISSN: 0196-2892.

[203] X. X. Zhu and R. Bamler. "Tomographic SAR inversion by L1-norm regularization—The compressive sensing approach". In: *IEEE Transactions on Geoscience and Remote Sensing* 48.10 (2010), pp. 3839–3846.

[204] X. X. Zhu and R. Bamler. "Tomographic SAR inversion from mixed repeat- and single-Pass data stacks – the TerraSAR-X/TanDEM-X case". In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Melbourne, Australia, 2012, pp. 97–102.

[205] X. X. Zhu, Claas Grohnfeldt, and R. Bamler. "Exploiting Joint Sparsity for Pan-sharpening textendash The J-SparseFI Algorithm". In: *IEEE Transactions on Geoscience and Remote Sensing* (2015). Ed. by A. J. Plaza. ISSN: 0196-2892.

[206] X. X. Zhu, C. Grohnfeldt, and R. Bamler. "Collaborative sparse image fusion with application to pan-sharpening". In: *2013 18th International Conference on Digital Signal Processing (DSP)*. July 2013, pp. 1–6.

[207] X. X. Zhu, C. Grohnfeldt, and R. Bamler. "Collaborative sparse reconstruction for pan-sharpening". In: *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. July 2013, pp. 868–871.

[208] X. X. Zhu, S. Spiridonova, and R. Bamler. "A pan-sharpening algorithm based on joint sparsity". In: *Advances in Radar and Remote Sensing (TyWRRS), 2012 Tyrrhenian Workshop on*. 2012, pp. 177–184.

[209] X. X. Zhu, X. Wang, and R. Bamler. "Compressive sensing for image fusion - with application to pan-sharpening". In: *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. July 2011, pp. 2793–2796.

Danksagung/Acknowledgements

Diese Arbeit wäre nicht ohne die großzügige finanzielle und administrative Unterstützung von Munich Aerospace möglich gewesen. Insbesondere danke ich den Verantwortlichen im Allgemeinen für die stete Verbesserung der Situation von uns Stipendiaten und im Speziellen für die Finanzierung meines Forschungsaufenthaltes in der USA.

I would like to thank my supervisor Prof. Dr. Massimo Fornasier for his great support in all scientific questions and his guidance through the world of research. His high level of confidence in my work was essential for my personal development.

Sincerely I would like to express my gratitude to Prof. Dr. Xiao Xiang Zhu and the SiPEO Team from DLR for the fruitful technical discussions and feedback, the continuous exchange of ideas, and the provision and preparation of satellite data. I also thank the Gauss Centre for Supercomputing e.V. for providing computing time on the GCS Supercomputer SuperMUC at the Leibniz Supercomputing Centre (LRZ).

I would like to thank Dr. Valeriya Naumova, Dr. Marco Artina and Stephan Worm for the successful scientific collaboration, as well as my M15 team members, in particular Juliane Sigl, Dr. Giacomo Albi, Mattia Bongini, and Dr. Benjamin Scharf for the nice time together at TUM and for the numerous conversations far beyond mathematics. Special thanks goes to Prof. Dr. Holger Rauhut for his invitation to a research stay in Aachen, and the numerous discussions and hints, which turned out to be very fruitful for our joint paper.

I am greatly in debt with Prof. Dr. Richard Baraniuk and the DSP group at Rice University in Houston, who gave me the opportunity to get to know and understand a different country, society, and academic environment; an experience that enormously broadened my horizon. In this regard I also thank Prof. Dr. Marc Davenport and Prof. Dr. Tom Goldstein for their scientific discussions and help.

I thank Dr. Colas Schretter for his cooperation and interesting ideas on iteratively reweighted least squares methods.

Ein ganz besonderer Dank geht an meinen Freund, Mitbewohner und Projektpartner Claas, der mich für den Abschnitt „Promotion in München“ geworben hat, diesen mit mir gemeinsam privat und professionell angegangen ist und mit dem ich in den letzten vier Jahren viele Höhen und Täler zusammen durchschritten habe. Nicht zuletzt die vertrauensvolle und verlässliche Zusammenarbeit hat wesentlich zur Erstellung dieser Dissertation beigetragen.

Ich danke meinen Freunden Mischa und Alexander für ihre Korrekturen und die

Bereitschaft, sich mit der faszinierenden Welt des Sparse Recovery auseinander zu setzen.

Mein Dank gebührt allen lieben Menschen, die mich in den letzten Jahren auf verschiedenste Art und Weise unterstützt und immer wieder angetrieben haben.

Ich danke Julia dafür, dass sie mit mir diese aufreibende und aufregende Zeit gemeinsam durchlebt hat und mir die letzten Monate an allen Ecken und Enden auf ihre Weise erleichtert hat. Sie weiß besser als ich, wann eine Pause meine Akkus wieder auflädt.

Zu guter Letzt möchte ich meiner Familie danken, die bereits früh auf mich verzichten musste, damit ich mich leidenschaftlich der Mathematik widmen konnte und die meinen Weg immer mit uneingeschränktem Vertrauen unterstützt hat.