Computational Aspects of Compressed Sensing

Vom Fachbereich Mathematik
der Technischen Universität Darmstadt
zur Erlangung des Grades eines

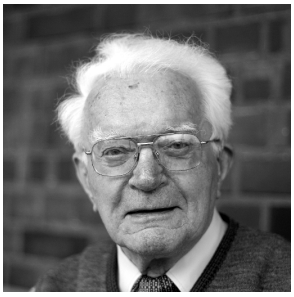
Doktors der Naturwissenschaften
(Dr. rer. nat.)

genehmigte
Dissertation

von
Dipl.-Math. Oec. Andreas M. Tillmann
aus Braunschweig

Referent:	Prof. Dr. Marc E. Pfetsch
1. Korreferent:	Prof. Dr. Dirk A. Lorenz
2. Korreferent:	Prof. Stephen J. Wright, PhD
Tag der Einreichung:	30. September 2013
Tag der mündlichen Prüfung:	16. Dezember 2013

Darmstadt 2013
D 17



Meinem Großvater

– *Gaya Scienza!*

Acknowledgments

Various people contributed in one way or another to the successful completion of my doctorate. I appreciate all their support and efforts and would like to express my sincere gratitude.

First and foremost, I thank my advisor Marc Pfetsch for suggesting Sparse Recovery as a field of research for my dissertation, for his advice and intuition regarding questions worth investigating and, in particular, the freedom he granted me to pursue my own ideas and interests.

I also very much enjoyed working with Dirk Lorenz and would like to thank him for his advice and help over the years, stimulating discussions and encouraging feedback, and for gladly agreeing to be a referee for my thesis. Moreover, I thank Steve Wright for going to the time and effort of reviewing my dissertation. Also, I am grateful to Stefan Ulbrich and Herbert Egger for serving on my thesis committee, and to Karsten Große-Brauckmann for acting as committee chair.

I enjoyed the pleasant working environments both in Braunschweig and in Darmstadt, for which I would like to thank the Institute for Mathematical Optimization at TU Braunschweig (in particular, Ronny Hansmann and Thomas Rieger) and the Research Group Optimization at TU Darmstadt. Special thanks go out to Kai "5 minutes later" Habermehl for the friendly competition and quips during the (more or less) simultaneous completion of our theses, and to Katrin Herr for the help with creating the cross-polytope image for the front cover.

Furthermore, I thankfully acknowledge the financial support I received from the Deutsche Forschungsgemeinschaft (DFG) during my time on the "SPEAR – Sparse Exact and Approximate Recovery" research project (2011–2013), and thank Marc and Dirk again for bringing this project into being in the first place.

Last, but certainly not least, my deepest gratitude goes to my family for their continuous support over the years—in particular, many thanks to Imke! Not only has she carefully read the whole manuscript (and her feedback certainly helped improve it), but in fact, had she not been there to care for me during the past six months, I probably would never have left our apartment at all anymore, slowly

melting into the couch, incessantly mumbling grumpy mumblings, covered by an increasingly thick layer of dust, with the laptop growing stuck to my ... well, lap, I guess. Actually, no—I would have starved long before any of that. So, thank you very much indeed, and it will be my pleasure to return the favor!

Zusammenfassung

Angesichts der fortschreitenden Digitalisierung in nahezu allen Bereichen des täglichen Lebens bedarf es mehr denn je effizienter Techniken zur Akquise, Speicherung und Verarbeitung digitaler Daten und Signale. Das noch relativ junge Forschungsgebiet des *Compressed Sensing (CS)* verfolgt diesbezüglich – wie der Name bereits vermuten lässt – die Idee, die Schritte der Datengewinnung und Kompression zu kombinieren. Ermöglicht wird dies durch Ausnutzung der Tatsache, dass viele verschiedene Typen von digitalen Signalen eine *dünnbesetzte Darstellung* besitzen, d.h., sie lassen sich gut oder sogar exakt durch wesentlich weniger Koeffizienten beschreiben als ihre Dimensionalität vermuten ließe. Diese Eigenschaft impliziert hohe Komprimierbarkeit, da die wesentlichen Informationen auf einen kleinen Teil der gesamten Datenmenge konzentriert sind. Die CS-Theorie basiert essentiell auf der Feststellung, dass sich solch dünnbesetzte Signale unter bestimmten Voraussetzungen bereits aus weit weniger Messungen (mit entsprechend verkürztem Datengewinnungsprozess) als traditionell angenommen sehr gut rekonstruieren lassen.

Die vorliegende Dissertation behandelt (vorrangig) einige grundlegende Aspekte der Compressed-Sensing-orientierten Rekonstruierbarkeit dünnbesetzter Signale aus Sicht der mathematischen Optimierung und Komplexitätstheorie.

Die hierbei zentralen Rekonstruktionsprobleme lassen sich mathematisch als die Suche nach einer (exakten oder näherungsweisen) Lösung eines unterbestimmten linearen Gleichungssystems formulieren, welche die wenigsten von Null verschiedenen Einträge aufweist. Aus komplexitätstheoretischer Sicht ist diese Art von Problemen im Allgemeinen „schwer“, d.h., es sind keine Lösungsalgorithmen bekannt, die beliebige Probleminstanzen schnell lösen könnten – tatsächlich gilt selbst die Existenz solcher Verfahren weitläufig als unmöglich. Zu den bahnbrechenden Entdeckungen der CS-Forschung zählt daher die Identifizierung gewisser Voraussetzungen, unter denen eine dünnbesetzte Lösung mit *effizienten* Algorithmen bestimmt werden kann.

Wir untersuchen zunächst die rechnerische Komplexität diverser solcher Bedingungen für Eindeutigkeit und effiziente Rekonstruierbarkeit exakt oder approximativ dünnbesetzter Signale. Insbesondere zeigen wir, dass die Erfüllung der wichtigsten dieser Bedingungen selbst schwer nachweisbar ist. Dies wurde in der CS-

Forschungsgemeinde zwar bereits seit Längerem vermutet, jedoch bisher nicht mathematisch rigoros bewiesen. Unsere Argumente zeigen Verbindungen zur Matroid- und Graphentheorie; zudem diskutieren wir diverse verwandte Probleme und Komplexitätsergebnisse.

Anschließend gilt unser Augenmerk einem der beliebtesten Ansätze zur Ermittlung dünnbesetzter Lösungen unterbestimmter linearer Gleichungssysteme, namentlich *Basis Pursuit* (kurz: BP). Hierbei wird das eigentliche Ziel – die Minimierung der Nicht-Null-Einträge in der Lösung – ersetzt durch die Minimierung der Summe der Absolutbeträge (d.h. der ℓ_1-Norm) des Lösungsvektors.

Der empirisch und theoretisch belegte Erfolg dieser Vorgehensweise hatte die Entwicklung einer Vielzahl verschiedener Lösungsmethoden zur Folge. Um herauszufinden, welcher algorithmische Ansatz „der beste“ ist, erstellen wir umfangreiche Testdaten mit bekannten Optimallösungen und führen darauf einen detaillierten numerischen Vergleich etablierter BP-Löser durch.

In diesem Rahmen stellen wir zudem einen heuristischen Optimalitätstest vor, der ermöglicht, frühzeitig zur (exakten) Optimallösung zu „springen“. Wir geben theoretische Erfolgsgarantien für diesen Test und belegen anhand weiterer Rechnungen, dass sich damit sowohl die Laufzeit als auch die Ergebnisgenauigkeit diverser Löser deutlich verbessern lässt.

Des Weiteren zeigen wir, dass das BP-Problem im Wesentlichen äquivalent zu linearer Programmierung ist; während man schon lange weiß, dass Basis Pursuit als lineares Programm formuliert werden kann, war die umgekehrte Richtung offenbar bisher nicht bekannt. Darüber hinaus diskutieren wir Varianten des Optimalitätstests für zwei (beide als *Basis Pursuit Denoising* bekannte) ℓ_1-Norm-Minimierungsprobleme, die kleinere Verletzungen der Gleichungsrestriktionen gestatten und daher auf approximativ (nicht exakt) dünnbesetzte Signale beziehungsweise fehlerbehaftete Messungen ausgelegt sind.

In unserem Rechenvergleich wird insbesondere auch ein eigener Löser, ISAL1, betrachtet. Dieser Algorithmus ist eine Spezialisierung eines Lösungsverfahrens (genannt ISA) für allgemeine konvexe Minimierungsprobleme mit Nebenbedingungen, das wir im letzten Kapitel des Hauptteils dieser Dissertation vorstellen. ISA ist eine Abwandlung der klassischen projizierten Subgradientenmethode aus der nichtglatten Optimierung, in der wir die üblichen exakten Projektionen der Iterationspunkte auf die zulässige Menge durch adaptive approximative Projektionen ersetzen. Wir beweisen die Konvergenz des ISA-Verfahrens für verschiedene Arten von Schrittweiten, geben diverse Beispielkonstruktionen adaptiver Projektionsoperatoren und diskutieren insbesondere die Konkretisierung der allgemeinen Methode hinsichtlich Basis Pursuit (und BP Denoising Varianten), sowie unsere Implementierung ISAL1 des hieraus resultierenden Algorithmus.

Abschließend erläutern wir offene Fragen und potentielle Erweiterungen bezüglich der in der Dissertation behandelten Aspekte und geben einen Ausblick auf mögliche weiterführende oder verwandte Forschungsthemen.

Contents

CHAPTER 1

Introduction

In today's world, we face a huge and ever-increasing amount of high-dimensional digital data and the challenges arising from the need to effectively acquire, store and process such data. With respect to these tasks, a crucial observation is that one often does not require the full amount of information contained in some digital object to be able to recognize, or reconstruct, its content. In other words, a suitable representation of the data is often *sparse* in the sense that it contains only a few significant components that already capture most of the information, which opens possibilities for data reduction.

For instance, many signal compression methods selectively discard large parts of the data in order to retain a suitable approximation of the original. Such compression is called "lossy", since parts of the true data are irretrievably lost and are not contained in the compressed signal. Thus, there clearly is a trade-off between the retained amount of information and the quality of the resulting approximate signal. Familiar examples for lossy compression are the ubiquitous JPEG compression standard from image processing, video codecs like MPEG, or the MP3 audio compression. All of them can provide signal representations that are nearly indistinguishable from the original digital object by human perception but require dramatically fewer storage space.

The observation that it seems wasteful to first collect the complete data set and then just neglect large portions of it (to obtain the desired compressed object) inspired the development of a paradigm now known as *Compressed Sensing (CS)*, or *Compressive Sampling*, see, e.g., [89, 47, 49, 52, 109, 165, 111]. Here, the idea is to perform a sort of compression already during data acquisition. Under the assumption that a signal indeed has an exact or approximate sparse representation, this can be accomplished by taking only a surprisingly small number (compared

to the full information content) of linear measurements, which still allow for the recovery of the significant signal features but circumvent the traditional two-stage sampling and compression process. Importantly, the reconstruction can in fact be achieved by efficient algorithms, under certain conditions on signal sparsity and the conditioning of the sampling operations.

Since its development about a decade ago, the CS concept has already been successfully employed in a wide range of sampling and signal processing applications. One particular well-known example is biomedical image processing, or more precisely, (X-ray) computed tomography and magnetic resonance imaging; see, e.g., [178, 179, 169]. Here, one wishes to acquire sharp high-resolution (cross-sectional) images of certain inner parts of the body, say, the brain. The image quality is clearly decisive for accurate diagnosis, e.g., early identification or precise localization of tumor cells. Compressed Sensing methodologies can help speed up the scanning process while maintaining a high quality of the results, which is desirable for economic purposes as well as to reduce the patients' exposure to harmful radiation. Other examples of the application of CS and sparse recovery results can be found in radiotherapy treatment planning [268], astronomical signal processing [33] and visualization of outer-space phenomena [254], geophysical/seismic imaging [132], radar [131, 214], error correction in data transmission [50], audio source separation [210], morphological component analysis of images and other image processing tasks such as denoising and inpainting (cf., for instance, [231, 100]), to name just a few.

It is nigh impossible to provide a comprehensive overview of the vast amount of research on sparse representations, sensing and recovery schemes, CS applications and extensions that was conducted over the last years. In the following, we will therefore only touch upon those aspects that are closely tied to the scope of this thesis: Computational aspects of (efficient) sparse reconstruction, in the usual finite-dimensional real-valued setting. (Although it is noteworthy that some CS-based results extend to infinite dimensions and/or the field of complex numbers, we will not concern ourselves with such extensions.) For more general introductions to the field of Compressed Sensing and its connections to sparse representation theory and other fields like sampling or nonlinear approximation theory, we refer to the recent first books on the subject, [111, 165, 108, 100], and also mention the articles [88, 89, 47, 52, 40, 109, 64]; many more papers and other material can be found online, e.g., at the CS Resources webpage of the Rice University [73], the Nuit Blanche blog [53] and the Wiki pages [230].

1.1 Finding Sparse Exact and Approximate Solutions to Underdetermined Linear Equation Systems

Besides the original name-giving principle, the term Compressed Sensing is now often (but somewhat imprecisely) used as a general catchphrase for a broad variety of related aspects regarding sparse representations of digital signals and their efficient recovery. This can be explained by the paramount importance of these latter concepts throughout CS theory, and in particular, of the problem to find a sparsest solution to an underdetermined linear equation system, i.e.,

$$\min \|x\|_0 \quad \text{s.t.} \quad Ax = b, \tag{P_0}$$

where $A \in \mathbb{R}^{m \times n}$ with $m \leq n$, $b \in \mathbb{R}^m$, and $\|x\|_0$ denotes the number of nonzeros in a vector x. Typically, the matrix A is assumed to have full rank m (because otherwise $Ax = b$ might not have a solution at all), and $m < n$. Then, the system $Ax = b$ has infinitely many solutions, and (P_0) seeks one with as few nonzero entries as possible. The idea is that if the original signal x^* of interest is highly sparse then it can be encoded, or "compressively sampled", by $b := Ax^*$ and recovered as the unique solution of (P_0).

In practice, a signal will hardly be exactly sparse (such that most entries are truly equal to zero), but instead only *compressible*, i.e., approximately sparse in the sense that it contains many insignificant entries that can be dropped without sacrificing too much valuable information. The following model adapts to this setting by relaxing the equality constraint to allow small (ℓ_2-norm) deviations:

$$\min \|x\|_0 \quad \text{s.t.} \quad \|Ax - b\|_2 \leq \delta, \tag{P_0^δ}$$

for some $\delta > 0$. (Clearly, for $\delta = 0$, we obtain (P_0) again.) Often, this second model is also used in the context of *denoising*, where one assumes that the linear measurements Ax^* are contaminated by additive noise, i.e., $b = Ax^* + r$ with a noise vector r. Appropriate choices for δ are often available from the application, e.g., from certain statistical properties of the noise vector r. (Note that, depending on the noise model, other norm choices in the constraint of (P_0^δ) may be preferable; see, e.g., [104, Section 4.3].) Since solution uniqueness is not as well-defined for (P_0^δ) as for the exact sparse recovery problem (P_0)—tiny perturbations of the nonzero entries of an optimal solution typically yield alternate optimal solutions—the focus here lies on *stable and robust recovery*: We wish to obtain a solution that is reliably close to the (unknown) original signal x^* both in terms of sparsity and the magnitudes of its entries; see, e.g., [91, 64, 100, 165] for detailed discussions.

Remark 1.1. As alluded to earlier, certain classes of signals are expected to be sparsely representable—at least approximately—in some known (non-canonical) basis B, i.e., not the signal x^* itself but Bx^* has few significant entries. For instance, natural images are well-known to typically have sparse approximate representations with respect to certain wavelet transforms (cf. [182]); this is exploited, e.g., in state-of-the-art image compression or denoising algorithms, see (for example) [231, 100, 87, 165, 182]. With such prior information, a problem of interest could actually read, e.g., $\min\{\, \|Bx\|_0 \,:\, Ax = b \,\}$, and indeed, if B is not a basis, this problem becomes genuinely different from (P_0) (cf. [102, 191]). Similarly, one can concatenate multiple bases into one so-called *dictionary* (i.e., an "overcomplete" matrix with more columns than rows) or perform *dictionary learning* to directly obtain matrices that offer sparse representability for certain signal types from specific tasks such as, e.g., face recognition [1, 255, 238]. Throughout this thesis, we will not delve into the technical aspects of these extensions, or others such as additionally incorporating nonnegativity constraints, and focus our attention on the general problems (P_0) and (P_0^δ).

Unfortunately, both sparse recovery problems (P_0) and (P_0^δ) are generally NP-hard (in the strong sense), see [MP5] in [115], [192, Theorem 1] and [82, Theorem 2.1]. Thus, unless the most widely believed theoretical complexity assumption—P≠NP—turns out to be wrong, there exist no (pseudo-)polynomial-time algorithms to solve either problem. In fact, even approximating a solution is considered intractable, see [6, 7].

The apparently only serious effort to come up with something more sophisticated than plain exhaustive search (through all exponentially many column subsets) to *exactly* solve general instances of (P_0) was made in [143], where this problem (and a linearly-constrained variant of (P_0^δ)) was tackled by a Branch & Cut method applied to an integer programming reformulation. However, this turned out to be reasonably efficient only for very small problem instances and is therefore not suitable for large-scale "real-world" problems. (Preliminary experiments by the author of this thesis, based on a different exact integer programming model of (P_0), were even more discouraging.)

Inspired by the applications in CS, and justified by the computational intractability of (P_0) and (P_0^δ), various (polynomial-time) heuristics for sparse recovery problems have been proposed, along with several theoretical results—*sparse recovery conditions*—on when such methods actually succeed in obtaining or stably approximating the respective sparsest representations. Most of these heuristic approaches can be broadly classified into two main categories: Greedy algorithms or relaxation (and regularization) methods.

A prominent role among the greedy heuristics is taken by the so-called *(Orthogo-*

nal) Matching Pursuit techniques, see, e.g., [183, 83, 207, 239, 94, 196, 195]. Here, the general idea is to approximate the optimal solution by iteratively increasing the *support* $S = \operatorname{supp}(x) \coloneqq \{\, j \in [n] \,:\, x_j \neq 0 \,\}$, starting with the empty set and successively including indices chosen by some locally (but not globally) optimal rule until the requirement $\|Ax - b\|_2 \leq \delta$, or $Ax = b$, is met. For instance, one can pick $j \notin S$ such that the residual error $\|Ax - b\|_2$ achievable by allowing nonzero entries in x only on $S \cup \{j\}$ is reduced as much as possible. The hope is that few iterations suffice to reach the feasibility goal, leading to a sparse solution since one keeps $x_j = 0$ for $j \notin S$ throughout this process.

Relaxation[1] models replace the discrete objective function $\|x\|_0$ by some continuous approximation. For instance, the contribution [184] proposed the function $f_c(x) = \sum_{j=1}^{n}(1 - e^{-c|x_j|})$ for a parameter $c > 0$ (e is the Euler constant), and shows that there exists a value of c for which the solutions of $\min\{\, f_c(x) \,:\, Ax = b \,\}$ and (P_0) coincide.

Using the Euclidean (ℓ_2-)norm, one obtains the so-called "minimum energy" solution, which is useful in many contexts, but usually not sparse at all. The work [58] may be considered a starting point for the intensified research on sparse recovery that eventually also spawned the field of Compressed Sensing; it showed empirically that sparse solutions can sometimes be obtained by solving

$$\min \|x\|_1 \quad \text{s.t.} \quad Ax = b, \tag{P_1}$$

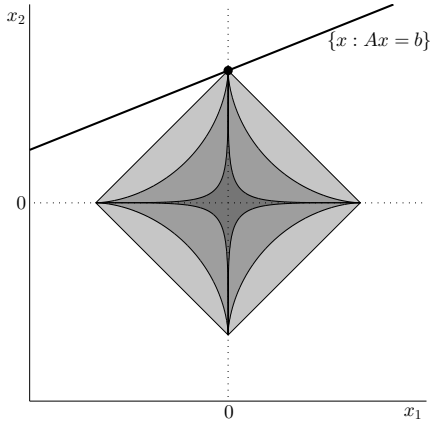
where $\|x\|_1 = \sum_{j=1}^{n}|x_j|$ is the ℓ_1-norm. This problem, called *Basis Pursuit (BP)*, and its denoising counterpart

$$\min \|x\|_1 \quad \text{s.t.} \quad \|Ax - b\|_2 \leq \delta, \tag{P_1^δ}$$

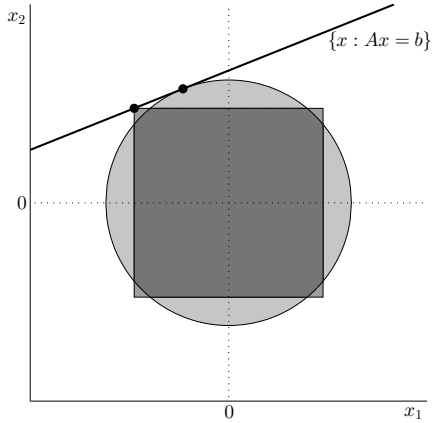
are among the most popular approaches to sparse recovery in CS today and were considered in many ground-breaking works that defined the field, see, e.g., [92, 90, 126, 50, 49]. The phenomenon that the (unique) optimal solution to (P_1) also solves (P_0) is known as ℓ_0-ℓ_1-*equivalence*, and can be brought about by several sparse recovery conditions.

There is a nice geometric intuition for why ℓ_1-minimization promotes sparsity in the solution (see Figure 1.1 for a visualization): The set of solutions of an underdetermined system $Ax = b$ forms an affine subspace of $\mathbb{R}^n$, e.g., a line in $\mathbb{R}^2$. A minimum-norm vector residing in this subspace can be found by increasing the (radius r of the) norm ball $\{\, x \,:\, \|x\| \leq r \,\}$ around the origin until it "touches" the set $\{\, x \,:\, Ax = b \,\}$. Intuitively, the pointedness of the ℓ_1-ball (which is, e.g., diamond-

[1] This use of the term "relaxation" differs from the usual one in optimization theory, where it typically describes relaxation of *constraints*, e.g., replacing $x \in \{0, 1\}$ by $x \in [0, 1]$.



(a) Sparse solution of $Ax = b$ for ℓ_p-minimization with $p \in \{0.3, 0.6, 1\}$ (dark to light shapes, respectively).



(b) Dense solutions of $Ax = b$ for ℓ_p-minimization with $p = 2$ (disc) or $p = \infty$ (square).

Figure 1.1. Geometric intuition: Why ℓ_p-norm minimization promotes solution sparsity for $0 < p \leq 1$, but typically yields dense solutions for $p \geq 2$.

shaped in $\mathbb{R}^2$) therefore leads to a sparse solution. In fact, similar sparsity-inducing properties hold for the ℓ_p-(quasi-)norms $\|x\|_p = (\sum_{j=1}^n |x_j|^p)^{1/p}$ with $0 < p < 1$, cf. Figure 1.1(a); however, these functions are nonconvex and the corresponding problems $\min\{\, \|x\|_p \,:\, Ax = b \,\}$ are actually strongly NP-hard [116], like (P_0) itself. On the other hand, (P_1) is a convex optimization problem (in fact, it can be restated as a linear program), and can thus be solved efficiently. Indeed, it can be seen as the closest convex approximation of (P_0), since $\lim_{p\to 0} \|x\|_p^p = \|x\|_0$. Moreover, despite being nonsmooth and therefore somewhat harder to handle (analytically and algorithmically) than functions which are differentiable everywhere, the ℓ_1-norm is preferable to the other convex ℓ_p-norms with $1 < p < \infty$ with respect to sparse recovery: Increasing the norm balls for the latter norms, one usually reaches $Ax = b$ at a non-sparse point (the same holds true for the ℓ_∞-norm, $\|x\|_\infty = \max\{\, |x_j| \,:\, 1 \leq j \leq n \,\}$); cf. Figure 1.1(b).

The potential of ℓ_1-minimization for CS sparse reconstruction is further illustrated by the synthetic examples depicted in Figures 1.2 and 1.3, respectively: Figure 1.2 shows how a sparse vector can be exactly recovered via (P_1) whereas ℓ_2-minimization yields a poor reconstruction. In Figure 1.3, we see a grayscale image with sparse coefficient vector; we corrupted the original by adding (artificial) noise and see that a very good approximation of the true image can be achieved, using (P_1^δ) with δ attuned to the noise norm, from relatively few linear (Gaussian) measurements of

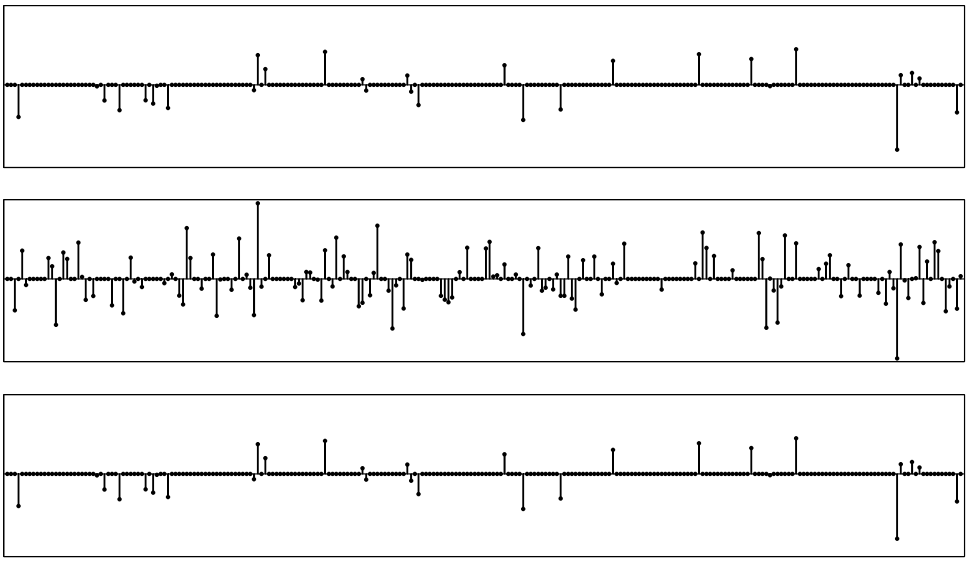


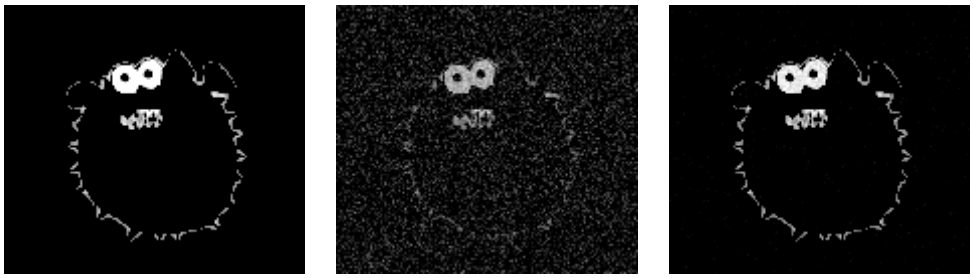
Figure 1.2. A relatively sparse "signal" $x^* \in \mathbb{R}^{256}$ with $\|x^*\|_0 = 30$ (top) and the reconstructions obtained via $\min\{\, \|x\|_2 : Ax = b \,\}$ (middle) and (P_1) (bottom) from 128 (Gaussian) linear measurements $b := Ax^*$. Basis Pursuit recovers x^* exactly, but ℓ_2-minimization fails.

the corrupted image.

While we will refer to (P_1^δ) as *Basis Pursuit Denoising*, this name was originally (in [58]) affixed to the problem

$$\min \tfrac{1}{2}\|Ax - b\|_2^2 + \lambda \|x\|_1 \qquad (\mathrm{QP}_\lambda)$$

(with $\lambda > 0$). In fact, most algorithms for BP Denoising consider the latter formulation because, as an unconstrained problem, it is generally easier. On the other hand, the parameter δ in (P_1^δ) offers a more natural interpretation than λ in (QP_λ), as it directly models the norm ("energy") of the noise vector. For certain λ and δ, the two problems actually become equivalent, but the parameter relationship is implicit (it depends on the respective solutions) and hence generally not known a priori, see, e.g., [245, 107, 173, 111]. The problem (QP_λ) can also be seen as a *regularization* of the inverse problem to recover an unknown x^* from (noisy) incomplete measurements $Ax = b := Ax^*(+r)$: Given the infinite number of possible solutions for an underdetermined linear system (if it has one at all), this task is generally *ill-posed* and we need to impose some assumption, like sparsity, to get a well-defined formulation. As opposed to (P_0) and (P_0^δ), which enforce consistency with the mea-



(a) Original image (b) Corrupted image (c) CS reconstruction

Figure 1.3. Stylized CS denoising problem: The original 134×150 pixel image has only 787 nonzeros among its 20100 coefficients (in the canonical basis). Exploiting sparsity, near-perfect recovery is achieved from linear measurements of only about 20% of the noisy data. (Image courtesy of Mathias Krisp / blowfish.)

surements via explicit constraints, the regularization approach combines the data fidelity term ($\|Ax - b\|_2$) and the regularizing term into a single objective function, and the parameter λ controls the trade-off between the two terms. Indeed, the ℓ_1-regularization term in (QP_λ) serves the same purpose as the ℓ_1-objective in the Basis Pursuit problems, namely, inducing sparsity in the solution.

Several related problems also became prominent in CS applications in statistics and signal processing, and can be used to obtain sparse approximate solutions; for instance, the *Least Absolute Shrinkage and Selection Operator (LASSO)* [235]

$$\min \|Ax - b\|_2 \quad \text{s.t.} \quad \|x\|_1 \leq \tau, \tag{LS_τ}$$

or the *Dantzig Selector* [51]

$$\min \|x\|_1 \quad \text{s.t.} \quad \|A^\top (Ax - b)\|_\infty \leq \eta. \tag{DS_η}$$

One can interpret (QP_λ) as a Lagrangean form of (LS_τ), and also show that for certain parameter values, (DS_η) and (QP_λ) are equivalent [10] (see also [256]).

There exist various other (algorithmic) approaches to sparse recovery, see [242, 108, 111] for recent surveys. Many methods involve *thresholding* operations of a greedy flavor in which a (fixed or variable) threshold value ϵ is used to decide which entries of a solution candidate x are treated as significant or irrelevant. For instance, a very simple, yet often effective, strategy is the *Iterative Hard Thresholding* algorithm (see [32]), which alternates a gradient step to reduce the error $\|Ax - b\|_2$ and the projection onto the set of k-sparse vectors (i.e., those with at most k nonze-

ros, for a fixed choice of k), given simply by keeping the k components with largest absolute values while setting the rest to zero. Other popular thresholding-related strategies are discussed in, e.g., [87, 79, 27, 256, 111].

1.2 Contributions and Outline of the Thesis

The central questions in Compressed Sensing can be briefly summarized as: How many linear measurements (and of which type) are sufficient to capture the information contained in a sparse signal? Given a set of such measurements, how (and up to which sparsity levels) can sparse exact or approximate solutions of the corresponding underdetermined linear equation systems indeed be successfully, and efficiently, recovered?

In this thesis, we will focus on the second question, i.e., our primary concern is *sparse recovery*. Our contributions come in three main parts:

- First, in Chapter 2, we discuss several important conditions for successful recovery of sparse (exact or approximate) solutions by efficient algorithms. In a very broad sense, such conditions can be classified into two groups: Some ensure that *specific* k-sparse vectors can be recovered, while most conditions yield *uniform* k-sparse recovery (i.e., *every* vector with at most k nonzero entries can be efficiently recovered). Moreover, many popular sparse recovery conditions are only *sufficient* in general, although some are indeed also necessary.

  The focus of our interest lies on the computational complexity of sparse recovery conditions. While some are easily evaluated, the more powerful ones had long been rumored to be intractable, but rigorous proofs were lacking. We confirm these conjectures by a series of NP-hardness results, i.e., we show that several ubiquitous conditions for uniqueness and recoverability of sparse solutions cannot be evaluated in polynomial time, unless P=NP. The main results of Chapter 2 were obtained jointly with Marc Pfetsch, see [237].

- Due to its well-founded popularity, an abundance of different algorithms to solve the Basis Pursuit problem (P_1) has been proposed over the last decade or so. In Chapter 3, we investigate the question which one of these methods is "the best". (As we will see, the answer is somewhat ambiguous.)

  To that end, we present a large test set of (P_1) instances with known (unique) optimal solutions, and numerical results of an extensive computational comparison of various state-of-the-art ℓ_1-solvers on this test set. We also contribute a new solver, ISAL1, which turns out to be competitive in some settings.

Moreover, we introduce a novel heuristic optimality check (HOC) that oftentimes allows for "jumping" to the optimal point long before a solver (practically) converges. Indeed, for several of the considered solvers, our numerical results show that HOC can significantly improve both accuracy and running time. We also establish theoretical guarantees for HOC success.

The above-described work presented in Chapter 3 emerged from a collaboration with Dirk Lorenz and Marc Pfetsch, and was previously reported in [174]. Additionally, we will establish that linear programming (LP) and Basis Pursuit problems are equivalent in the sense that one can solve any instance of either one problem via a suitable instance of the other. While the fact that (P_1) can be reformulated as an LP is well-known, the converse result appears to be new. Furthermore, we discuss extensions of the HOC procedure to the Basis Pursuit Denoising problem (P_1^δ) and its variant (QP_λ), and present some numerical results indicating its potential usefulness in the respective settings.

- The solver ISAL1 mentioned earlier is a specialization to ℓ_1-minimization problems of a new general-purpose framework for (nonsmooth) constrained convex optimization, which we develop in Chapter 4. The method is an extension of the classical projected subgradient algorithm: The usual exact projections of iterate points onto the feasible set are replaced by adaptive approximate projections. Since inexact projection can result in infeasible iterates, we call the method "infeasible-point subgradient algorithm" or simply "ISA".

  We present several variants of this ISA framework, provide corresponding convergence proofs, and give examples for the construction of adaptive approximate projection operators. In particular, we discuss ISAL1 in detail, i.e., how ISA can be adapted to (P_1) and (P_1^δ) (and some related denoising models), thus building the bridge back to the prevalent CS sparse recovery theme of the thesis.

  Many results of Chapter 4 were developed together with Dirk Lorenz and Marc Pfetsch and can be found in [175]. We complement this work with a "variable target-value" version of ISA, additional examples for projection operators, and more details about the specialization ISAL1 (in particular, the extension to (P_1^δ) was not treated previously).

Finally, Chapter 5 contains closing remarks on some open questions and possible extensions of the material covered in this thesis, and provides pointers to further lines of research and related problems.

Remark 1.2. Further work by the author that did not find its way into this thesis concerned the strong NP-hardness of projection onto the set of so-called *cosparse* vectors, $\{ x \ : \ \|\Omega x\|_0 \leq k \}$ with some matrix Ω, which arises in algorithmic ap-

proaches to certain alternative sparse representation models, see [236] (joint work with Rémi Gribonval and Marc Pfetsch). Moreover, the author was involved in applying CS-based ideas to the construction of 3D-visualizations of planetary nebulae, see [254] (a collaboration with Stephan Wenger, Marco Ament, Stefan Guthe, Dirk Lorenz, Daniel Weiskopf and Marcus Magnor).

1.3 Notation and Preliminaries

In the following, we shall fix some notation and terminology, and recall several useful basic results, that will be used throughout this thesis. Since our notation is mostly standard, the so-inclined reader may of course skip this section entirely, and consult it only should the need arise to look up unfamiliar expressions.

Fields and Sets

By $\mathbb{C}$, $\mathbb{R}$, $\mathbb{Q}$, $\mathbb{Z}$ and $\mathbb{N}$ we denote the fields of *complex, real, rational, integer and natural numbers*, respectively (the corresponding n-dimensional *vector spaces* are marked with an additional superscript n), with the convention that $0 \notin \mathbb{N}$; the binary field is denoted by $\mathbb{F}_2$. We will often abbreviate $[n] := \{1, 2, \dots, n\} \subset \mathbb{N}$ for some $n \in \mathbb{N}$. Closed, open, and half-open intervals of real numbers are specified as $[a, b]$, (a, b), and $[a, b)$ or $(a, b]$, respectively. The *empty set* is denoted by $\emptyset$. The *cardinality* of a (discrete) set S is $|S|$; note that for scalars α, $|\alpha|$ means its *absolute value*. Moreover, for a subset $S \subseteq T$ of some set T, its *complement* in T is denoted by $S^c = T \setminus S$ (the dependency on T will be contextually clear).

Vectors, Matrices and Linear Algebra

For a vector x or matrix A, their respective *transposes* will be denoted by $x^\top$ and $A^\top$. Any vector x we encounter will be a column vector; thus, $x^\top$ always describes a row vector. The all-ones vector will be denoted by $\mathbb{1}$ and the *identity matrix* by I; sometimes we add a subscript indicating the size (e.g., I_k for the $k \times k$ identity matrix). Moreover, the number 0 will occasionally also denote the all-zero vector or matrix of suitable size. A *diagonal matrix* with entries $\alpha_1, \dots, \alpha_n$ along the diagonal is denoted by $\mathrm{Diag}(\alpha_1, \dots, \alpha_n)$. The element in row i and column j of a matrix A is denoted by a_{ij} (or $a_{i,j}$); the i-th row is written as $a_i^\top$ and the j-th column as A_j. A *submatrix* of A induced by a subset C of its columns is denoted by A_C; similarly, x_C denotes the vector consisting of entries x_j with $j \in C$ of a

vector x. Occasionally, we consider submatrices obtained by restriction to a row subset R *and* a column subset C, which are denoted by A_{RC} (or $A_{R,C}$).

The *sign-vector* $\operatorname{sign}(x)$ contains entries ± 1 in accordance with the signs of the nonzero entries of x, and zeros wherever $x_j = 0$. The *support* of a vector is the set of indices of its nonzero entries; we will sometimes write $\operatorname{supp}(x)$.

For a matrix $A \in \mathbb{R}^{m\times n}$, we denote its *column space* (or *range*) by

$$\mathcal{R}(A) \coloneqq \{\, y \in \mathbb{R}^m \,:\, y = Ax \text{ for some } x \in \mathbb{R}^n \,\} \subseteq \mathbb{R}^m$$

(thus, the *row space* of A is $\mathcal{R}(A^\top)$) and its *nullspace* (or *kernel*) by

$$\mathcal{N}(A) \coloneqq \{\, x \in \mathbb{R}^n \,:\, Ax = 0 \,\} \subseteq \mathbb{R}^n.$$

Recall that $\mathcal{N}(A) = \mathcal{R}(A^\top)^\perp$, i.e., the nullspace of A is the *orthogonal complement* of its row space: $x^\top z = 0$ for all $x \in \mathcal{N}(A),\ z \in \mathcal{R}(A^\top)$ (see, e.g., [38, p. 646]).

The *rank* of A is the maximal size of a collection of linearly independent columns, or equivalently the maximal number of linearly independent rows (i.e., $\operatorname{rank}(A) = \operatorname{rank}(A^\top)$). Thus, $\operatorname{rank}(A) \le \min\{m, n\}$ with equality if and only if A has *full rank*. We will sometimes use the terms *column rank* and *row rank* to remind of which size parameter of A is rank-defining. If A is full-rank and square ($m = n$), it is invertible; we denote its inverse by A^{-1}. A useful generalization of the matrix inverse is the *(Moore-Penrose) pseudo-inverse* $A^\dagger$ (see, e.g., [38, Section A.5.4]); in particular, if $\operatorname{rank}(A) = m < n$ then $AA^\top$ is invertible and $A^\dagger = A^\top(AA^\top)^{-1}$ (so that $AA^\dagger = I$), if $\operatorname{rank}(A) = n < m$ then $A^\dagger = (A^\top A)^{-1}A^\top$ (whence $A^\dagger A = I$) analogously, and if A is invertible, $A^\dagger = A^{-1}$.

The *singular values* of A are denoted by $\sigma_i(A)$ ($i \in [\min\{m, n\}]$) and, if $m = n$, its *eigenvalues* by $\lambda_i(A)$ ($i \in [n]$). A (square) matrix is *positive semi-definite* if $x^\top Ax \ge 0$ for all vectors x (or equivalently, all eigenvalues of A are nonnegative), and *positive definite* if for all $x \ne 0$, $x^\top Ax > 0$ (all eigenvalues are positive).

Functions, Convexity, (Sub-)Differentiability, Optimization and Duality

A function $f : \mathbb{R}^n \to \mathbb{R}$ is *convex* if $f(\lambda x + (1-\lambda)y) \le \lambda f(x) + (1-\lambda)f(y)$ holds for all $x, y \in \mathbb{R}^n$ and any $\lambda \in [0, 1]$. A set $S \subseteq \mathbb{R}^n$ is convex, if $\lambda x + (1-\lambda)y \in S$ for all $x, y \in S$ and $\lambda \in [0, 1]$.

The *domain* $\operatorname{dom} f$ of a function f is the subset of $\mathbb{R}^n$ on which it is defined, i.e., $f : \operatorname{dom} f \to \mathbb{R}$. It is often convenient to have a function defined on all of $\mathbb{R}^n$, which can be achieved by *extended-value extension* (see, e.g., [38, Section 3.1.2]): Set $\hat{f}(x) \coloneqq f(x)$ for $x \in \operatorname{dom} f$, and $\hat{f}(x) \coloneqq \infty$ for all $x \notin \operatorname{dom} f$; then, $\hat{f} : \mathbb{R}^n \to \overline{\mathbb{R}} \coloneqq \mathbb{R} \cup \{\infty\}$ with *effective domain* $\operatorname{dom} \hat{f} \coloneqq \{\, x \,:\, \hat{f}(x) < \infty \,\} = \operatorname{dom} f$. In particular, $\hat{f}$

is convex if f is convex over its domain and $\operatorname{dom} f$ is itself a convex set.

Throughout this thesis, we will deal almost exclusively with *proper* functions (i.e., f with $\operatorname{dom} f \neq \emptyset$), and therefore usually do not state this explicitly. Moreover, for notational convenience, we will sometimes make an implicit transition of a (convex) function f to its extended-value extension $\hat{f}$, i.e., work with and refer to f as if it were $\hat{f}$. This should not cause any confusion. (Also, if necessary in this context, we implicitly adhere to standard extended arithmetic and ordering involving expressions containing ∞, e.g., $\infty + \infty = \infty$ and $\infty \leq \infty$.)

The *gradient* of a smooth (continuously differentiable) function f at a point x is denoted by $\nabla f(x)$. For a convex, but not necessarily smooth, function f, a *subgradient* of f at x is any vector $h \in \mathbb{R}^n$ satisfying the *subgradient inequality*

$$f(y) \geq f(x) + h^\top (y - x) \qquad \text{for all } y \in \mathbb{R}^n. \tag{1.1}$$

The collection of all subgradients of f at x is called the *subdifferential*, denoted by $\partial f(x)$; it is always nonempty, convex and compact (see, e.g., [24, Proposition B.24]). Moreover, if f is differentiable at x, then $\partial f(x) = \{\nabla f(x)\}$.

The (extended-valued) *indicator* (or *characteristic*) *function* ι_X of a set $X \subseteq \mathbb{R}^n$ is given by $\iota_X(x) := 0$ if $x \in X$, and $\iota_X(x) := \infty$ otherwise. Clearly, $\iota_X : \mathbb{R}^n \to \overline{\mathbb{R}}$ with $\operatorname{dom} \iota_X = X$; thus, ι_X is proper if and only if $X \neq \emptyset$. Moreover, ι_X is convex on $\mathbb{R}^n$. The indicator function is useful, for instance, to move constraints of an optimization problem into the objective function (which then is also extended-valued); see, e.g., Lemma 1.5 below.

The (possibly extended-valued) *conjugate function* f^* of a function $f : \operatorname{dom} f \to \mathbb{R}$ (or its extension $\hat{f} : \mathbb{R}^n \to \overline{\mathbb{R}}$) is given by

$$f^*(y) := \sup\{\, y^\top x - f(x) \;:\; x \in \operatorname{dom} f \,\} \quad (= \sup\{\, y^\top x - \hat{f}(x) \;:\; x \in \mathbb{R}^n \,\});$$

it is always convex and its (effective) domain is the set of those y for which the supremum is finite. Conjugate functions play an important role, e.g., in duality theory for convex optimization problems, as the following result shows (this is an instance of *Fenchel-Rockafellar duality*, see, e.g., [41, 124, 34, 219, 218]).

Lemma 1.3. *Let $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ and $g : \mathbb{R}^m \to \overline{\mathbb{R}}$ be two proper convex functions, and let $A \in \mathbb{R}^{m \times n}$ be a matrix. The dual problem of* $\min\{\, f(x) + g(Ax) \;:\; x \in \mathbb{R}^n \,\}$ *reads* $\max\{\, -f^*(-A^\top y) - g^*(y) \;:\; y \in \mathbb{R}^m \,\}$. *The dual (maximization) problem provides a lower bound for the primal (minimization) problem.* Strong duality *(i.e., equality of the objectives) holds under mild assumptions, e.g., if either* $\operatorname{dom} f = \mathbb{R}^n$ *or* $\operatorname{dom} g = \mathbb{R}^m$, *$Ax \in \operatorname{dom} g$ for some $x \in \mathbb{R}^n$ and both optima are finitely attained. Moreover, under strong duality, a primal-dual optimal pair (x^*, y^*) obeys*

the (saddle-point) relation $\min_x (Ax)^\top y^* + f(x) - g^*(y^*) = \min_x \max_y (Ax)^\top y + f(x) - g^*(y) = \max_y \min_x (Ax)^\top y + f(x) - g^*(y) = \max_y (Ax^*)^\top y + f(x^*) - g^*(y)$.

The stated conditions to ensure strong duality can be found, for instance, in [111, Theorem B.30]; other (and weaker) conditions are given in, e.g., [69, Remark 2.1] and other works on (Fenchel-Rockafellar) duality such as those listed above. The above saddle-point property stems from *Lagrangean duality* theory, of which the above duality principle is a special case, derived from the primal-equivalent constrained problem obtained by substituting $z \coloneqq Ax$ (see [111, Appendix B]); for details and many more results about Lagrange multiplier and duality theory, we refer to [221, 38, 24].

Similar very well-known results pertain to linear programs, see, e.g., [62, 224, 128]; for instance, the dual problem of $\max\{\, c^\top x \;:\; Ax \le b,\, x \ge 0 \,\}$ reads $\min\{\, b^\top y \;:\; A^\top y \ge c,\, y \ge 0 \,\}$, and the two problems share the same (finite) optimal value if and only if both are feasible and bounded (this also holds in the setting of Lemma 1.3). For an application of duality, see, e.g., Lemma 1.5 below.

The *normal cone* $N_X(x) = \{\, y \;:\; y^\top(z - x) \le 0 \;\; \forall\, z \in X \,\}$ of a closed convex set X at a point $x \in X$ can be used to state the following well-known *optimality condition*, see, e.g., [24, Proposition B.24(f)], [166, Proposition 2.5] or [221, Theorem 3.33].

Lemma 1.4. *Let $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ be a proper convex function, and let $X \subset \mathbb{R}^n$ be a closed convex set with $X \cap \operatorname{dom} f \neq \emptyset$. A point $x^* \in X$ is optimal for the problem $\min\{\, f(x) \;:\; x \in X \,\}$ if and only if $-\partial f(x^*) \cap N_X(x^*) \neq \emptyset$.*

Vector and Matrix Norms

We let $\|\cdot\|_p : \mathbb{R}^n \to \mathbb{R}$, $x \mapsto \|x\|_p \coloneqq \big(\sum_{i\in[n]} |x_i|^p\big)^{1/p}$; for $1 \le p < \infty$, this gives the usual (convex) *ℓ_p-norm*. In particular, we have the

$$\ell_1\text{-norm} \quad \|x\|_1 = \sum_{i=1}^n |x_i| = \operatorname{sign}(x)^\top x,$$

$$\text{the Euclidean/}\ell_2\text{-norm} \quad \|x\|_2 = \Big(\sum_{i=1}^n x_i^2\Big)^{1/2} = \sqrt{x^\top x}$$

and, extending the notation as usual, the *ℓ_∞-norm* $\|x\|_\infty \coloneqq \max\{\, |x_i| \;:\; i \in [n] \,\}$.

Although often also called "ℓ_p-norm", $\|\cdot\|_p$ with $0 < p < 1$ is only a quasinorm (and nonconvex), because—unlike true norms—it can violate the *triangle inequality* $\|x + y\| \le \|x\| + \|y\|$. Similarly, we let $\|x\|_0 \coloneqq |\operatorname{supp}(x)|$ denote the number of nonzero elements of a vector x, and adopt the conventional slight abuse of ter-

minology and call it ℓ_0-*norm*. Note that $\|\cdot\|_0$ is also not a proper norm, since $\|\alpha x\|_0 \neq |\alpha| \cdot \|x\|_0$ for scalars $\alpha \notin \{0, \pm 1\}$.

The *dual norm* to $\|\cdot\|_p$ $(1 \leq p \leq \infty)$ is $\|\cdot\|_p^* = \|\cdot\|_q$ with q such that $1/p + 1/q = 1$ (with $1/\infty = 0$ per convention); in particular, $\|\cdot\|_2^* = \|\cdot\|_2$ and $\|\cdot\|_1^* = \|\cdot\|_\infty$.

The ℓ_p-norms induce corresponding *matrix norms* $\|A\|_p = \max\{\, \|Ax\|_p / \|x\|_p \;:\; x \neq 0 \,\}$; in particular, the *column-sum norm* $\|A\|_1 = \max\{\, \|A_j\|_1 \;:\; j \in [n] \,\}$, the *row-sum norm* $\|A\|_\infty = \max\{\, \|a_i^\top\|_1 \;:\; i \in [m] \,\}$, and the

$$\textit{spectral norm} \quad \|A\|_2 = \sigma_{\max}(A) = \sqrt{\lambda_{\max}(A^\top A)},$$

where $\sigma_{\max}(A)$ ($\lambda_{\max}(A^\top A)$) is the largest singular value (eigenvalue) of A ($A^\top A$).

Some useful norm inequalities are (let $x, y \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$): $|x^\top y| \leq \|x\|_2 \, \|y\|_2$ (*Cauchy-Schwarz inequality*), $\|x\|_2 \leq \|x\|_1 \leq \sqrt{n}\|x\|_2$, $\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n}\|x\|_\infty$, $\|x\|_\infty \leq \|x\|_1 \leq n \, \|x\|_\infty$, $\|Ax\|_2 \leq \|A\|_2 \|x\|_2$, $\|A^{-1}\|_2 = 1/\|A\|_2$, and $\|A\|_2 \leq \sqrt{\|A\|_1 \|A\|_\infty}$.

Duals of ℓ_1-Minimization Problems

The Basis Pursuit problem

$$\min \|x\|_1 \quad \text{s.t.} \quad Ax = b \tag{P_1}$$

and (to a lesser extent) its denoising counterpart

$$\min \|x\|_1 \quad \text{s.t.} \quad \|Ax - b\|_2 \leq \delta \tag{P_1^δ}$$

play a prominent role in this thesis. The corresponding dual problems are well-known; for the sake of completeness, we derive them explicitly in the following.

Lemma 1.5. *Let $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. The dual problem of (P_1) reads*

$$\max -b^\top y \quad \text{s.t.} \quad \|A^\top y\|_\infty \leq 1, \tag{D_1}$$

and that of (P_1^δ) is

$$\max -b^\top y - \delta \|y\|_2 \quad \text{s.t.} \quad \|A^\top y\|_\infty \leq 1. \tag{D_1^δ}$$

Proof. Considering (P_1^δ), note that it can be equivalently written as

$$\min \|x\|_1 + \iota_{\{\, v \,:\, \|v - b\|_2 \leq \delta \,\}}(Ax).$$

It is well-known that the conjugate function of an ℓ_p-norm is the indicator function

of the unit ball w.r.t. the dual norm (see, e.g., [38, Example 3.26]). Thus, for $f(x) = \|x\|_1$, we have

$$f^*(y) = \sup\{\, y^\top x - \|x\|_1 \,:\, x \in \mathbb{R}^n \,\} = \iota_{\{\, z \,:\, \|z\|_1^* \le 1 \,\}}(y) = \begin{cases} 0, & \|z\|_\infty \le 1, \\ \infty, & \text{otherwise.} \end{cases}$$

Similarly, for $g(x) = \iota_\delta(x) \coloneqq \iota_{\{\, v \,:\, \|v-b\|_2 \le \delta \,\}}(x)$, we obtain

$$\begin{aligned} g^*(y) &= \max_x\{\, y^\top x \,:\, \|x-b\|_2 \le \delta \,\} = \max_u\{\, y^\top (u+b) \,:\, \|u\|_2 \le \delta \,\} \\ &= b^\top y + \max_u\{\, y^\top u \,:\, \|u\|_2 \le \delta \,\} = b^\top y + \max_w\{\, y^\top (\delta\, w) \,:\, \tfrac{1}{\delta}\|\delta\, w\|_2 \le 1 \,\} \\ &= b^\top y + \delta \max_w\{\, y^\top w \,:\, \|w\|_2 \le 1 \,\} = b^\top y + \delta\, \|y\|_2, \end{aligned}$$

where the last equality follows because the support function of a unit norm ball is the dual norm, see, e.g., [221, Example 2.96]; recall that $\|\cdot\|_2^* = \|\cdot\|_2$.

Thus, by Lemma 1.3, the dual of (P_1^δ) is given by

$$\max\ -f^*(-A^\top y) - g^*(y) \quad \Leftrightarrow \quad \max\ -b^\top y - \delta\|y\|_2 \quad \text{s.t.} \quad \|-A^\top y\|_\infty \le 1,$$

which is obviously equivalent to (D_1^δ).

Finally, it remains to note that for $\delta = 0$, (P_1) and (P_1^δ) of course coincide; in particular, we then have $g^*(y) = \max\{\, y^\top x \,:\, x = b \,\} = b^\top y$, and completely analogously to the above derivation, Lemma 1.3 gives (D_1) as the dual of (P_1). □

Remark 1.6. Due to the symmetry of the constraints in (D_1) and (D_1^δ), we could also state them with objective functions $b^\top y$ and $b^\top y - \delta\|y\|_2$, respectively. For consistency, we will always use the forms given in the above lemma.

The Conjugate Gradient Method

The *method of conjugate gradients* [133], CG for short, is a very popular iterative algorithm for computing the solution of the linear equation system $Sv = s$ with $S \in \mathbb{R}^{n\times n}$ symmetric (i.e., $S = S^\top$) positive definite and $s \in \mathbb{R}^n$. In fact, CG can be used to minimize any continuous differentiable function; in the particular case of a quadratic form $\frac{1}{2}v^\top Sv - s^\top x + c$, the minimizer actually solves $Sv = s$ if S is symmetric positive definite, see, e.g., [227]. The CG iterations can be written as:

Choose $v^0 \in \mathbb{R}^n$, $K_{\max} \in \mathbb{N}$ and initialize $d_0 \coloneqq r_0 \coloneqq s - Sv^0$

For $k = 0, 1, 2, \ldots, K_{\max}$

Set $v^{k+1} \coloneqq v^k + \frac{\|r_k\|_2^2}{d_k^\top S d_k} d_k$, $r_{k+1} \coloneqq r_k - \frac{\|r_k\|_2^2}{d_k^\top S d_k} S d_k$, $d_{k+1} \coloneqq r_{k+1} + \frac{\|r_{k+1}\|_2^2}{\|r_k\|_2^2} d_k$.

The name "conjugate gradient method" derives from the fact that the search directions d_k are constructed by conjugation of the residuals $s - Sv^k$ (which incidentally are the respective gradients of the aforementioned quadratic form at v^k), so it holds that $d_j^\top S d_k = 0$ for all j, k (also called S-orthogonality). Detailed descriptions of the CG method and useful properties of the iterative process can be found, e.g., in [227, 222, 133, 203].

For a matrix $A \in \mathbb{R}^{m\times n}$ with $\operatorname{rank}(A) = m < n$, a solution of the underdetermined system $Ax = b$ can be obtained by applying CG to the associated *normal equation* $AA^\top v = A^\top b$ (note that $AA^\top$ is symm. pos. def.); similarly, if $\operatorname{rank}(A) = n < m$, the normal equation $A^\top A v = A^\top b$ can be solved by CG. In both cases, solving the respective normal equation yields the vector $\hat{x} = A^\dagger b$, which gives the minimum-ℓ_2-norm solution in the first case ($Ax = b$ is underdetermined) and the *least-squares solution* in the second (in which $Ax = b$ is overdetermined), see, e.g., [111, Corollaries A.21 and A.22]. It is also important to note (especially regarding normal equations) that CG can be implemented using only products with A or $A^\top$, i.e., the method does not require explicit knowledge of $AA^\top$ or $A^\top A$, respectively.

Projection

For a closed convex set $X \subseteq \mathbb{R}^n$, the (Euclidean) *projection* of a point $x \in \mathbb{R}^n$ onto the set X is the unique point $\mathcal{P}_X(x) \in X$ that is closest to x (w.r.t. the ℓ_2-norm), i.e.,

$$\mathcal{P}_X(x) \coloneqq \arg\min\{\, \|x - z\|_2 \,:\, z \in X \,\}.$$

If $x \in X$, then trivially $\mathcal{P}_X(x) = x$. The *distance* of x to X is given by $d_X(x) \coloneqq \|x - \mathcal{P}_X(x)\|_2$. The projection is easily seen to be *nonexpansive*, i.e.,

$$\|\mathcal{P}_X(x) - \mathcal{P}_X(y)\|_2 \le \|x - y\|_2 \qquad \text{for all } x,\, y \in \mathbb{R}^n.$$

The above definition of projection can be applied to other classes of sets; however, uniqueness (or even existence) is then not always guaranteed.

Asymptotics, Encoding Lengths and Complexity Theory

Asymptotic upper and lower bounds for functions (often of positive integers) are specified using the Landau symbols $\mathcal{O}$ and Ω, respectively. Thus, $f(n) \in \mathcal{O}(g(n))$ means that f does not grow significantly faster than g (as $n \to \infty$), while $f(n) \in \Omega(g(n))$ means that the growth of f is of the same order as or faster than that of g (as $n \to \infty$). More precisely, we write $f(n) \in \mathcal{O}(g(n))$ if there exist constants $0 < \alpha, \beta < \infty$ such that $|f(n)| \le \alpha\,|g(n)| + \beta$ for all n, and $f(n) \in \Omega(g(n))$ if

and only if $g(n) \in \mathcal{O}(f(n))$; see, e.g., [161, Definition 1.2]. In particular, if f is a polynomial of degree k, then $f(n) \in \mathcal{O}(n^k)$.

The *encoding length* of an integer n (i.e., the length of its binary/bit representation) is defined as $\langle n \rangle := 1 + \lceil \log_2(|n| + 1) \rceil$, that of a rational number r is $\langle r \rangle := \langle s \rangle + \langle t \rangle$, where s and t are mutually prime integers (w.l.o.g., $t > 0$) such that $r = s/t$. For vectors and matrices, the encoding length is defined analogously as the sum of the encoding lengths of all entries and will be denoted in the same way. Clearly, for any $r \in \mathbb{Q}$, $\langle r \rangle = \langle -r \rangle$. Moreover, note that $\langle 0 \rangle = 1$, since the bit that otherwise encodes the sign can be omitted.

The arguably most important computational complexity classes are P and NP. Although the technical definitions are based on so-called formal languages, the following widespread viewpoint is sufficient for our purposes (see [115] for a thorough treatment): Somewhat loosely speaking, the complexity class P contains "easy" (yes/no-decision) problems, i.e., those that can be solved "efficiently"—namely, in time polynomially bounded by the encoding length of the input. The class NP contains decision problems for which a positive answer can be *verified* in input-polynomial time; the abbreviation means *nondeterministic polynomial-time*, indicating that one could "guess" a correct solution and then efficiently verify that it yields a "yes" answer. Clearly, P$\subseteq$NP. However, it is widely believed that P$\neq$NP, i.e., not for every problem in NP can the answer be deterministically *computed* (not only verified) in input-polynomial time. This assumption implies that there are indeed "hard" problems that cannot be solved efficiently in general. A problem is called *NP-hard* if every problem in NP can be polynomially *reduced* (i.e., transformed using input-polynomial time and space) to it; if a problem is both NP-hard and contained in NP, it is called *NP-complete*. Recall that NP-hardness of a problem implies that, unless P=NP, there is no polynomial-time algorithm for solving this problem in general (for NP-complete problems, the word "unless" can be strengthened to "if and only if")—if there was, *every* problem in NP would be polynomial-time solvable.

An optimization problem is called NP-hard if the associated decision problem is NP-hard. Moreover, *strong NP-hardness*, or *NP-hardness in the strong sense*, implies that, unless P=NP, there cannot exist a fully polynomial-time approximation scheme (FPTAS), i.e., an algorithm that solves a (say, minimization) problem within a factor of $(1 + \varepsilon)$ of the optimal value in polynomial time with respect to the input size and $1/\varepsilon$; an FPTAS often exists for weakly (not strongly) NP-hard problems. Strong NP-hardness can also be understood as an indication that a problem's intractability does not depend on ill-conditioning of the input data (due to the occurrence of very large numbers). In fact, for strong NP-hard problems, there also cannot exist a pseudo-polynomial time solution algorithm (unless P=NP), i.e., one with runtime polynomial in the input encoding length and the magnitude of

the largest numerical value in a given instance. Resolving the "P vs. NP" question (e.g., by actually proving that P≠NP) is often considered the most important open problem in computational complexity theory today.

We will also encounter coNP, the complementary class to NP (i.e., decision problems with polynomially verifiable "no" answers). The notions of (strong) hardness and completeness apply to coNP in full analogy to their use with respect to NP, cf. [115] (see also Remark 2.15 on p. 35 below).

Background Reading Material

Naturally, the basics we gathered above only reflect what is most relevant to the contents of this thesis; while intended to offer at least a small measure of self-containment, we make no claim of completeness. For comprehensive treatments of all the matters touched upon, many more fundamental results and detailed expositions, we refer to standard works on the respective subjects, e.g.,

- [38, 24, 221, 203] for convex and nonlinear optimization theory,
- [224, 62] for linear programming,
- [118, 121, 138] for linear algebra and matrix theory, and
- [115, 128, 161] for computational complexity,

to name a few. Moreover, we will utilize some graph theoretical concepts, cf. [161, 35], and ideas from matroid theory, see, e.g., [205].

CHAPTER 2

Recovery Conditions and Their Computational Complexity

The immense interest in the field of Compressed Sensing was triggered in large parts by the discovery that the NP-hard sparse representation problems (P_0) and (P_0^δ) can, under certain conditions, be solved by efficient algorithms. Several such *sparse recovery conditions (SRCs)* have been introduced over the past years—the associated terms spark, mutual coherence, exact recovery condition, restricted isometry property and nullspace property are ubiquitous in the CS literature (and will be formally defined as we progress through this chapter).

Essentially, an SRC captures certain properties of the sensing matrix A that make it suitable for CS sparse recovery tasks. Thus, SRCs are useful tools to assert uniqueness of sparse representations, to assess the possibilities for their efficient recovery (a priori) and to (post-)validate heuristic solutions. Note also that the best (regarding the number of measurements ensuring recoverability) currently known CS matrix constructions involve randomness, and desirable properties can be shown to hold with high probability—but, to gain final certainty, one would still have to check the corresponding recovery condition explicitly. Therefore, a question of paramount importance, to which we devote this chapter, is:

For a (fixed) matrix A, can a given SRC be efficiently evaluated?

We will begin our discussion by exhibiting two important cases where this is indeed possible: Incoherence and the exact recovery condition, see Sections 2.1 and 2.2, respectively. In the main parts of this chapter (Sections 2.3-2.5), we establish NP-hardness of evaluating several other well-known (and more powerful) SRCs that are based on the spark, restricted isometry property and nullspace property. The

computational intractability of such SRCs had been suspected in the CS community for a long time, but no rigorous proofs were given prior to our recent work [237] (coauthored by Marc Pfetsch). The majority of Sections 2.3–2.5 consists of the contents of this paper (up to slight modifications and revisions), with some additional remarks and an updated overview of known SRCs.

2.1 Incoherence of the Sensing Matrix

For a matrix $A \in \mathbb{R}^{m \times n}$, its *mutual coherence* $\mu(A)$ is defined as

$$\mu(A) \coloneqq \max_{\substack{i,j \in [n], \\ i \neq j}} \frac{|A_i^\top A_j|}{\|A_i\|_2 \, \|A_j\|_2}. \tag{2.1}$$

It can be seen as a measure of the conditioning of submatrices formed by collections of columns (see, e.g., [111]). A matrix with small $\mu(A)$ is called *incoherent*; this property is advantageous for the sparse recovery setting, as the following famous result shows.

Theorem 2.1 ([90, 126])**.** *Suppose that the underdetermined linear system $Ax = b$ has a solution x^* with*

$$\|x^*\|_0 \; < \; \frac{1}{2}\left(1 + \frac{1}{\mu(A)}\right). \tag{2.2}$$

Then x^* is the unique optimal solution for both (P_0) and (P_1).

Indeed, this was one of the first theoretical results that reveals the possibility of solving the NP-hard problem (P_0) by computing a solution to a much simpler problem instead: Every sufficiently sparse vector $\hat{x}$ can be recovered by solving the Basis Pursuit instance given by A and $b \coloneqq A\hat{x}$. In turn, if we obtain some solution x^* for $Ax = b$ that obeys the sparsity bound (2.2), then we know we have found the unique optimal solution to (P_0). Thus, Theorem 2.1 provides a sufficient (but not necessary) condition for ℓ_0-ℓ_1-equivalence. In fact, (2.2) is also sufficient for k-sparse recovery by several Matching Pursuit variants, see, e.g., [239].

Clearly, computing $\mu(A)$ takes $\mathcal{O}(n^2 m)$ floating point operations, which is widely considered tractable even for large-scale instances. Thus, the SRC (2.2) can be efficiently evaluated.

However, the sparsity bounds it provides are quite conservative, or pessimistic, and often cannot be expected to be achievable in practice [100]. Nonetheless, incoherence is a useful concept, e.g., for the design of sensing matrices. For instance,

if and only if A is an orthonormal basis for $\mathbb{R}^m$, then $\mu(A) = 0$ and every vector can be recovered (this is of course trivial, since here $Ax = b$ always has the single solution $A^{-1}b$). The coherence between any pair of orthonormal bases is between $1/\sqrt{n}$ and 1; the lower bound is attained for various choices such as the (complex) identity-Fourier pair or the concatenation of the identity matrix with a Hadamard matrix; see, e.g., [92]. (In such two-bases situations, (2.2) can be improved to $\|x^*\|_0 < (\sqrt{2} - 1/2)/\mu(A)$ [101], which actually characterizes ℓ_0-ℓ_1-equivalence in this case [106].) For general $m \times n$ matrices A with $\operatorname{rank}(A) = m \leq n$, it is known that the mutual coherence satisfies the *Welch bound* $\mu(A) \geq \sqrt{(n-m)/(m(n-1))}$, cf. [252]. Matrices that achieve this lower bound actually exist (although not for every pair m, n); they are called *(optimal) Grassmannian frames* or *equiangular tight frames*; see, e.g., [232, 234]. Consequently, this choice of sensing matrix yields the best possible properties w.r.t. the incoherence-based SRC (2.2): It allows uniform recovery of k-sparse solutions whenever $k < (1/2)(1 + \sqrt{m(n-1)/(n-m)})$. Favorable incoherence properties are also established to hold with high probability in the asymptotic regime (i.e., with m and n very large) for certain matrices involving random entries; for instance, for a random orthogonal $m \times n$ matrix A, $\mu(A)$ typically behaves like $\sqrt{\log(mn)/m}$ [92, 100], and if the distribution of the random entries has zero mean and finite variance, $\mu(A) = \sqrt{2\log(n)/m}$ [165, p. 26]. (Note that these mutual coherence values are relatively small if the undersampling ratio m/n is not too small.)

2.2 The Exact Recovery Condition (ERC)

The *exact recovery condition (ERC)* from [239] is fulfilled if, for a solution x^* of the (underdetermined) system $Ax = b$ with support $S \coloneqq \operatorname{supp}(x^*)$, it holds that

$$\operatorname{erc}(A, S) \coloneqq \max_{j \notin S} \|(A_S^\top A_S)^{-1} A_S^\top A_j\|_1 = \max_{j \notin S} \|A_j^\top (A_S^\top)^\dagger\|_1 < 1. \tag{2.3}$$

In this case, x^* is the unique solution of (P_1) (and, if sufficiently sparse, of (P_0)) and it can also be recovered using Matching Pursuit techniques, see [239]. Note that the ERC provides a recovery guarantee for *individual* solutions (in the sense that they are unique ℓ_1-minimizers), but not uniformly for all (k-)sparse solutions. Thus, the ERC shows that recovery can still be guaranteed for certain supports with cardinalities going beyond bounds like (2.2). The sparsity of x^* appears only implicitly in the ERC, as the size of the support set S. The cost for evaluating $\operatorname{erc}(A, S)$ depends mainly on computing the pseudo-inverse $(A_S^\top)^\dagger$ and matrix-vector

multiplications; thus, it can roughly be estimated as $\mathcal{O}(m^3 + nm^2) \subseteq \mathcal{O}(nm^2)$.

Of course, since $\operatorname{supp}(x^*)$ will hardly be known a priori, evaluating (2.3) directly is essentially restricted to validation of candidate solutions at hand. However, one easily obtains a sufficient condition for uniform k-sparse recovery by requiring (2.3) to hold for *all* index sets $S \subset [n]$ with $|S| \leq k$ (in this case, the unique ℓ_1-minimizer also uniquely solves (P_0), cf. [239]). Verifying this condition directly is potentially very expensive. Nevertheless, it can be shown to hold if the condition (2.2) given by the mutual coherence is satisfied (see [100, Theorem 4.7]) or, more generally, if $\mu_{k-1}(A) + \mu_k(A) < 1$, where

$$\mu_k(A) \;:=\; \max_{\substack{S\subset[n],\\ |S|=k}} \max_{j\in[n]\setminus S} \sum_{i\in S} \frac{|A_i^\top A_j|}{\|A_i\|_2\,\|A_j\|_2} \;\leq\; k\,\mu(A)$$

is the *cumulative mutual coherence*, see [239].

Moreover, it is noteworthy that the (uniform) ERC, while only providing a sufficient condition for ℓ_0-ℓ_1-equivalence, is actually "worst-case necessary" for recovery by Orthogonal Matching Pursuit (OMP), i.e., OMP can fail if the optimal support violates (2.3) [239, Theorem 3.10]. Note also that the original ERC definition in [239] pertains to A with unit ℓ_2-norm columns; however, this assumption is in fact not needed for the results about BP (see also Remark 3.3 on p. 60), and it also appears to be nonessential to many of those concerning OMP given in [239]. The normalization assumption is common throughout the CS literature and clearly not very restrictive (at least computationally); a possible justification is that, generally, different column norms may adversarially affect the capability of (P_1) to mimic (P_0)—a column with very small norm is less likely to "receive" a nonzero coefficient in an optimal ℓ_1-minimizer (because this coefficient would need to be relatively large), while this situation is of no concern for the ℓ_0-objective. Nevertheless, slightly generalized versions of the ERC were recently introduced in [233] which explicitly account for the case when the columns of A are not normalized to unit Euclidean length and more directly accommodate whether recovery is attempted via BP or OMP.

2.3 The Spark of a Matrix

The *spark* of a matrix is defined as the smallest number of linearly dependent columns. This term was first defined in [90], where strong results concerning uniqueness of sparse representations were proven. The fundamental such result is the following.

Theorem 2.2 ([90, 126]; see also [165, Theorem 1.1])**.** *Every k-sparse vector x^* is the respective unique sparsest solution of $Ax = b$ with $b := Ax^*$ if and only if $k < \operatorname{spark}(A)/2$.*

Note that, for instance, the uniqueness w.r.t. (P_0) in the incoherence-based SRC of Theorem 2.1 is a direct consequence of this result, because the mutual coherence provides a lower bound on the spark: $\operatorname{spark}(A) \geq 1+1/\mu(A)$ (see [100, Lemma 2.1]).

Remark 2.3. It is important to note that the above theorem provides a characterization of *uniform* recoverability of k-sparse solutions. In general, $\|x^*\|_0 < \operatorname{spark}(A)/2$ is only a sufficient condition for x^* to be the unique sparsest solution of $Ax = b := Ax^*$ (see, e.g., [100, Theorem 2.4]), but not necessary: Consider, for example,

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 0 \end{pmatrix};$$

then, $x^* = (1,0,0)^\top$ is the unique (and sparsest) solution, but $\|x^*\|_0 = 1 = \operatorname{spark}(A)/2$.

The value $\operatorname{spark}(A)$ is also known as the *girth* of the vector matroid associated with the columns of A, cf. [205]. A matroid can be described in terms of its circuits; in our context, these are the inclusion-wise minimal collections of linearly dependent columns of A. More precisely, a *circuit* is a set $C \subseteq [n]$ of column indices such that $A_C\, x = 0$ has a nonzero solution, but every proper subset of C does not have this property, i.e., $\operatorname{rank}(A_C) = |C| - 1 = \operatorname{rank}(A_{C\setminus\{j\}})$ for every $j \in C$. (For notational simplicity, we will sometimes identify circuits C with the associated solutions $x \in \mathbb{R}^n$ of $Ax = 0$ having support C.) The smallest size (cardinality) of a circuit can be expressed as

$$\operatorname{spark}(A) \;:=\; \min\{\, \|x\|_0 \,:\, Ax = 0,\; x \neq 0 \,\}. \tag{2.4}$$

Example 2.4. Consider the matrix

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$$

Clearly, the first two columns yield a minimum-size circuit (in fact, the only one), i.e., $\operatorname{spark}(A) = 2$. In particular, note that generally, $\operatorname{spark}(A) \leq k$ does *not* guarantee that there also exists a *vector with k nonzeros* in the nullspace of A; e.g., take $k = 3$ for the above A. On the other hand, it is immediately clear that a nullspace vector with support size k does not yield $\operatorname{spark}(A) = k$, but only $\operatorname{spark}(A) \leq k$. This

distinction between circuits and nullspace vectors in general will be crucial in the proofs below.

2.3.1 Complexity of Spark Computation

Ever since its introduction, the value spark(A) has been claimed to be NP-hard to calculate, but, to the best of our knowledge, without a proof or reference for this fact. It seems to have escaped researchers' notice that [148] contains a proof that deciding whether the spark equals the number of rows is NP-hard, by reduction from the Subset Sum Problem (cf. [MP9] in [115]); for details, see Remark 2.11 below. Moreover, [59] provides a different proof for this special case, by a reduction from the (homogeneous) Maximum Feasible Subsystem problem [5]; yet another proof is contained in [103]. Even earlier, the authors of [67] claimed to have a proof[2], but gave credit to the dissertation [185] for establishing NP-hardness of spark computations. However, a closer inspection reveals that the result in [185] is in fact *not* about the spark but the girth of *transversal matroids* (of bipartite graphs). Only recently, a variant of the latter proof has resurfaced in [3], where it is used to derive (nondeterministic) complexity results for constructing so-called *full spark frames*, i.e., matrices exhibiting the highest-possible spark. Every transversal matroid can be represented by a matrix over any infinite field or finite field with sufficiently large cardinality [208], but there is no known deterministic way to construct such a matrix.

We will adapt the proof idea from [185], a reduction from the k-Clique Problem (cf. [GT19] in [115], or [147]), to *vector matroids* and thus establish that spark computation is NP-hard (without the restriction that the spark equals the row size).

The main result of this section is the following.

Theorem 2.5. *Given a matrix $A \in \mathbb{Q}^{m\times n}$ and a positive integer k, the problem to decide whether (the vector matroid associated with the columns of) A has a circuit of size at most k is NP-complete.*

For our proof, we need several auxiliary results (cf. [35, 161] for graph theory basics):

Lemma 2.6. *The vertex-edge incidence matrix of an undirected simple graph with N vertices, B bipartite components, and Q isolated vertices has rank $N - B - Q$.*

This result seems to be rediscovered every once in a while. The earliest proof we

[2] They probably refer to [66, Theorem 6.1], but its proof is incomplete; see Remark 2.10.

are aware of is due to van Nuffelen [248] and works through various case distinctions considering linear dependencies of the matrix rows and consequences of the existence of isolated or bipartite components.

Lemma 2.7. *Let $G = (V, E)$ be a simple undirected graph with vertex set V and edge set E. Let A be its vertex-edge incidence matrix, and let $k > 4$ be some integer. Suppose that G only has connected components with at least four vertices each, $|E| = \binom{k}{2}$, and $\operatorname{rank}(A) = k$. Then the graph G has exactly $|V| = k$ vertices.*

Proof. Let the preconditions of the lemma hold. Then, since G has no isolated vertices, Lemma 2.6 tells us that the number of vertices is

$$N \;=\; \operatorname{rank}(A) + B \;=\; k + B,$$

where B is the number of bipartite components in G. Assume that $B > 0$, since otherwise the lemma is trivially true.

We claim that the number of edges in G can be at most

$$|E| \;\leq\; \binom{N}{2} - \frac{4(N-4)}{2} B - 2\,B. \tag{2.5}$$

To see this, recall that G can have at most $\binom{N}{2}$ edges. Each connected component has at least four vertices. Since there are no edges between such a component and vertices outside, the total number of possible edges is reduced by at least $4(N-4)/2$ per component (the factor $1/2$ ensures that we do not count any edges twice). Since G has at least B connected components, the possible number of edges is hence decreased at least by the second term in the right hand side of (2.5). Moreover, since each bipartite component has at least four vertices, at least two of the potential edges cannot be present inside each such component, which yields the last term in (2.5). Note that the bound (2.5) is sharp if G consists only of bipartite components with four vertices each.

Expanding (2.5) using $N = k + B$, we obtain

$$\begin{aligned} |E| \;&\leq \binom{N}{2} - \frac{4(N-4)}{2} B - 2\,B \;=\; \frac{(k+B)(k+B-1)}{2} - \frac{4(k+B-4)}{2} B - 2\,B \\ &= \frac{1}{2}k^2 - \frac{1}{2}k - k\,B - \frac{3}{2}B^2 + \frac{11}{2}\,B \;=\; \binom{k}{2} - \left(k\,B + \frac{3}{2}B^2 - \frac{11}{2}\,B\right), \end{aligned}$$

and observe that

$$k\,B + \frac{3}{2}B^2 - \frac{11}{2}\,B \;\geq\; 5\,B + \frac{3}{2}B^2 - \frac{11}{2}\,B \;=\; \frac{3}{2}B^2 - \frac{1}{2}\,B \;>\; 0$$

if $B > 0$ (recall that B is integer). Thus, there are strictly less than $\binom{k}{2}$ edges, contradicting the premise $|E| = \binom{k}{2}$. Hence, $B = 0$. □

Lemma 2.8. *Let $H = (h_{ij}) \in \mathbb{Z}^{m\times n}$ be a full-rank integer matrix with $m \leq n$ and let $\eta := \max\{\, |h_{ij}| \,:\, i \in [m],\, j \in [n] \,\}$. Let $q \in \{m, \dots, n\}$ and define the $q \times n$ integer matrix*

$$H(x) := \begin{pmatrix} & & H & & \\ 1 & x & (x)^2 & \dots & (x)^{n-1} \\ 1 & x+1 & (x+1)^2 & \dots & (x+1)^{n-1} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & x+q-m-1 & (x+q-m-1)^2 & \dots & (x+q-m-1)^{n-1} \end{pmatrix}.$$

Suppose $|x| \geq \eta^m q^{qn} + 1$. Then, if $\operatorname{rank}(H_S) = m$ for any column subset S with $|S| = q$, the submatrix $H(x)_S \in \mathbb{Z}^{q\times q}$ has full rank q.

Proof. This result is a combination of Lemma 1 and (ideas from the proof of) Proposition 4 from [59]. For clarity, we give the details here. Consider some $S \subseteq [n]$ with $|S| = q$; w.l.o.g., assume $q \geq m+1$ (otherwise, the statement is trivial). Assume that $\operatorname{rank}(H_S) = m$ and note that the last $q-m$ rows of $H(x)_S$ form a submatrix of a generalized Vandermonde matrix with distinct nodes (see, e.g., [84]), which is easily seen to have full rank as well; see also [59]. Consider the polynomial $p(x) := \det(H(x)_S)$. From [59, Lemma 1], we know that there exists some x for which the row space of H_S and the subspace spanned by the last $q-m$ rows of $H(x)_S$ are transversal, i.e., they only intersect trivially. In particular, this shows that p cannot be identical to the zero polynomial (both transversal parts have full rank). Let d be the degree of $p(x)$ (which depends on the choice of S); thus, $p(x) = \beta_0 + \beta_1 x + \dots + \beta_d x^d$ with $\beta_d \neq 0$. It is easy to see that $\beta_i \in \mathbb{Z}$ for all $i \in \{0, 1, \dots, d\}$.

Moreover, we now show that $|\beta_i| \leq \eta^m q^{qn}$ for all i: Write $S = \{s_1, \dots, s_q\}$, let Σ_q denote the set of all permutations of $1, \dots, q$ and let $\operatorname{sgn}(\sigma) \in \{\pm 1\}$ be the signum of a permutation $\sigma \in \Sigma_q$ (i.e., the determinant of the permutation matrix associated with σ). Expanding the determinant $\det H(x)_S$ using Leibniz's formula, we obtain

$$\begin{aligned} p(x) = \det(H(x)_S) &= \sum_{\sigma\in\Sigma_q} \left(\operatorname{sgn}(\sigma) \prod_{i=1}^{q} h(x)_{i,s_{\sigma(i)}} \right) \\ &= \sum_{\sigma\in\Sigma_q} \left(\operatorname{sgn}(\sigma) \prod_{i=1}^{m} h_{i,s_{\sigma(i)}} \prod_{j=0}^{q-m-1} (x+j)^{s_{\sigma(m+1+j)}-1} \right). \end{aligned}$$

(Recall that every summand of the first sum contains exactly one entry from each

row and column of $H(x)_S$, i.e., in particular, m entries from H_S and $q-m$ entries from the Vandermonde-like last $q-m$ rows of $H(x)_S$.)

Note that by definition of η and since $\operatorname{sgn}(\sigma) \in \{\pm 1\}$, $\operatorname{sgn}(\sigma) \prod_{i=1}^{m} h_{i,s_{\sigma(i)}} \leq \eta^m$ for all $\sigma \in \Sigma_q$. Therefore, the coefficients β_i of $p(x)$ cannot have larger absolute values than those of the polynomial $\eta^m \sum_{\sigma\in\Sigma_q} \big(\prod_{j=0}^{q-m-1}(x+j)^{s_{\sigma(m+1+j)}-1}\big)$. There are $q! \leq q^q$ permutations of the elements of S, i.e., summands in the latter expression. In particular, the coefficient contribution of each such summand is nonnegative (since $j \geq 0$ in each factor). Here, the largest possible numbers occur if S contains the indices of the last $q-m$ columns of $H(x)$, i.e., $\{n-q+m+1,\ldots,n\} \subset S$. Hence, all values $|\beta_i|$ are no larger than the largest possible coefficient in the polynomial $\eta^m q^q \prod_{j=0}^{q-m-1}(x+j)^{n-q+m+j}$.

By the binomial formula, we have (for every $j \in \{0,1,\ldots,q-m-1\}$)

$$(x+j)^{n-q+m+j} \;=\; \sum_{\ell=0}^{n-q+m+j} \binom{n-q+m+j}{\ell} x^{n-q+m+j-\ell}\, j^{\ell}.$$

Clearly, the largest coefficients here are attained for $j = q-m-1$; thus, every $|\beta_i|$ is upper bounded by the largest coefficient of the polynomial

$$\begin{aligned}
&\eta^m q^q \prod_{j=0}^{q-m-1} (x+q-m-1)^{n-q+m+q-m-1} \;=\; \eta^m q^q \prod_{j=1}^{q-m} (x+q-m-1)^{n-1} \\
&= \eta^m q^q \Big((x+q-m-1)^{n-1}\Big)^{q-m} \;=\; \eta^m q^q (x+q-m-1)^{(n-1)(q-m)} \\
&= \eta^m q^q \sum_{\ell=0}^{(n-1)(q-m)} \binom{(n-1)(q-m)}{\ell} x^{(n-1)(q-m)-\ell} (q-m-1)^{\ell}.
\end{aligned}$$

This largest coefficient can be bounded as

$$\begin{aligned}
&\max_{0\leq\ell\leq(n-1)(q-m)} \left\{ \binom{(n-1)(q-m)}{\ell} (q-m-1)^{\ell} \right\} \\
&\leq \sum_{\ell=0}^{(n-1)(q-m)} \binom{(n-1)(q-m)}{\ell} (q-m-1)^{\ell} \\
&= (q-m-1+1)^{(n-1)(q-m)} \;=\; (q-m)^{(q-m)(n-1)} \;<\; q^{q(n-1)} \;=\; q^{qn-q}.
\end{aligned}$$

Consequently, to summarize, we have established the claimed bound

$$|\beta_i| \;\leq\; \eta^m\, q^q\, q^{qn-q} \;=\; \eta^m\, q^{q+qn-q} \;=\; \eta^m\, q^{qn}.$$

To complete the proof, we apply Cauchy's bound [136] to the monic polynomial

obtained from dividing $p(x)$ by β_d and obtain that for all r with $p(r) = 0$, it holds that

$$|r| \;<\; 1 + \max_{0\le i\le d-1} \left\{\, |\beta_i/\beta_d| \,\right\} \;\le\; 1 + \max_{0\le i\le d-1} \left\{\, |\beta_i| \,\right\} \;\le\; 1 + \eta^m q^{qn}.$$

Thus, any x with $|x| \ge \eta^m q^{qn}+1$ is *not* a root of $p(x)$; equivalently, $\operatorname{rank}(H(x)_S) = q$ for such x. □

Proof of Theorem 2.5. The problem is clearly in NP: Given a subset C of column indices of A, it can be verified in polynomial time that $|C| \le k$ and that $\operatorname{rank}(A_C) = |C| - 1 = \operatorname{rank}(A_{C\setminus\{j\}})$ for every $j \in C$ (by Gaussian elimination, see, e.g., [128]).

To show hardness, we reduce the NP-complete k-Clique Problem: Given a simple undirected graph G, decide whether G has a *clique* (i.e., a vertex-induced complete subgraph) of size k. We may assume w.l.o.g. that $k > 4$.

For the given graph G with n vertices and m edges, construct a matrix $A = (a_{ie})$ of size $(n + \binom{k}{2} - k - 1) \times m$ as follows: Index the first n rows of A by the vertices of G and its columns by the edges of G (we will also identify the vertices and edges with their indices). Let the first n rows of A contain the vertex-edge incidence matrix of G (i.e., set $a_{ie} = 1$ if $i \in e$, and 0 otherwise). For the non-vertex rows $n + i$, $i \in \{1, \dots, \binom{k}{2} - k - 1\}$, set $a_{(n+i)e} = (U + i - 1)^{e-1}$ with $U := k^{2k^2m} + 1$; note that this corresponds to the bottom part of A consisting of a Vandermonde matrix (each row consists of increasing powers of the distinct numbers $U, \dots, U + \binom{k}{2} - k - 2$). Clearly, this matrix A can be constructed in polynomial time, and its encoding length is polynomially related to that of the input (in particular, that of its largest entry is $\mathcal{O}(k^2 m^2 \log_2(k))$).

We first show that G has a k-clique if and only if A has a circuit of size $\binom{k}{2}$. Suppose that G has a k-clique, $k > 4$, say on the vertices in the set R (so that $|R| = k$), and with its $\binom{k}{2}$ edges in the set C. Since A_C has all-zero rows for each vertex outside of R,

$$|R| + (\text{number of non-vertex rows}) \;=\; \binom{k}{2} - 1 \;=\; |C| - 1 \;\ge\; \operatorname{rank}(A_C).$$

Clearly, a clique is never bipartite (it always contains odd cycles, for $k \ge 3$). Hence, by Lemma 2.6, the rows of A_C indexed by R are linearly independent. Now observe that removing any edge from a k-clique does not affect the rank of the associated incidence matrix, since the subgraph remains connected and non-bipartite with fewer vertices than edges (for $k \ge 4$). Thus, by Lemma 2.6, the rank of the nonzero vertex row part of A_C remains k if any column from C is removed. Therefore, since $a_{ie} \le 1$ for all $i \le n$ and all $e \le m$, Lemma 2.8 applies to $A_{C\setminus\{e\}}$ for every $e \in C$ (with

$x = U$, $H(x) = A$, $S = C \setminus \{e\}$, and $q = \binom{k}{2} - 1$, respectively) and yields that for all $e \in C$,

$$\operatorname{rank}(A_{C\setminus\{e\}}) \;=\; k + \binom{k}{2} - k - 1 \;=\; |C| - 1.$$

Consequently, it must hold that $\operatorname{rank}(A_C) = |C| - 1$. Thus, C is a circuit.

Conversely, suppose that A has a circuit C of size $|C| = \binom{k}{2}$ with $k > 4$. Then, by definition of a circuit, $\operatorname{rank}(A_C) = |C| - 1$, so A_C has at least $|C| - 1$ nonzero rows. Since these include the $|C| - k - 1$ non-vertex rows, the set R of nonzero vertex rows of A_C has size $|R| \geq \big(|C| - 1\big) - \big(|C| - k - 1\big) = k$. Let A_{RC} and A_{NC} denote the vertex and non-vertex row submatrices of A_C, respectively. Since

$$|C| - 1 \;=\; \operatorname{rank}(A_C) \;\leq\; \operatorname{rank}(A_{NC}) + \operatorname{rank}(A_{RC}) \;=\; |C| - k - 1 + \operatorname{rank}(A_{RC}),$$

clearly $\operatorname{rank}(A_{RC}) \geq k$. Suppose that $\operatorname{rank}(A_{RC}) > k$; then there must exist a subset $R' \subseteq R$ with $|R'| = k + 1$ and $\operatorname{rank}(A_{R'C}) = k + 1$. But by Lemma 2.8, the square matrix $(A_{R'C}^\top, A_{NC}^\top)^\top$ would then have full rank $|C|$, which implies $\operatorname{rank}(A_C) = |C| > |C| - 1$, contradicting the fact that C is a circuit. Thus, the upper part A_{RC} of A_C must in fact have rank *exactly* k.

Observe that the subgraph (R, C) of G with vertex set R and edge set C cannot contain components with less than 4 vertices: Such a subgraph (R', C') could have at most as many edges as vertices, so that the associated incidence matrix $A'_{C'}$ has full *column* rank. Removing a column corresponding to an edge $e \in C'$ would reduce the rank, i.e., $\operatorname{rank}(A'_{C'\setminus\{e\}}) < \operatorname{rank}(A'_{C'})$. Moreover, note that A_{RC} has block diagonal form where the blocks are the incidence matrices of the separate graph components within (R, C), so that the rank is the sum of the ranks of the blocks (one of which is $A_{R'C'}$). In particular, the non-vertex row part of A_C maintains full (row) rank when any column is removed from C, so that deleting $e \in C'$ would yield $\operatorname{rank}(A_{C\setminus\{e\}}) = \operatorname{rank}(A_C) - 1$, contradicting the fact that C is a circuit. Thus, Lemma 2.7 applies to the graph (R, C) and yields that $|R| = k$. This implies that the vertices in R form a k-clique, because R can induce at most $\binom{k}{2}$ edges and the $\binom{k}{2}$ edges in C are among them.

We now show that each circuit of A has size *at least* $\binom{k}{2}$. This proves the claim, since by the arguments above, it shows that there exists a circuit of size *at most* (in fact, exactly) $\binom{k}{2}$ if and only if G has a k-clique, i.e., for the given construction a solution to the spark problem yields a solution to the clique problem as well.

Suppose that A has a circuit C with $c := |C| < \binom{k}{2}$, and let $d := \binom{k}{2} - c > 0$. Clearly, not all vertex rows restricted to any column subset can be zero, and any submatrix of the non-vertex part of A with fewer than $\binom{k}{2} - k$ columns is of full (column) rank. Therefore, $c > \binom{k}{2} - k$ necessarily. Since C is a circuit, A_C has $c - 1$

nonzero rows (similar to the arguments above, it can be seen that the lower bound $c-1$ holds with equality). Because the $\binom{k}{2}-k-1$ non-vertex rows are among these, and by Lemmas 2.7 and 2.8, A_C has $(c-1)-\left(\binom{k}{2}-k-1\right) = k-d > 0$ nonzero vertex rows. Denote the index set of such rows by R, and let r be the number of edges in the subgraph of G induced by the vertices in R. Since $|R| = k-d$ vertices can induce at most $\binom{k-d}{2}$ edges, $r \leq \binom{k-d}{2}$. But surely all the edges in C are among those induced by R, so that $r \geq c = \binom{k}{2} - d$. Putting these inequalities together yields $\binom{k}{2} - d \leq r \leq \binom{k-d}{2}$. However, since we assumed $k > 4$, it holds that $\binom{k}{2} - d > \binom{k-d}{2}$, yielding a contradiction. Consequently, every circuit C of A must satisfy $|C| \geq \binom{k}{2}$. □

Clearly, there exists a circuit of size at most k if and only if the spark is at most k (recall (2.4)). Hence, Theorem 2.5 immediately yields the following.

Corollary 2.9. *Computing* $\operatorname{spark}(A)$ *is NP-hard.*

Remark 2.10. The idea of reducing from the clique problem is due to Larry Stockmeyer and appears in [185, Theorem 3.3.6] (see also [3]). However, [185] uses *generic* matrices that, in fact, represent transversal matroids (of bipartite graphs) and therefore have certain properties needed in the proof. The entries of these generic matrices are not specified, and to date there is no known deterministic way to do so such that the matrix represents a transversal matroid. We replaced the corresponding machinery by our explicit matrix construction and the arguments using Lemmas 2.6, 2.7 and 2.8 to become independent of transversal matroid representations and work directly on vector matroids. Note that the proof of Theorem 2.5 also shows NP-completeness of the problem to decide whether A has a circuit C of size *equal to* k; cf. Example 2.4.

It must also be noted that the NP-hardness of the spark decision problem also appears in [66, Theorem 6.1] (cf. [67]). However, while the proof in [66] builds on the same ideas and employs a very similar matrix construction as ours, it is in fact incomplete, for similar reasons as those preventing the proof of [185, Theorem 3.3.6] to work directly for vector matroids: The argumentation in [66] implicitly assumes certain rank properties, or a kind of genericity, for submatrices which are nontrivial to establish deterministically and certainly do not hold for arbitrary matrices such as those apparently used in their construction. This could be fixed essentially along the same lines by which we extended the proof of [185] from transversal to vector matroids, using Vandermonde-like matrix parts and Lemmas 2.6–2.8.

Remark 2.11. The results above are related to, but different from, the following.

1. Theorem 1 in [148] shows that for an $m \times n$ matrix A, it is NP-complete to decide whether A has an $m \times m$ submatrix with zero determinant, which

implies NP-completeness of deciding whether $\text{spark}(A) \leq k$ for the special case $k = m$. This restriction of k to the row number m of A could in principle be removed by appending all-zero rows, but one would then no longer be in the interesting case where the matrix has full (row) rank. Our proof admits spark values other than the row number for full-rank matrices; however, the row and column numbers in the reduction depend on the instance.

2. In contrast to the results above, for graphic matroids, the girth can be computed in polynomial time [141, 185, 205]. A graphic matroid (on an undirected simple graph G) can be represented as a vector matroid via an oriented incidence matrix, i.e., the incidence matrix of a directed graph G' obtained from G by assigning an arbitrary orientation to each edge. The spark of such a matrix equals precisely the length of the shortest cycle in G.
3. The paper [249] proves NP-hardness of computing the girth of a binary matroid, i.e., a vector matroid over $\mathbb{F}_2$. This, however, does not imply NP-hardness over the field of rational or real numbers, and the proof cannot be extended accordingly. Similarly, in [20] it was shown that, over $\mathbb{F}_2$, it is NP-complete to decide whether there exists a vector with exactly k nonzero entries in the nullspace of a matrix. However, while the proof for this result can straightforwardly be extended to the rational case, it does not imply hardness of computing the spark either: Since in [20], there is no lower bound on the spark (such as we provide in the last paragraph of the proof of Theorem 2.5), a situation as in Example 2.4 is not explicitly avoided there. (Note also that it was already remarked in [20] that the problem to decide whether there exists an ($\mathbb{F}_2$-)nullspace vector with *at most* k nonzeros is not covered by their proof.)

2.3.2 Related Problems

As mentioned in [90], one can reduce spark computations to (P_0) as follows: For each column of $A \in \mathbb{Q}^{m \times n}$ in turn, add a new row with a 1 in this column and 0 elsewhere. The right hand side b is the $(m+1)$-th unit vector. Now solve each such (P_0) problem, and take the solution with smallest support. Note that this is actually a (Turing-)reduction, showing NP-hardness of (P_0) using Theorem 2.5. Interestingly, we do not know a reduction for the reverse direction.

The following result shows another relation between minimum cardinality circuits and the task of finding sparsest solutions to underdetermined linear systems:

Corollary 2.12. *Given a matrix B, a specific column b of B, and a positive integer k, the problem of deciding whether there exists a circuit of B of size k which contains b is NP-complete in the strong sense. Consequently, it is strongly NP-hard to determine the minimum cardinality of circuits that contain a specific column b of B.*

Proof. Denote by A the matrix B without the column b. Then it is easy to see that B has a circuit of size k that contains b if and only if $Ax = b$ has a solution with $k-1$ nonzero entries. Deciding the latter is well-known to be NP-complete in the strong sense (it amounts to the decision version of (P_0)), see [MP5] in [115]. □

Some algorithmic aspects of the above problem variant are discussed, from the matroid theoretical viewpoint, in [60].

The matroid perspective also allows for easy NP-hardness proofs for two very interesting problems known as the *Sparse Nullspace Basis problem (SNB)* [65, 21, 67, 68, 117, 209, 149] and *Matrix Sparsification (MS)* [185, 137, 186, 54, 55, 99]. In the former, one asks for a sparsest-possible basis for the nullspace of a given matrix, and the latter seeks a sparsest equivalent representation of a given matrix, i.e., one that spans the same row and column spaces but has as few nonzero entries as possible. The interest in these problems is fairly obvious: Many ubiquitous operations involving matrices, such as multiplication and linear system solving, can be carried out significantly faster with sparse matrices rather than general (dense) ones—thus, the fewer nonzeros in a matrix, the better.

Using basic linear algebra, the two problems SNB and MS can be seen to be polynomially equivalent (we can always express one by the other via simple polynomial transformations), see, e.g., [66, 123]. For MS, insights from matroid theory show that the "matroid greedy algorithm" (cf. [205]) is optimal and solves the problem, see, e.g., [99]. In particular, this greedy property implies that a sparsest nullspace basis must in fact contain a spark-defining circuit vector. Thus, solving SNB (or the equivalent MS instance) would also solve the spark problem, and therefore is NP-hard. Moreover, [123] shows that strong inapproximability results from [5] for (P_0) and related problems carry over to these two matrix problems, and how techniques from Compressed Sensing can be employed in heuristic approaches to MS or SNB; see also [202].

Remark 2.13. The original NP-hardness proof for SNB from [67] makes use of the result from [185], falsely attributing it to vector matroids instead of just transversal matroids; in [66], SNB hardness hinges on the incomplete proof for NP-hardness of computing the spark discussed in Remark 2.10. Thus, our Theorem 2.5 can be seen to resolve this slight inaccuracy, which had appar-

ently gone unnoticed amongst SNB/MS researchers, and reinforce the claims from [66, 67].

Let us now briefly consider full spark frames. An $m \times n$ matrix A with full rank m ($m \leq n$) is said to be *full spark* if $\operatorname{spark}(A) = m+1$, i.e., every submatrix consisting of at most m columns of A has full rank. In [3], it was shown that testing a matrix for this property is hard for NP under randomized reductions. In fact, the following stronger result holds:

Corollary 2.14. *Given a rational matrix A, deciding whether A is a full spark frame is coNP-complete.*

Proof. We can assume w.l.o.g. that $A \in \mathbb{Q}^{m\times n}$ with rank $m \leq n$. Thus, A is full spark precisely if $\operatorname{spark}(A) = m + 1$. Clearly, $\operatorname{spark}(A) = m + 1$ holds if and only if the question whether A has a singular $m \times m$ submatrix has a negative answer. Since the latter decision problem is NP-complete by [148, Theorem 1] (or [59, Proposition 4] and the results in [5]), deciding whether A is a full spark frame is NP-hard. Moreover, this problem is contained in coNP, since the "no" answer can be certified in polynomial time by specifying a singular (square) submatrix and because singularity can be verified in polynomial time, e.g., by Gaussian elimination. □

Note that above, we cannot employ Theorem 2.5, because this would require considering the matrix construction used in its proof with $k = n - 1$, and the clique problem can be solved in polynomial time $\mathcal{O}(n^{\ell+2})$ for any $k = n - \ell$ with *constant* ℓ (by enumeratively testing if for any of the $\binom{n}{n-\ell} \in \mathcal{O}(n^\ell)$ vertex subsets all $\binom{n-\ell}{2} \in \mathcal{O}(n^2)$ possible edges in the respective induced subgraph exist).

Remark 2.15. Above, and in several complexity results to follow, we show that the decision problem under consideration has a *negative* answer if and only if a known NP-complete problem has a *positive* answer. Since, by definition, the complementary problem of an NP-complete problem is coNP-complete, the respective hardness results follow; cf. [115]. Both NP- and coNP-completeness imply that no polynomial time algorithm exists unless P=NP (or equivalently P=coNP). Since a problem is NP-hard if and only if it is coNP-hard (every problem in coNP can be Turing-reduced to it; cf. [161, Chapter 15]), we use the term NP-hard throughout.

Finally, note that the spark of A is related to the *Kruskal rank* ($= \operatorname{spark}(A) - 1$) in the context of tensor decompositions [163, 160], and to *k-stability* (holds if and only if $\operatorname{spark}(A) = n - k + 1$) from matrix completion [269] (which in turn is linked to so-called Maximum Distance Separable codes; cf., e.g., [247, Chapter 5] or other introductory books on coding theory). This was already noted in [171, Lemma 5.2],

where it was later remarked that these concepts are expected (but unknown) to be computationally intractable over $\mathbb{R}$ (or $\mathbb{C}$), given the hardness results from [249] over the binary field. Indeed, by the described relations, our Theorem 2.5 immediately implies that the Kruskal rank is NP-hard to compute and k-stability is NP-hard to verify.

In the next sections, we shall see how we can deduce NP-hardness of evaluating several important sparse recovery conditions from the results about the spark.

2.4 The Restricted Isometry Property (RIP)

Many SRCs employ the famous *restricted isometry property (RIP)* (see [50] and also [14, 31]), which is satisfied with order $k \in \mathbb{N}$ and a constant δ_k by a given matrix A if

$$(1-\delta_k)\|x\|_2^2 \;\le\; \|Ax\|_2^2 \;\le\; (1+\delta_k)\|x\|_2^2 \qquad \text{for all } x \text{ with } 1 \le \|x\|_0 \le k. \tag{2.6}$$

Note that (2.6) holds with $\delta_k = 0$ if and only if A is orthogonal (i.e., $A^\top A = I$), and that $\delta_k < 0$ is impossible.

One is usually interested in the smallest constant δ_k for which (2.6) holds, i.e., the *restricted isometry constant (RIC)*

$$\underline{\delta}_k := \min_{\delta \ge 0} \; \delta \quad \text{s.t.} \quad (1-\delta)\|x\|_2^2 \le \|Ax\|_2^2 \le (1+\delta)\|x\|_2^2 \quad \text{for all } x \text{ with } 1 \le \|x\|_0 \le k. \tag{2.7}$$

For instance, a popular early result states that if $\underline{\delta}_{2k} < 1$ then (P_0) has a unique solution, and moreover, if $\underline{\delta}_{2k} < \sqrt{2}-1$, all x with at most k nonzero entries can be recovered (from A and $b := Ax$) via Basis Pursuit, see [48]. A series of papers has been devoted to developing SRCs of this flavor. The steady improvements of RIC bounds very recently culminated in the SRCs $\underline{\delta}_k < 1/3$ [44] and $\underline{\delta}_{\lceil ck \rceil} < \sqrt{(c-1)/c}$ for any constant $c \ge 4/3$ [45], which are sufficient for k-sparse recovery by ℓ_1-minimization; see also [43].

In fact, [44, 45] also establish that these SRCs are "worst-case necessary": There exist instances for which k-sparse recovery via (P_1) fails if the respective RIC bound does not hold; therefore, larger bounds cannot give sufficient SRCs in general. (For $c = 2$, [45] recovers a special case of an earlier result from [81], which showed that ℓ_p-recovery, for $0 < p \le 1$, can fail if $\underline{\delta}_{2k} \ge 1/\sqrt{2}$.) In particular, the (generally) unbridgeable gap between $1/\sqrt{2}$ and 1 shows that order-$2k$ RIP-based SRCs

cannot yield recovery for *all* unique k-sparse solutions (recall that $\underline{\delta}_{2k} < 1$ ensures uniqueness). Nevertheless, the known RIP-SRCs can provide much better (sufficient) recoverability guarantees than the mutual coherence can offer. (Note, for instance, that for A with unit ℓ_2-norm columns, $\underline{\delta}_2 = \mu(A)$ and, more generally, $\underline{\delta}_k \leq (k-1)\mu(A)$, see, e.g., [108, Proposition 2.10]. Thus, the incoherence-SRC (2.2) implies an RIP-based SRC; however, this bound is often far from sharp.) Therefore, the interest in the RIP is well-justified.

Moreover, several probabilistic results show that certain random matrices are highly likely to satisfy the RIP with desirable values of $\underline{\delta}_k$, see, for instance, [12, 31]. There also has been work on deterministic matrix constructions aiming at relatively good RIPs, see, e.g., [85, 142, 36]. The RIP also provides sparse recovery guarantees for other heuristics such as Matching Pursuit variants [80, 239], as well as for the Basis Pursuit Denoising problem (e.g., [48, 44]) and even low-rank matrix completion, see [217, 45] among many others. For instance, if $\underline{\delta}_{k+1} < 1/(1+\sqrt{k})$ then OMP recovers k-sparse signals (in k iterations), but can fail to do so already when $\underline{\delta}_{k+1} = 1/\sqrt{k}$ [189].

2.4.1 Complexity of RIP-based Recovery Conditions

In the literature, it is often mentioned that evaluating the RIP—i.e., computing the constant $\underline{\delta}_k$ for some A and k—is presumably a computationally hard problem. (Most papers seem to refer to NP-hardness, but this is often not explicitly stated.) The suspicions about computational intractability of the RIP are based on the observation that a brute-force method would have to inspect all submatrices induced by column subsets of sizes up to k. Although, by itself, this does not generally rule out the possible existence of an efficient algorithm, it motivated the development of several (polynomial-time) algorithms to approximate $\underline{\delta}_k$ by, e.g., semidefinite relaxation [78, 167]. (Note also the "lazy algorithm" from [158, 159], which sometimes allows for doing better than exhaustive search to confirm that the RIP holds for a given matrix, order and constant, by exploiting the monotonicity of the RIC: If A (with ℓ_2-normalized columns) obeys the RIP of order k_1 with constant δ_{k_1}, it also does so with constant $\delta_{k_1}(k_2-1)/(k_1-1)$ for any order $k_2 \geq k_1$ [158, Theorem 1].)

However, while a widely accepted conjecture in the CS community, NP-hardness of evaluating the RIP had, to the best of our knowledge, never actually been proven. In the following, we confirm the intractability rumors by means of several NP-hardness results. (One such result—hardness of certifying the RIP (2.6) for a given matrix A, order k *and* constant $\delta_k \in (0,1)$—was obtained independently by us [237] and by the authors of [13].)

2.4.1.1 NP-hardness of RIC Computation and RIP Certification

First, we employ our results about the spark to establish (weak) NP-hardness of computing the RIC of a given matrix A with given order k, and of the above-mentioned RIP certification problem.

We will need the following technical result.

Lemma 2.16. *Let* $A = (a_{ij}) \in \mathbb{Q}^{m\times n}$ *be a matrix,* $\alpha := \max\{\, |a_{ij}| : i \in [m],\ j \in [n] \,\}$*, and define* $C := 2^{\lceil \log_2(\alpha\sqrt{mn})\rceil}$*. Then* $\tilde{A} := \frac{1}{C}A$ *satisfies*

$$\|\tilde{A}x\|_2^2 \ \leq\ (1+\delta)\|x\|_2^2 \qquad \text{for all } x \in \mathbb{R}^n \text{ and } \delta \geq 0$$

and its encoding length is polynomial in that of A*.*

Proof. First, observe that the largest singular value of A, $\sigma_{\max}(A)$, satisfies

$$\sigma_{\max}(A) \ =\ \|A\|_2 \ \leq\ \alpha\sqrt{mn} \ =\ 2^{\log_2(\alpha\sqrt{mn})} \ \leq\ 2^{\lceil \log_2(\alpha\sqrt{mn})\rceil} \ =\ C.$$

It follows that, with an arbitrary $x \in \mathbb{R}^n$,

$$\|\tilde{A}x\|_2^2 \ \leq\ \|\tfrac{1}{C}A\|_2^2\,\|x\|_2^2 \ \leq\ \tfrac{1}{\sigma_{\max}(A)^2}\|A\|_2^2\|x\|_2^2 \ =\ \|x\|_2^2 \ \leq\ (1+\delta)\|x\|_2^2$$

for any $\delta \geq 0$. Moreover, the encoding lengths of C and of $\tilde{A}$ are straightforwardly seen to be polynomially bounded by that of A, as claimed (note that $m, n \in \mathcal{O}(\langle A\rangle)$). □

By the singular value interlacing theorem (see, e.g., [216]), $\sigma_{\max}(A)$ is an upper bound for the largest singular value of every submatrix of A. Thus, the above lemma essentially shows that by scaling the matrix A, one can focus on the lower part of the RIP (2.6) (a similar argument has been derived independently in [13]). This leads to the following complexity result.

Theorem 2.17. *Given a matrix* $A \in \mathbb{Q}^{m\times n}$ *and a positive integer* k*, the problem to decide whether there exists some constant* $\delta_k < 1$ *such that* A *satisfies the RIP of order* k *with constant* δ_k *is* coNP*-complete.*

Proof. We first show that the problem is in coNP. To certify the "no" answer, it suffices to consider a vector $\tilde{x}$ with $1 \leq \|\tilde{x}\|_0 \leq k$ that tightly satisfies (2.6) for $\delta_k = 1$. This implies $A\tilde{x} = 0$. Clearly, since $\tilde{x}$ is contained in the nullspace $\mathcal{N}(A)$ of A, we can assume that $\tilde{x}$ is rational with encoding length polynomially bounded by that of A. Then, we can verify $1 \leq \|\tilde{x}\|_0 \leq k$ and $A\tilde{x} = 0$ in polynomial time, which shows that the "no" answer can be certified in polynomial time.

To show hardness, we reduce the problem to decide whether there exists a circuit of size at most k, which is NP-complete by Theorem 2.5. (We again identify circuits by corresponding nullspace vectors to simplify notation.) Consider the matrix $\tilde{A}$ as defined in Lemma 2.16, and note that the circuits of A and $\tilde{A}$ coincide, since nonzero scaling does not affect linear dependencies among columns. We claim that there exists a circuit $\tilde{x}$ with $1 \leq \|\tilde{x}\|_0 \leq k$ if and only if $\tilde{A}$ *violates* (2.6) for *all* $\delta_k < 1$. Since deciding the former question is NP-complete, this completes the proof.

Clearly, if such an $\tilde{x} \neq 0$ exists, then

$$(1 - \delta_k)\|\tilde{x}\|_2^2 \;\leq\; \|\tilde{A}\tilde{x}\|_2^2 \;=\; 0$$

implies that we must have $\delta_k \geq 1$.

For the converse, assume that there does not exist any $\delta_k < 1$ for which (2.6) holds. By Lemma 2.16, the upper part of (2.6) is always satisfied. Consequently, there must exist a vector $\hat{x}$ with $1 \leq \|\hat{x}\|_0 \leq k$ such that the lower part is tight, i.e.,

$$0 \;\geq\; (1 - \delta_k)\|\hat{x}\|_2^2 \;=\; \|\tilde{A}\hat{x}\|_2^2 \;\geq\; 0.$$

This implies that $\hat{x} \in \mathcal{N}(\tilde{A})$, i.e., $\tilde{A}\hat{x} = 0$. Thus, there also exists a *circuit* $\tilde{x}$ with $1 \leq \|\tilde{x}\|_0 \leq k$, which shows the claim. □

Obviously, knowledge of the value of $\underline{\delta}_k$ allows us to answer the decision problem of Theorem 2.17. Hence, we immediately obtain the following complexity result.

Corollary 2.18. *For a given matrix $A \in \mathbb{Q}^{m \times n}$ and positive integer k, it is NP-hard to compute the RIC $\underline{\delta}_k$.*

Remark 2.19. Clearly, under the standard assumption that $\operatorname{rank}(A) = m \leq n$, it must necessarily hold that $\underline{\delta}_k \geq 1$ for any $k > m$, since $\operatorname{spark}(A) \leq m + 1$.

As seen at the beginning of Section 2.4, in RIP-based SRCs that establish (for instance) ℓ_0-ℓ_1-equivalence, one is particularly interested in whether the RIP is satisfied with a specific $\delta_k = \delta \in (0, 1)$ (or whether $\underline{\delta}_k < \delta$) for some *fixed* $\delta < 1$; this type of question differs from the one considered in Theorem 2.17. Nevertheless, we show in the following that the RIP certification problem—i.e., deciding whether a matrix A satisfies the RIP with given order k and *given* constant $\delta_k \in (0, 1)$—is indeed also (co)NP-hard. (As mentioned earlier, the main arguments used in the proofs of the following lemma and theorem have also been independently derived by the authors of [13].)

The following observation is essential.

Lemma 2.20. *Given a matrix $A \in \mathbb{Q}^{m\times n}$ and a positive integer k, if* $\operatorname{spark}(A) > k$*, then there exists a constant $\varepsilon > 0$ with encoding length polynomially bounded by that of A such that $\|Ax\|_2^2 \geq \varepsilon \|x\|_2^2$ for all x with $\|x\|_0 \leq k$.*

Proof. Assume w.l.o.g. that A has only integer entries (this can always be achieved by scaling with the least common denominator of the matrix entries, which influences ε by a polynomial factor only).

Define α as in Lemma 2.16. Note that $\operatorname{spark}(A) > k$ implies that every submatrix A_S with $S \subseteq [n]$, $|S| \leq k$, has linearly independent columns. Consider an arbitrary such S; then, $A_S^\top A_S$ is positive definite, so its smallest eigenvalue fulfills $\lambda_{\min}(A_S^\top A_S) > 0$, and also, $\det(A_S^\top A_S) > 0$. Moreover, since the absolute values of entries of A are integers in $\{0, 1, \ldots, \alpha\}$, the absolute values of the entries of $A_S^\top A_S$ are also integral and lie in $\{0, 1, \ldots, m\,\alpha^2\}$. Hence, it must in fact hold that $\det(A_S^\top A_S) \geq 1$ and $\lambda_{\min}(A_S^\top A_S) \geq 1$. Using the well-known eigenvalue-based identity of the determinant, it follows that

$$\begin{aligned} 1 \;\leq\; \det(A_S^\top A_S) \;&=\; \prod_{i=1}^{|S|} \lambda_i(A_S^\top A_S) \;\leq\; \lambda_{\min}(A_S^\top A_S) \cdot \lambda_{\max}(A_S^\top A_S)^{k-1} \\ &\leq\; \lambda_{\min}(A_S^\top A_S) \left(|S| \cdot \max_{1\leq i,j\leq |S|} |(A_S^\top A_S)_{ij}| \right)^{k-1} \;\leq\; \lambda_{\min}(A_S^\top A_S) \left(k\,m\,\alpha^2\right)^{k-1}. \end{aligned}$$

Consequently, we have that

$$\lambda_{\min}(A_S^\top A_S) \;\geq\; \frac{1}{(k\,m\,\alpha^2)^{k-1}} \;=:\; \varepsilon \;>\; 0. \tag{2.8}$$

Since S was arbitrary, $\|Ax\|_2^2 \geq \lambda_{\min}(A_S^\top A_S)\|x\|_2^2 \geq \varepsilon\|x\|_2^2$ holds for all x with support $S \subseteq [n]$, $|S| \leq k$. Moreover, the encoding length of α, and therefore that of ε, is clearly polynomially bounded by the encoding length of A, which completes the proof. □

Theorem 2.21. *Given a matrix $A \in \mathbb{Q}^{m\times n}$, a positive integer k, and some rational constant $\delta_k \in (0, 1)$, it is NP-hard to decide whether A satisfies the RIP of order k with constant δ_k.*

Proof. Consider a matrix $\tilde{A}$ as in Lemma 2.16, so we can again focus on the lower inequality of the RIP. Clearly, if $\tilde{A}$ has a circuit of size at most k, $\tilde{A}$ cannot satisfy the RIP of order k with any given $\delta_k \in (0, 1)$, since in this case, $\|\tilde{A}\tilde{x}\|_2^2 = 0 < (1-\delta_k)\|\tilde{x}\|_2^2$ for some $\tilde{x}$ with $1 \leq \|\tilde{x}\|_0 \leq k$. Moreover, Lemma 2.20 shows that if $\tilde{A}$ has no circuit of size at most k, $\tilde{A}$ satisfies the RIP of order k with constant $1 - \tilde{\varepsilon} \in (0, 1)$, where $\tilde{\varepsilon}$ has encoding length polynomially bounded by k and that of $\tilde{A}$, cf. (2.8).

By Theorem 2.17, it is coNP-complete to decide whether there exists a constant $\delta_k < 1$ such that $\tilde{A}$ satisfies the RIP of a given order k with this constant. But as seen above, such a constant exists if and only if $\tilde{A}$ satisfies the RIP of k with the constant $1 - \tilde{\varepsilon}$, too. Thus, deciding whether the RIP holds for a given matrix, order *and* constant is (co)NP-hard. □

In fact, we also obtain NP-hardness for the problem variant which asks to certify that $\underline{\delta}_k < \delta$ for a given $\delta \in (0, 1)$:

Corollary 2.22. *Given a matrix $A \in \mathbb{Q}^{m \times n}$, a positive integer k, and some rational constant $\delta \in (0, 1)$, it is NP-hard to decide whether there exists some constant $\delta_k < \delta$ such that A satisfies the RIP of order k with constant δ_k.*

Proof. The proof of Theorem 2.21 can be directly extended to the problem at hand by setting $\delta = 1 - \tilde{\varepsilon}/2$ (or any other value $1 - \tilde{\varepsilon}/r$ with $1 < r \in \mathbb{Q}$). □

Remark 2.23. It is an open question whether the decision problems in Theorem 2.21 and Corollary 2.22 are contained in coNP.

Remark 2.24. Clearly, Theorem 2.21 leads to another easy proof for Corollary 2.18 (and Corollary 2.25 below): Computing the (lower asymmetric) RIC would also decide the RIP certification problem. On the other hand, our proof of Theorem 2.21 essentially reduces the RIP certification problem to the setting of Theorem 2.17, which therefore can be seen as a core RIP hardness result (by establishing the direct link to spark computations); see also Remark 2.34 below.

2.4.1.2 Asymmetric RIP Constants and Strong NP-hardness of Sparse PCA and RIC Computation

It has been remarked in [31] that the symmetric nature of the RIP can be overly restrictive. In particular, the influence of the upper inequality in (2.7) is often stronger, although the lower inequality is more important in the context of sparse recovery. For instance, the often stated condition $\underline{\delta}_{2k} < 1$ for uniqueness of k-sparse solutions (see, e.g., [48]) should in fact read $\underline{\delta}^L_{2k} < 1$, where

$$\underline{\delta}^L_k := \min_{\delta \geq 0} \delta \quad \text{s.t.} \quad (1 - \delta)\|x\|_2^2 \leq \|Ax\|_2^2 \quad \forall x : \|x\|_0 \leq k$$

is the lower *asymmetric* restricted isometry constant [31] (see also [110]). Correspondingly, the upper asymmetric RIC is

$$\underline{\delta}^U_k := \min_{\delta \geq 0} \delta \quad \text{s.t.} \quad (1 + \delta)\|x\|_2^2 \geq \|Ax\|_2^2 \quad \forall x : \|x\|_0 \leq k.$$

The central argument in the proof of Theorem 2.17 in fact shows the following:

Corollary 2.25. *Given a matrix $A \in \mathbb{Q}^{m\times n}$ and a positive integer k, it is NP-hard to compute the lower asymmetric RIC $\underline{\delta}_k^L$.*

Moreover, the next result settles the computational complexity of computing the upper asymmetric RIC $\underline{\delta}_k^U$.

Theorem 2.26. *Given a matrix $A \in \mathbb{Q}^{m\times n}$, a positive integer k and a rational parameter $\delta > 0$, it is NP-hard in the strong sense to decide whether $\underline{\delta}_k^U < \delta$, even in the square case $m = n$. Consequently, it is strongly NP-hard to compute the upper asymmetric RIC $\underline{\delta}_k^U$.*

To prove this theorem, we need some auxiliary results. In particular, we will show that solving the so-called sparse principal component analysis problem is strongly NP-hard (see Proposition 2.29 and Remark 2.30 below). We then extend the arguments to prove Theorem 2.26 and thus, ultimately, that the NP-hardness of RIC computation in fact also holds in the strong sense (Corollary 2.32).

Lemma 2.27. *Let $G = (V, E)$ be a simple undirected graph with $n = |V| \geq 2$ and let A_G be its $n \times n$ adjacency matrix, i.e., $(A_G)_{ij} = 1$ if $\{i, j\} \in E$ and 0 otherwise (in particular, on the diagonal). Denote by K_n the complete graph with n vertices.*

1. *If $G = K_n$, i.e., G is a clique, then A_G has eigenvalues -1 and $n - 1$ with respective multiplicities $n - 1$ and 1.*
2. *Removing an edge from G does not increase $\lambda_{\max}(A_G)$. In fact, if G is connected, this strictly decreases $\lambda_{\max}(A_G)$.*
3. *If $G = K_n \setminus e$, i.e., a clique with any one edge $e \in E$ removed, then the largest eigenvalue of A_G is $(n - 3 + \sqrt{n^2 + 2n - 7})/2$.*

Proof. The first two statements can be found in, or deduced easily from, [39, Chapter 1.4.1 and Proposition 3.1.1], respectively. The third result is a special case of [112, Theorem 1]. □

Remark 2.28. Lemma 2.27 shows that, in a graph $G = (V, E)$ with $|V| \geq 2$, the largest eigenvalue of the adjacency matrix of any induced subgraph with $k \in \{2, \ldots, |V|\}$ vertices is either $k - 1$ (if and only if the subgraph is a k-clique) or at most $(k - 3 + \sqrt{k^2 + 2k - 7})/2$.

Proposition 2.29. *Given a matrix $H \in \mathbb{Q}^{n\times n}$, a positive integer $k \le n$ and a parameter $\lambda > 0$, it is coNP-complete in the strong sense to decide whether $\lambda^{(k)}_{\max} < \lambda$, where*

$$\begin{aligned}\lambda^{(k)}_{\max} &:= \max\{\, x^\top H x \,:\, \|x\|_2^2 = 1,\ \|x\|_0 \le k \,\} \\ &= \max\{\, \lambda_{\max}(H_{SS}) \,:\, S \subseteq [n],\ |S| \le k \,\}.\end{aligned} \tag{2.9}$$

Consequently, solving (2.9)—known as the sparse principal component analysis (Sparse PCA) problem*—is strongly NP-hard.*

Proof. We reduce from the k-Clique Problem. Let $G = (V, E)$ be a simple undirected graph with n vertices (w.l.o.g., $n \ge k \ge 2$). From G, construct its $n \times n$ adjacency matrix A_G. By the previous lemma, G contains a k-clique if and only if $\lambda^{(k)}_{\max}(A_G) = k-1 =: \lambda$. (Note that $\lambda^{(k)}_{\max} \le k-1$ always holds by construction.) Hence, the Sparse PCA decision problem has a negative answer for the instance (A_G, k, λ) if and only if the k-Clique Problem has a positive answer. Since the latter is NP-complete in the strong sense, and because all numbers appearing in the constructed Sparse PCA instance, and their encoding lengths, are polynomially bounded by n, the former is strongly (co)NP-hard.

Moreover, consider a "no" instance of the Sparse PCA decision problem. Then, as we just saw, there is a k-clique S in G, and it is easily verified that $\hat{x}$ with $\hat{x}_i = 1/\sqrt{k}$ for $i \in S$, and zeros everywhere else, achieves $\hat{x}^\top A_G\, \hat{x} = \lambda^{(k)}_{\max}(A_G) = \lambda$. Scaling (2.9) by k, we see that *equivalently*,

$$k\,\lambda^{(k)}_{\max} \;=\; \max\{\, x^\top A_G\, x \,:\, \|x\|_2^2 = k,\ \|x\|_0 \le k \,\} \;=\; k\,\lambda \;=\; k^2 - k.$$

Thus, a *rational* certificate for the "no" answer is given by the vector $\tilde{x}$ with $\tilde{x}_i = 1$ for $i \in S$, and zeros everywhere else. Since we can clearly check all constraints on $\tilde{x}$ (from the scaled problem) and that $\tilde{x}^\top A_G\, \tilde{x} = k\lambda$ in polynomial time, the Sparse PCA decision problem is contained in coNP. □

Remark 2.30. The Sparse PCA problem (see, e.g., [177, 144, 77, 78, 270]) is often mentioned to be (NP-)hard, but we could not locate a rigorous proof of this fact. In [23, Section 6], the authors sketch a reduction from the k-Clique Problem but do not give the details; the central spectral argument mentioned there, however, is exactly what we exploit in the above proof.

We will extend the proof of Proposition 2.29 to show Theorem 2.26 by suitably approximating the Cholesky decomposition of a matrix very similar to the adjacency matrix; the following technical result will be useful for this extension.

Lemma 2.31. *Let A_G be the adjacency matrix of a simple undirected graph $G = (V, E)$ with n vertices, and let $H := A_G + n^2 I$. Then H has a unique Cholesky factorization $H = LDL^\top$ with diagonal matrix $D = \mathrm{Diag}(d_1, \ldots, d_n) \in \mathbb{Q}^{n\times n}$ and unit lower triangular matrix $L = (\ell_{ij}) \in \mathbb{Q}^{n\times n}$, whose respective entries fulfill (for all $i, j \in [n]$, $i \neq j$)*

$$d_i \in \left[\frac{n^4 - 2n + 2}{n^2}, n^2\right] \quad \text{and} \quad \ell_{ij} \in \left[\frac{2 - n^2 - 2n}{n^4 - 2n + 2}, \frac{2n - 2}{n^4 - 2n + 2}\right].$$

Proof. With $\deg(v)$ denoting the degree of a vertex v of G, and $\lambda_i(A_G)$, $i = 1, \ldots, n$, the eigenvalues of A_G, it holds that

$$\|A_G\|_2 \;=\; \max_{1\le i\le n} \left\{\, |\lambda_i(A_G)| \,\right\} \;\le\; \sqrt{n}\, \|A_G\|_\infty \;<\; n \max_{v\in V} \left\{\, \deg(v) \,\right\} \;<\; n^2.$$

Thus, it is easy to see that $H = A_G + n^2 I$ is (symmetric) positive definite; in particular, the eigenvalues of H obey $\lambda_i(H) = \lambda_i(A_G) + n^2 > 0$ for all i. Then, H has a unique Cholesky factorization $H = LDL^\top$, and $L \in \mathbb{Q}^{n\times n}$ (unit lower triangular) and $D \in \mathbb{Q}^{n\times n}$ (diagonal) can be obtained by Gaussian elimination; see, e.g., [118, Section 4.9.2] or [121, Section 4.2.3].

Let $H^{(0)} := H$ and let $H^{(k)} = (h^{(k)}_{ij})$ be the matrix obtained from H after k iterations of (symmetric) Gaussian elimination. There are $n - 1$ such iterations, and each matrix $H^{(k)}$ has block structure with $\mathrm{Diag}(d_1, \ldots, d_k)$ in the upper left part and a symmetric positive definite matrix in the lower right block. We show by induction that for all $k \in [n]$, and all $i, j \in \{0, \ldots, n - k\}$,

$$h^{(k-1)}_{k+i,k+i} \in \left[n^2 - \frac{2(k-1)}{n^2}, n^2\right] \tag{2.10}$$

and (for $i \neq j$)

$$h^{(k-1)}_{k+i,k+j} \in \left[-\frac{2(k-1)}{n^2}, 1 + \frac{2(k-1)}{n^2}\right]. \tag{2.11}$$

Clearly, by construction of H, $h^{(0)}_{ii} = n^2$ and $h^{(0)}_{ij} \in \{0, 1\}$ for all $i, j \in [n]$, $i \neq j$, so (2.10) and (2.11) hold true for $k = 1$.

Suppose they hold for some $k \in [n-1]$, i.e., throughout the first $k-1$ iterations of the Gaussian elimination process. Performing the k-th iteration, we obtain $h^{(k)}_{ik} = h^{(k)}_{ki} = 0$ for all $i > k$, $h^{(k)}_{kk} = h^{(k-1)}_{kk}$, and

$$h^{(k)}_{k+i,k+j} \;=\; h^{(k-1)}_{k+i,k+j} - \frac{h^{(k-1)}_{k+i,k}}{h^{(k-1)}_{kk}} \cdot h^{(k-1)}_{k,k+j} \qquad \text{for all } i, j \in [n-k]. \tag{2.12}$$

Thus, in particular, by symmetry of $H^{(k-1)}$, for any $i \in [n-k]$,

$$h^{(k)}_{k+i,k+i} \;=\; h^{(k-1)}_{k+i,k+i} - \frac{h^{(k-1)}_{k+i,k}}{h^{(k-1)}_{kk}} \cdot h^{(k-1)}_{k,k+i} \;=\; h^{(k-1)}_{k+i,k+i} - \frac{\big(h^{(k-1)}_{k+i,k}\big)^2}{h^{(k-1)}_{kk}}. \tag{2.13}$$

Applying the induction hypothesis (2.10) to (2.13) yields the first interval inclusion (note that $h^{(k-1)}_{kk} > 0$, so $h^{(k)}_{k+i,k+i} \le h^{(k-1)}_{k+i,k+i}$):

$$\begin{aligned} n^2 \;\ge\; h^{(k-1)}_{k+i,k+i} \;\ge h^{(k)}_{k+i,k+i} \;&\ge\; n^2 - \frac{2(k-1)}{n^2} - \frac{\big(1+\frac{2(k-1)}{n^2}\big)^2}{\big(n^2-\frac{2(k-1)}{n^2}\big)} \\ &= n^2 - \frac{2(k-1)}{n^2} - \underbrace{\frac{n^2+4(k-1)+\frac{4(k-1)^2}{n^2}}{n^4-2k+2}}_{\le\, 2/n^2} \;\ge\; n^2 - \frac{2k}{n^2}. \end{aligned} \tag{2.14}$$

Similarly, for the off-diagonal entries $h^{(k)}_{k+i,k+j}$ with $i,j \in [n-k]$, $i \ne j$, from (2.12) and (2.11) we obtain

$$\begin{aligned} h^{(k)}_{k+i,k+j} \;&\le 1 + \frac{2(k-1)}{n^2} - \frac{\big(-\frac{2(k-1)}{n^2}\big)\big(1+\frac{2(k-1)}{n^2}\big)}{\big(n^2-\frac{2(k-1)}{n^2}\big)} \\ &= 1 + \frac{2(k-1)}{n^2} + \underbrace{\frac{2(k-1)+\frac{4(k-1)^2}{n^2}}{n^4-2k+2}}_{\le\, 2/n^2} \;\le\; 1 + \frac{2k}{n^2} \end{aligned}$$

and (compare with (2.14))

$$h^{(k)}_{k+i,k+j} \;\ge\; -\frac{2(k-1)}{n^2} - \frac{\big(1+\frac{2(k-1)}{n^2}\big)^2}{\big(n^2-\frac{2(k-1)}{n^2}\big)} \;\ge\; \frac{2-2k}{n^2} - \frac{2}{n^2} \;=\; -\frac{2k}{n^2},$$

which shows the second interval inclusion and completes the induction.

The statement of the lemma now follows from observing that $d_k = h^{(k-1)}_{kk}$ for all $k \in [n]$, and because the entries in the lower triangular part of L, i.e., ℓ_{ij} with $i > j$, $j \in [n-1]$, contain precisely the negated elimination coefficients (from the j-th iteration, respectively), whence

$$\begin{aligned} \frac{2-n^2-2n}{n^4-2n+2} \;=\; -\frac{\big(1+\frac{2(n-1)}{n^2}\big)}{\big(n^2-\frac{2(n-1)}{n^2}\big)} \;\le\; \ell_{ij} \;&=\; -\frac{h^{(j-1)}_{ij}}{h^{(j-1)}_{jj}} \;=\; -\frac{h^{(j-1)}_{ij}}{d_j} \\ &\le\; -\frac{\big(-\frac{2(n-1)}{n^2}\big)}{\big(n^2-\frac{2(n-1)}{n^2}\big)} \;=\; \frac{2n-2}{n^4-2n+2}. \end{aligned}$$

(By construction, $\ell_{ii} = 1$ and $\ell_{ij} = 0$ for all $i \in [n]$, $j > i$.) This concludes the proof. □

Proof of Theorem 2.26. Given an instance (G, k) for the k-Clique Problem, we construct $H = A_G + n^2 I$ from the graph's adjacency matrix A_G (w.l.o.g., $n \geq 2$). From the proof of Proposition 2.29, recall that G has no k-clique if and only if $\lambda^{(k)}_{\max}(A_G) < k - 1$, or equivalently $\lambda^{(k)}_{\max}(H) < n^2 + k - 1$ (cf. the beginning of the proof of Lemma 2.31). Let D and L be the Cholesky factors of H, i.e., $H = LDL^\top$. Letting $D^{1/2} \coloneqq \mathrm{Diag}(\sqrt{d_1}, \ldots, \sqrt{d_n})$, observe that the upper asymmetric RIC for the matrix $A' \coloneqq D^{1/2}L^\top$ can be written as

$$\underline{\delta}^U_k(A') \;=\; \max\Big\{ x^\top L D^{1/2} D^{1/2} L^\top x \;:\; \|x\|_2^2 = 1,\; \|x\|_0 \leq k \Big\} - 1 \;=\; \lambda^{(k)}_{\max}(H) - 1.$$

Consequently, G has a k-clique S if and only if H has a $k \times k$ submatrix H_{SS} with largest eigenvalue $n^2 + k - 1$, i.e., $\underline{\delta}^U_k(A') = n^2 + k - 2$. Moreover, by Lemma 2.27,

$$\lambda_{\max}(H_{TT}) \;\leq\; n^2 + (k - 3 + \sqrt{k^2 + 2k - 7})/2 \;<\; n^2 + k - 1$$

for any incomplete induced subgraph of G with vertex set T, $|T| = k$. However, while L and D are rational, $D^{1/2}$ can contain *irrational* entries, so we cannot directly use A' as the input matrix for the upper asymmetric RIC decision problem. The remainder of this proof shows that we can replace $D^{1/2}$ by a rational approximation to within an accuracy that still allows us to distinguish between eigenvalues associated to k-cliques and those belonging to incomplete induced subgraphs.

Let us consider the rational approximation obtained by truncating each $\sqrt{d_i}$ after the p-th decimal number (we will specify p later), i.e., let $\tilde{D}^{1/2} \coloneqq \mathrm{Diag}(r_1, \ldots, r_n)$ with $r_i \coloneqq \lfloor 10^p \sqrt{d_i} \rfloor / 10^p$. Thus, $r_i \leq \sqrt{d_i}$ and $\sqrt{d_i} - r_i \leq 10^{-p}$ for all i by construction, and in particular, since $d_1 = n^2$ (see Lemma 2.31), $r_1 = n$. Consequently,

$$\|D^{1/2} - \tilde{D}^{1/2}\|_2 \;=\; \max_{1 \leq i \leq n} \big\{ |\sqrt{d_i} - r_i| \big\} \;=\; \max_{2 \leq i \leq n} \big\{ \sqrt{d_i} - r_i \big\} \;\leq\; 10^{-p}.$$

Denoting $\tilde{D} \coloneqq \tilde{D}^{1/2}\tilde{D}^{1/2}$ and using $d_i \leq n^2$ (true by Lemma 2.31), we obtain

$$\begin{aligned} \|D - \tilde{D}\|_2 \;&=\; \max_{2 \leq i \leq n} \big\{ d_i - r_i^2 \big\} \;=\; \max_{2 \leq i \leq n} \big\{ (\sqrt{d_i} + r_i)(\sqrt{d_i} - r_i) \big\} \\ &\leq\; 10^{-p} \cdot 2 \cdot \max_{2 \leq i \leq n} \big\{ \sqrt{d_i}, r_i \big\} \;\leq\; 2 \cdot 10^{-p} \cdot n. \end{aligned}$$

Let $\tilde{H} \coloneqq L\tilde{D}L^\top$ and note that, for all $i, j \in [n]$, we have $h_{ij} = \sum_{q=1}^n d_q \ell_{iq} \ell_{jq}$ and $\tilde{h}_{ij} = \sum_{q=1}^n r_q^2 \ell_{iq} \ell_{jq}$. Since $|\ell_{ij}| \leq 1$ for all i, j (by Lemma 2.31), and by symmetry

of H and $\tilde{H}$, it follows that

$$|(H-\tilde{H})_{ij}| \;=\; |(H-\tilde{H})_{ji}| \;=\; \Big|\sum_{q=1}^{n}(d_q - r_q^2)\ell_{iq}\ell_{jq}\Big|$$
$$\leq \sum_{q=1}^{n}|d_q - r_q^2|\,|\ell_{iq}|\,|\ell_{jq}| \;\leq\; \sum_{q=1}^{n}(d_q - r_q^2) \;\leq\; 2\cdot 10^{-p}\cdot n^2.$$

Thus, we have $\tilde{H} = H + \tilde{E}$, where $|\tilde{E}_{ij}| \leq 2\cdot 10^{-p}\cdot n^2$ and $\tilde{E}$ is also symmetric. Note that, for any $S \subseteq [n]$,

$$\lambda_{\max}(\tilde{E}_{SS}) \;\leq \lambda_{\max}(\tilde{E}) \;=\; \|\tilde{E}\|_2 \;\leq\; \sqrt{\|\tilde{E}\|_1\cdot\|\tilde{E}\|_\infty}$$
$$\leq \sqrt{(n\cdot 2\cdot 10^{-p}\cdot n^2)(n\cdot 2\cdot 10^{-p}\cdot n^2)} \;=\; 2\cdot 10^{-p}\cdot n^3.$$

Therefore (cf., e.g., [121, Corollary 8.1.6]), we have for all $S \subseteq [n]$ that

$$|\lambda_i(H_{SS}) - \lambda_i(\tilde{H}_{SS})| \;\leq\; \|\tilde{E}_{SS}\|_2 \;=\; \lambda_{\max}(\tilde{E}_{SS}) \;\leq\; 2\cdot 10^{-p}\cdot n^3$$

for all $i = 1,\ldots,|S|$. Consequently, if S induces a k-clique, we have

$$\lambda_{\max}(\tilde{H}_{SS}) \;\geq\; n^2 + k - 1 - 2\cdot 10^{-p}\cdot n^3, \tag{2.15}$$

whereas for any $T \subseteq [n]$ with $|T| = k$ that does not induce a clique, it holds that

$$\lambda_{\max}(\tilde{H}_{TT}) \;\leq\; n^2 + \frac{k-3+\sqrt{k^2+2k-7}}{2} + 2\cdot 10^{-p}\cdot n^3. \tag{2.16}$$

Now fix $p := 1 + \lceil 4\log_{10}(n)\rceil$, and let an instance for the upper asymmetric RIC decision problem be given by $A := \tilde{D}^{1/2}L^\top$ (where $\tilde{D}^{1/2}$ is computed from the Cholesky factors L and D of $H = A_G + n^2 I$ using this precision parameter p), $\delta := n^2 + k - 2 - 2\cdot 10^{-p}\cdot n^3$, and k.

If G has a k-clique then, by (2.15), $\underline{\delta}_k^U(A) \geq \delta$, and if not, $\underline{\delta}_k^U(A) \leq n^2 + (k-3+\sqrt{k^2+2k-7})/2 + 2\cdot 10^{-p}\cdot n^3 - 1$ by (2.16). In fact, our choice of p implies

$$p \;>\; \log_{10}(8) + 4\log_{10}(n) \quad\Rightarrow\quad 10^p \;>\; 8n^4 \;>\; \frac{8n^3}{k+1-\sqrt{k^2+2k-7}},$$

from which we can derive that

$$n^2 + k - 1 - 2\cdot 10^{-p}\cdot n^3 \;>\; n^2 + \frac{k-3+\sqrt{k^2+2k-7}}{2} + 2\cdot 10^{-p}\cdot n^3,$$

which shows that $\underline{\delta}_k^U(A) < \delta$ if no k-clique is contained in G. Therefore, G has a k-clique if and only if $\underline{\delta}_k^U(A) \geq \delta$, i.e., the upper asymmetric RIC decision problem under consideration has a negative answer.

Clearly, all computations in the above reduction can be performed in polynomial time (in particular, this holds true for rational approximation, cf. [128, Section 1.3]). To see that the encoding length $\langle A \rangle$ of A is in fact polynomially bounded by n, note that $h_{ij} \in \{0, 1, n^2\}$ for all i, j, whence $\langle H \rangle \in \mathcal{O}(\text{poly}(n))$. Since Gaussian elimination can be implemented to lead only to a polynomial growth of the encoding lengths [128], it follows that $\langle L \rangle, \langle D \rangle \in \mathcal{O}(\text{poly}(\langle H \rangle)) = \mathcal{O}(\text{poly}(n))$. In particular, the entries of $\tilde{D}^{1/2}$ then also have encoding length polynomially bounded by n, by construction of the rational approximation. This shows that indeed $\langle a_{ij} \rangle \in \mathcal{O}(\text{poly}(n))$ for all i, j. Furthermore, all the numerical values a_{ij} are also polynomially bounded by n (in fact, $|a_{ij}| \leq n$, by Lemma 2.31 and the construction of $\tilde{D}^{1/2}$). Moreover, $0 < \delta < n^2 + n$ and, clearly, its encoding length $\langle \delta \rangle$ is bounded polynomially by n as well.

Thus, since the Clique Problem is well-known to be NP-complete in the strong sense, and because our polynomial reduction in fact preserves boundedness of the numbers within $\mathcal{O}(\text{poly}(n))$, the upper asymmetric RIC decision problem is (co)NP-hard in the strong sense. This completes the proof of Theorem 2.26. □

In fact, observe that for the matrix A constructed in the proof of Theorem 2.26, the upper asymmetric RIC is always larger than the lower asymmetric RIC, whence the former coincides with the (symmetric) RIC $\underline{\delta}_k$ of A. Thus, the following result holds true, which slightly strengthens Corollary 2.18.

Corollary 2.32. *Given a matrix $A \in \mathbb{Q}^{m \times n}$ and a positive integer k, it is NP-hard in the strong sense to compute the RIC $\underline{\delta}_k$, even in the square case $m = n$.*

Remark 2.33. Similarly to the RIP certification problem, containment in coNP of the decision problem from Theorem 2.26 remains an open question. (Although we reduce from the Sparse PCA decision problem from Proposition 2.29, we may not simply use the same certificate because it is not *generally* valid for the upper asymmetric RIC problem, only for the instance constructed in the above proof.)

It is noteworthy that we can easily extend the construction from the proof of Theorem 2.26 to cover the non-square case $m < n$ as well. The idea is as follows: For instance, let B be the matrix from Example 2.4, and replace A in the above proof by the block diagonal matrix $\hat{A}$ which has A in the first block and B in the second. This matrix has dimensions $(n + 3) \times (n + 4)$, and since the eigenvalues of $B^\top B$ are contained in $[0, 5)$, it is easy to see that, whenever $n \geq 3$ and $k \geq 3$

(which can be assumed without loss of generality in the extended proof), $n^2 + (k - 3 + \sqrt{k^2 + 2k - 7})/2 - 2 \cdot 10^{-p} \cdot n^3 \geq 5$. Then, the relation between eigenvalues of symmetrically chosen submatrices and cliques is indeed the same for $\hat{A}^\top \hat{A}$ as for $A^\top A$ itself, and the extended proof works completely analogously to the one above. A similar idea is exploited in the proof of [159, Theorem 6].

Remark 2.34. The (weak) NP-hardness result Theorem 2.17 (which yielded Corollary 2.18) remains of interest in its own right, as it reveals, for instance, the intrinsic relation between the k-sparse solution uniqueness conditions $2k < \operatorname{spark}(A)$ and $\underline{\delta}_{2k} < 1$ (or $\underline{\delta}^L_{2k} < 1$, respectively, see Corollary 2.25); consequently, verifying uniqueness via these conditions is NP-hard because deciding whether $\operatorname{spark}(A) \leq k$ is.

Moreover, were we to scale down A in the above proof to achieve $\delta < 1$, we could not be certain anymore that the RIC coincides with the *upper* asymmetric RIC, not the *lower*. (In fact, as seen in Lemma 2.16, we can eliminate the dependence of the symmetric RIC on the upper RIP part completely by scaling A to have $\|A\|_2 \leq 1$.) Hence, in particular, we cannot simply transfer the "strong" part of the above NP-hardness result to the RIP certification problems considered in Theorem 2.21 and Corollary 2.22 (which require $\delta < 1$ by definition). Nevertheless, one could say that NP-hardness of computing the RIC $\underline{\delta}_k$ persists independently of the scaling of A, either by Theorem 2.17 (Corollary 2.18) or Theorem 2.26 (Corollary 2.32).

2.5 The Nullspace Property (NSP)

Another popular tool for guaranteeing ℓ_0-ℓ_1-equivalence is the *nullspace property (NSP)*, see, e.g., [92, 266, 64, 145], which characterizes recoverability by (P_1) for sufficiently sparse solutions of (P_0). (In fact, this extends to ℓ_p-minimization with $0 < p \leq 1$, see [126, 81].) The NSP of order k is satisfied with constant α_k if for all vectors $x \in \mathcal{N}(A)$, it holds that

$$\|x\|_{k,1} \;\leq\; \alpha_k \|x\|_1, \tag{2.17}$$

where $\|x\|_{k,1}$ denotes the sum of the k largest absolute values of entries in x. The NSP guarantees uniqueness of k-sparse optimal solutions of (P_1) whenever (2.17) holds with some constant $\alpha_k < 1/2$. Hence, similar to the RIP case, one is interested in the smallest constant such that (2.17) is fulfilled (given A and k); we call this the *nullspace constant (NSC)*, which can be formally defined as

$$\underline{\alpha}_k := \min \alpha \quad \text{s.t.} \quad \|x\|_{k,1} \leq \alpha \|x\|_1 \quad \text{for all } x \text{ with } Ax = 0,$$

or equivalently,

$$\underline{\alpha}_k \coloneqq \max \|x_S\|_1 \quad \text{s.t.} \quad Ax = 0,\ \|x\|_1 = 1,\ S \subseteq [n],\ |S| \leq k.$$

Indeed, if and only if $\underline{\alpha}_k < 1/2$, then any instance of ($P_1$) with $b \coloneqq A\tilde{x}$, $\|\tilde{x}\|_0 \leq k$, has the unique solution $\tilde{x}$. Most importantly, it can be shown that $\tilde{x}$ then also uniquely solves (P_0); see [111, Remark 4.6] for details. Moreover, one can also give NSP-based error bounds for recovery in the denoising case, see, e.g., [64, 165].

Remark 2.35. The NSP-based characterization of ℓ_0-ℓ_1-equivalence implies that if $\underline{\alpha}_k \geq 1/2$, ($P_0$) might still have a k-sparse and unique optimal solution, but such a solution cannot always be recovered by Basis Pursuit (in particular, $k < \operatorname{spark}(A)/2$ does not imply, but is necessary for, $\underline{\alpha}_k < 1/2$). On the other hand, if we focus on a specific index set S instead of all sets of a given cardinality k, then the corresponding NSP variant "$\|x_S\|_1 < \|x_{S^c}\|_1$ for all $0 \neq x \in \mathcal{N}(A)$" gives an *individual* recovery guarantee for all $\hat{x}$ with $\operatorname{supp}(\hat{x}) = S$ as unique solutions of BP instances with $b = A\hat{x}$, cf. [111, Theorem 4.4].

Thus, for sufficiently small k, the order-k NSP provides a both necessary and sufficient SRC (for uniform k-sparse recovery), whereas those based on the ERC, mutual coherence or RIP are only sufficient. The possible discrepancy in informative value of the corresponding SRCs is illustrated by the following toy example.

Example 2.36. Consider the 7×8 full-rank matrix

$$A \coloneqq \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Clearly, $\mu(A) = 1/2 = \underline{\delta}_2$ and in fact, $\underline{\delta}_3 > 1$ already. Thus, neither the mutual coherence nor the RIP guarantee recovery of any k-sparse vectors beyond the trivial case $k = 1$. However, it is easy to check that $\operatorname{spark}(A) = 6$, so we know that any 2-sparse $\hat{x}$ is the unique solution to (P_0) with the above A and $b \coloneqq A\hat{x}$. Moreover, it is easy to see that all nullspace vectors of A have nonzero entries with identical absolute values (indeed, here, $\mathcal{N}(A) = \operatorname{span}\{(0, -1, 1, -1, 0, 1, -1, 1)^\top\}$). Consequently, the NSP holds with $\underline{\alpha}_k < 1/2$ for all $1 \leq k \leq 2 < \operatorname{spark}(A)/2 = 3$. In particular, this shows that whenever an instance of (P_0) with this A has a unique sparsest solution

with no more than 2 nonzero entries, it can be recovered by (P_1). Note also that the ERC (2.3) does not identify the recoverability of all 2-sparse vectors; for instance, $\mathrm{erc}(A, S) = 1$ for $S = \{1, 2\}$.

Moreover, the fact that $\|x^*\|_0 < \mathrm{spark}(A)/2$ is not necessary for some x^* to be the unique sparsest solution of $Ax = b := Ax^*$, cf. Remark 2.3, is further illustrated by, for instance, $x^* = (1, 1, 1, 0, 0, 0, 0, 0)^\top$ (which gives $b = (1/\sqrt{2})(3, 1, 0, 1, 1, 0, 0)^\top$). In fact, none of the discussed SRCs (except for the "individual NSP" variant from Remark 2.35) indicate that this particular x^* could be recovered by either (P_0) or (P_1); however, x^* is indeed both the unique sparsest solution (which can easily be verified directly) and the unique ℓ_1-minimizer for A and the above b (by strict complementarity w.r.t. the corresponding dual optimal solution $(\sqrt{2}, 0, 0, 0, 0, 0, 0)^\top$; see also Remark 3.3 later on). Similarly, for $S = \{2, 3, 4\}$, even the individual NSP from Remark 2.35 does not hold, but with $x^* = (0, 1, 1, 1, 0, 0, 0, 0)^\top$ we again find a unique optimal solution of the respective (P_0) and (P_1) instances consisting of A and $b := Ax^*$.

Remark 2.37. The NSP has been stated in several different, but equivalent, forms, see, e.g., [266, 77, 64], or (2.18) below. Moreover, it is known that the NSP-based SRC $\underline{\alpha}_k < 1/2$ is equivalent to the necessary and sufficient recovery conditions described in terms of *s-goodness* (or *strict s-balancedness of* $\mathcal{N}(A)$) [145], with $s \equiv k$, or the *(k-uniform) Range Space Property* for $A^\top$ [267] (see also [114, 125, 265, 264, 145], and Remark 3.3 on page 60), and is actually sometimes called nullspace property itself (e.g., in [108]).

2.5.1 Complexity of Computing the Nullspace Constant

The computation of $\underline{\alpha}_k$ has also long been suspected to be NP-hard, and several heuristics have been developed to compute good bounds on $\underline{\alpha}_k$, e.g., the semidefinite programming approaches in [76, 77] or an LP-based relaxation in [145]. Note also that, e.g., if the order-$2k$ RIC of A satisfies $\underline{\delta}_{2k} < 1/3$ then the NSP of order k holds with $\underline{\alpha}_k < 1/2$ (and every k-sparse vector is the unique solution to both (P_0) and (P_1)), see [108, Theorem 2.6]. Clearly, this could in turn be combined with incoherence-based RIP bounds, cf. the beginning of Section 2.4.

Regarding exact algorithms to determine the NSC $\underline{\alpha}_k$, the only work we are aware of is the very recent [61], in which a lower/upper-bound-"sandwiching" procedure is proposed and empirically demonstrated (but not generally guaranteed) to be faster than brute force. In fact, no general exact method besides exhaustive search appears to be known. However, it seems that no rigorous proof of (NP-)hardness had been

given either, prior to our work: The following results show that computing $\underline{\alpha}_k$ is indeed a challenging problem.

Theorem 2.38. *Given a matrix $A \in \mathbb{Q}^{m\times n}$ and a positive integer k, the problem to decide whether A satisfies the NSP of order k with some constant $\alpha_k < 1$ is coNP-complete.*

Proof. First, note that the NSP (2.17) is equivalent to the condition that

$$\|x_S\|_1 \;\leq\; \alpha_k \|x\|_1 \tag{2.18}$$

holds for all $S \subseteq [n]$ with $|S| \leq k$, and all $x \in \mathbb{R}^n$ with $Ax = 0$. Clearly, (2.17) and (2.18) are always satisfied for some $\alpha_k \in [0, 1]$.

We first prove that the considered decision problem is in coNP. To certify the "no" answer, it suffices to consider a vector $\tilde{x} \neq 0$ with $A\tilde{x} = 0$ and a nonempty set $S \subseteq [n]$ with $|S| \leq k$ that tightly satisfy (2.18) for $\alpha_k = 1$. This implies that S contains the support of $\tilde{x}$. Thus, $1 \leq \|\tilde{x}\|_0 \leq k$. Clearly, since $\tilde{x}$ is contained in the nullspace $\mathcal{N}(A)$ of A, it can be assumed to be rational with encoding length polynomially bounded by that of A. This shows that the "no" answer can be certified in polynomial time.

To show hardness, we claim that the matrix A has a circuit of size at most k (i.e., $\operatorname{spark}(A) \leq k$) if and only if there does *not* exist any $\alpha_k < 1$ such that (2.17) holds. Since the former problem is NP-complete by Theorem 2.5, this completes the proof.

Assume A has a circuit of size at most k. Then there exists a vector $x \in \mathcal{N}(A)$ with $1 \leq \|x\|_0 \leq k$. It follows that $\|x\|_{k,1} = \|x\|_1$. Thus, validity of (2.17) necessitates $\alpha_k \geq 1$. (Since, trivially, $\alpha_k \leq 1$ is always possible, we could take $\alpha_k = 1$.)

Conversely, assume that there exists no $\alpha_k < 1$ such that (2.17) holds for A and k. This implies that there is a vector x with $Ax = 0$ such that $\|x\|_{k,1} = \|x\|_1$ and $1 \leq \|x\|_0 \leq k$, because otherwise, $\alpha_k < 1$ would be possible. But this means that the support of x contains a circuit of A of size at most k, which shows the claim. □

We immediately obtain the following.

Corollary 2.39. *Given a matrix $A \in \mathbb{Q}^{m\times n}$ and a positive integer k, it is NP-hard to compute the nullspace constant $\underline{\alpha}_k$.*

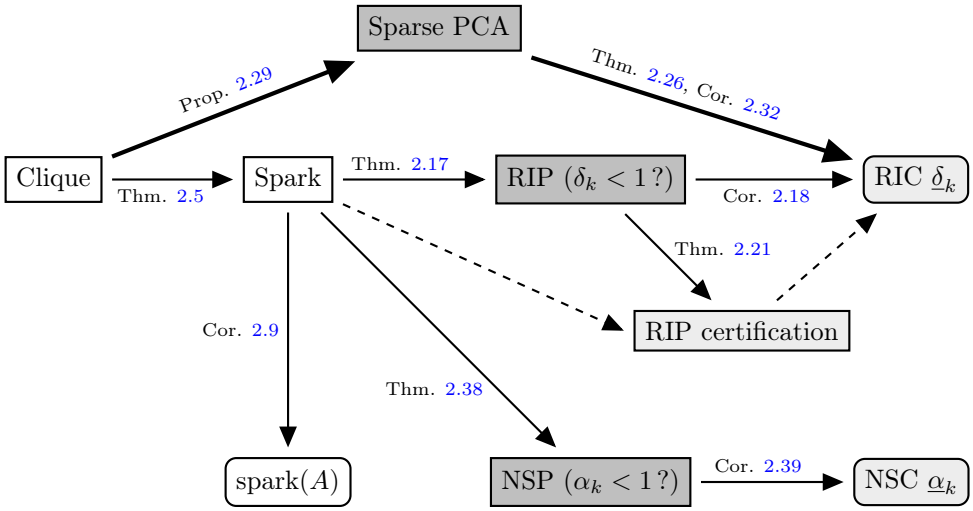


Figure 2.1. Schematic overview of the main SRC hardness results. Decision problems are set in rectangular boxes, computation problems in boxes with rounded corners; white background signals NP-hardness (for decision problems: NP-completeness), dark gray indicates coNP-completeness, and light gray means NP-hardness by reduction from a coNP-complete problem.

2.6 Summary

We conclude this chapter with two overviews:

Firstly, Figure 2.1 summarizes the most important NP-hardness results we obtained in the preceding sections. The arrows in the depicted diagram indicate the polynomial reductions employed in the proofs of the corresponding results (specified in the respective labels). A dashed line indicates an alternative way of proof; thick lines correspond to strong NP-hardness proofs (regular solid lines pertain to weak NP-hardness). Recall that the Clique Problem is NP-complete in the strong sense.

Moreover, Figure 2.2 illustrates the implications of the discussed sparse recovery conditions and the connections between them. Here, we focus entirely on Basis Pursuit and when it can be used to solve the actual (noise-free) sparse reconstruction problem (P_0). (For completeness, we also included the characterization of BP solution uniqueness known as "strong source condition" which incorporates not only S but also the sign pattern on S; it will be made precise in Remark 3.3 in the next chapter.)

The number k always refers to the ℓ_0-norm and the set S designates the support

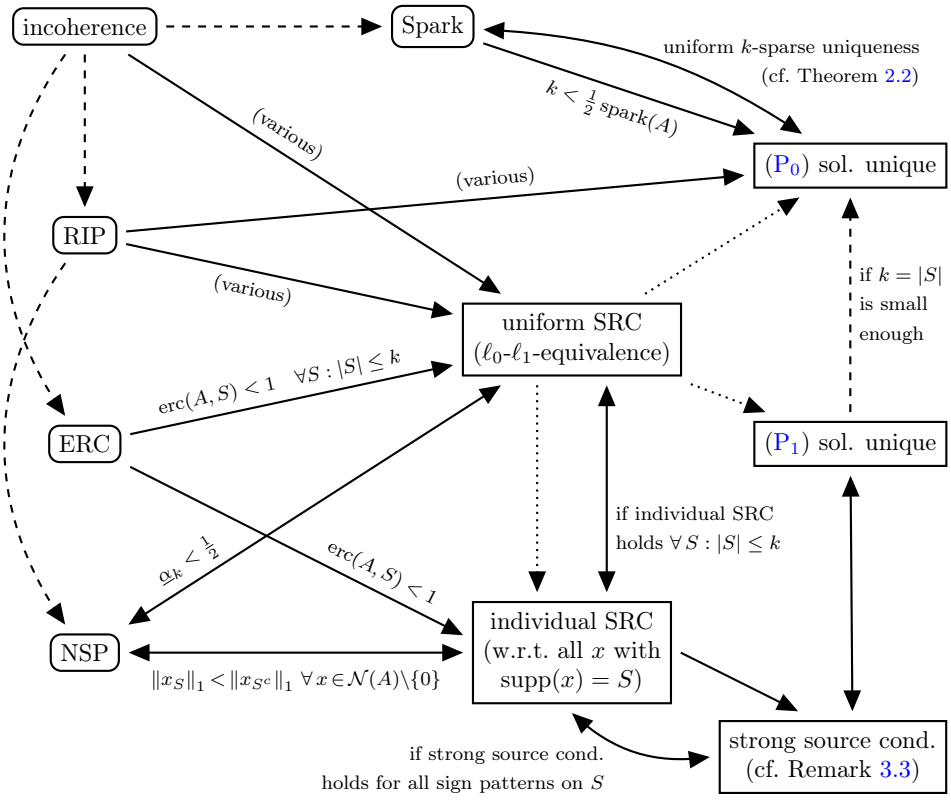


Figure 2.2. Illustration of relations between matrix properties and associated SRCs, and their implications w.r.t. (P$_0$) and (P$_1$). Uniform SRCs yield ℓ_0-ℓ_1-equivalence for all k-sparse vectors; individual recovery conditions pertain to uniqueness of BP solutions with support S.

of a solution in question. The dashed arrows indicate that one property implies a certain manifestation of another (such as, e.g., $\mathrm{spark}(A) \le 1 + 1/\mu(A)$); we only show the relations mentioned in this chapter, so a missing link does not necessarily mean that no such connection exists. Solid arrows mark implications—e.g., that the RIP provides various sufficient conditions for uniform sparse recovery or uniqueness of the sparsest representation—or equivalences like the NSP-based characterization of uniform k-sparse recovery via the SRC $\underline{\alpha}_k < 1/2$. The dotted arrows are natural relations stemming, essentially, from the definition of ℓ_0-ℓ_1-equivalence.

Let us emphasize that Figure 2.2 does not give a complete picture of how sparse solutions to underdetermined linear systems can be efficiently obtained—other ap-

proaches than BP and the case of approximately sparse solutions are not represented. However, the strongest known recovery results pertain to (P_1); moreover, Basis Pursuit will be the central problem in the next chapter.

CHAPTER 3

Solving Basis Pursuit

Due to its importance in Compressed Sensing, a broad variety of solution algorithms has been developed for the Basis Pursuit (BP) problem

$$\min \|x\|_1 \quad \text{s.t.} \quad Ax = b, \tag{P_1}$$

for $A \in \mathbb{R}^{m \times n}$ with $\operatorname{rank}(A) = m < n$ and $0 \neq b \in \mathbb{R}^m$. (We make these standard assumptions because, if $\operatorname{rank}(A) < m$, $Ax = b$ might not have a solution at all, but if a solution of $Ax = b$ does exist, then $\operatorname{rank}(A) = m \leq n$ can be assumed without loss of generality—otherwise, at least one row is redundant and can be omitted—and $m < n$ makes the problem nontrivial by allowing different representations of b by linear combinations of columns of A. Moreover, for $b = 0$, the trivial optimal solution of (P_1), and also of (P_0), is the all-zero vector.) In this chapter, we concentrate on the practical solution of (P_1) and mostly neglect the intricate connections to the sparse recovery problem (P_0) that explain the relevance of BP in Compressed Sensing.

An important practical concern is the typical slow (local) convergence of iterative solvers. In Section 3.1, we develop a heuristic method that, if applied repeatedly during an algorithm, often allows for "jumping" to an optimal point of (P_1) and hence for early termination. This heuristic optimality check (HOC) is based on the idea that the iterates allow to approximate the support of an optimal solution; using the support, we try to construct a feasible primal-dual pair. We will show that under certain conditions the constructed solutions are indeed optimal. It is noteworthy that HOC is related to, but different from, the idea of support estimation, which has been used for a penalized version of (P_1) to improve solution quality [107] or to speed up the algorithm [253] and in a modification of Basis Pursuit to enhance the sparse recovery properties (i.e., with respect to (P_0)) [251]. HOC is also similar to

the "debiasing" postprocessing routine from [256], but differs from it due to using approximate supports and, more importantly, the additional construction of a dual certificate to prove optimality.

A large part of this Chapter (Sections 3.2–3.4) is comprised of an extensive computational comparison of solvers for (P_1), including the proposal of a test set containing 100 dense and sparse matrices A and 548 right hand sides b. In general, such a comparison is complicated by several facts. The most important one is that the termination requirements for each solver may be different, resulting in final solution points that vary highly in their distance to feasibility and optimality. Of course, there is a trade-off between feasibility/optimality requirements and performance. We cope with this problem by designing the test set in such a way that the optimal solutions are unique. The results of the solvers are then compared both with respect to the running time as well as the distance to the optimal point, which allows to estimate both the distance to feasibility and optimality at the same time. Earlier comparisons of solvers for ℓ_1-related minimization problems can be found, e.g., in [245, 19, 258, 173].

Another complication is the fact that there is an abundance of possible data and solvers to choose from. As a consequence, we had to limit the number of tested solvers and the size of the test set. Concerning the solvers, we decided to only consider algorithms that *provably* solve (P_1) and for which implementations are publicly available. We are aware of the following six solvers that meet these requirements: ℓ_1-Homotopy, SPGL1, ℓ_1-Magic, SolveBP/PDCO, YALL1, and our own contribution, ISAL1 (which is explained in detail in Section 4.6); additionally, we include the commercial LP solver Cplex and the noncommercial LP solver SoPlex in our experiments. All these solvers are briefly reviewed in Section 3.2. Concerning the test set, we deliberately decided to only consider matrices with an explicit representation. Thus, we do not use the ability of several solvers to work with implicit matrices by callback functions that perform matrix-vector products. The test set should also be seen independently of ℓ_0-ℓ_1-equivalence, although we also use tools from Compressed Sensing to guarantee that the optimal solutions are unique, see Section 3.3.

The results of our computational solver comparison are presented in Section 3.4. We investigate the performance and quality of solutions generated by the eight ℓ_1-solvers mentioned above. We also investigate the impact of HOC on six of these solvers. It turns out that Cplex and SoPlex (using the dual simplex algorithm) perform remarkably well—this refutes the rumor that standard methods for LP solving perform badly for ℓ_1-minimization. SPGL1, ℓ_1-Homotopy, SolveBP/PDCO and ISAL1 usually produce acceptable solutions within reasonable time. Moreover, using HOC for early termination significantly improves the solution quality and run-

ning time of SPGL1, ℓ_1-Homotopy, ISAL1, SolveBP/PDCO and ℓ_1-Magic; YALL1 does not seem to benefit from HOC.

Our computational comparison can be seen as a step towards a common empirical evaluation of the many different algorithms proposed during the last years. This can help to select the right algorithm for different (practical) purposes, to further improve the available implementations, and possibly to create new algorithmic ideas.

Up to Section 3.4, the contents of this chapter are essentially a revised version of the joint work [174] with Dirk Lorenz and Marc Pfetsch. Additionally, in Section 3.5, we present a polynomial reduction from an arbitrary standard-form linear program to (the dual of) (P_1). While it is well-known that (P_1) can easily be written as an LP, the reverse direction is less obvious, but interesting as well: It shows that theoretically, every linear program can be solved as a Basis Pursuit problem. Moreover, while the focus of this chapter clearly lies on (P_1), we will briefly turn to the BP Denoising problem (P_1^δ) and its regularization variant (QP_λ) in the final Section 3.6. There, we show how the HOC idea can be applied to these problems and present a few numerical results to assess its potential in the respective settings.

The test set, implementations of ISAL1 and HOC, as well as comprehensive computational results can be downloaded from `wwwopt.mathematik.tu-darmstadt.de/spear`.

3.1 Heuristic Optimality Check

Many algorithms for (P_1) produce an (infinite) sequence of points converging to an optimal point. Typically, (practical) convergence becomes slow towards the end, i.e., there is a trade-off between solution speed and quality. In this section, we propose a general method to heuristically compute an optimal point from a given iterate. Applied repeatedly within an ℓ_1-solver, this routine often allows for early termination.

3.1.1 Theoretical Foundation

The approach is based on the following characterization of optimal solutions of (P_1):

Lemma 3.1. *A vector $x^* \in X := \{ x : Ax = b \}$ is optimal for the Basis Pursuit problem (P_1) if and only if there exists $w \in \mathbb{R}^m$ such that $A^\top w \in \partial \|x^*\|_1$.*

Proof. The normal cone of X is easily seen (e.g., using [221, Lemma 2.38]) to be

$$N_X = \{\, z \in \mathbb{R}^n \,:\, \exists\, w \in \mathbb{R}^m :\; z = A^\top w \,\} = \mathcal{R}(A^\top)$$

independently of the actual point (i.e., $N_X(x) = N_X(y) = N_X$ for all $x, y \in X$). Since $N_X = -N_X$, the claimed optimality condition can be rewritten as $-\partial\|x^*\|_1 \cap N_X \neq \emptyset$. The result now follows from Lemma 1.4. □

Remark 3.2. The subdifferential of the ℓ_1-norm at a point $x \in \mathbb{R}^n$ is given by

$$\partial\|x\|_1 = \{\, h \in [-1,1]^n \,:\, h_i = \operatorname{sign}(x_i) = x_i/|x_i| \text{ for all } i \text{ with } x_i \neq 0 \,\}\,.$$

Remark 3.3. The result in Lemma 3.1 is by no means new; for instance, it is also derived in the proof of [114, Theorem 4] via linear programming duality. Moreover, if and only if $\operatorname{rank}(A_{\operatorname{supp}(x^*)}) = \|x^*\|_0$ and w (with $A^\top w \in \partial\|x^*\|_1$) can be chosen such that $|(A^\top w)_j| < 1$ for all $j \notin \operatorname{supp}(x^*)$, then x^* is the *unique* optimal solution of (P_1); this extended condition is known (among other names) as the *strong source condition* [125]. (The ERC, see Section 2.2, yields BP solution uniqueness because it implies this condition with the specific choice of w as the least-squares solution to $(A_{\operatorname{supp}(x^*)})^\top w = \operatorname{sign}(x^*_{\operatorname{supp}(x^*)})$.) In fact, requiring the strong source condition to hold for *all* k-sparse vectors x (and a given A) is equivalent to asking that A satisfies the nullspace property of order k with NSC $\underline{\alpha}_k < 1/2$ (cf. Section 2.5), see [145, 267] for proofs. Furthermore, the strong source condition can be extended to necessary and sufficient criteria for uniqueness of optimal solutions for (P_1^δ), (QP_λ) and the respective "ℓ_1-analysis" problems (in which $\|Bx\|_1$ with some matrix B replaces $\|x\|_1$ in the objectives), as well as for (LS_τ), see [265, 264].

A consequence of Lemma 3.1 is that, for a given optimal solution x^* of (P_1), every other vector $x \in X$ with the same sign pattern as x^* must also be optimal, since then $\partial\|x^*\|_1 = \partial\|x\|_1$. (Consequently, multiple optimal solutions can only exist if X is parallel to a nontrivial non-vertex face of the ℓ_1-norm ball with radius equal to the optimal objective; all points on a face share the same sign pattern, with possible degeneracy.) Thus, if the iterates x^k of a solution algorithm converge to an optimal solution x^* with support S^*, we expect that $\operatorname{sign}(x^k_{S^*}) = \operatorname{sign}(x^*_{S^*})$ for sufficiently large k; however, x^k may still have many more nonzeros than x^*—although not necessarily, as S^* could also be approached by inner approximations $\operatorname{supp}(x^k) \subset S^*$—and can also be infeasible. Now, the idea is to try to identify S^* from x^k, construct a *feasible* solution $\hat{x}$ with support S^*, and prove its optimality (via Lemma 3.1) by constructing $\hat{w} \in \mathbb{R}^m$ with $A^\top \hat{w} \in \partial\|\hat{x}\|_1$ or, equivalently, $A_{S^*}^\top \hat{w} = \operatorname{sign}(\hat{x}_{S^*})$ and $-\mathbb{1} \leq A^\top \hat{w} \leq \mathbb{1}$. To simplify the computations, we only solve the equation system and then verify whether $\hat{w}$ obeys the box constraint as well.

Algorithm 3.1 Exact Optimality Check (EOC) for (P_1)

Input: matrix A, right hand side vector $b \neq 0$, vector x

1: deduce candidate (approx.) support S from x
2: **if** $\hat{x}$ exists with $A_S\,\hat{x}_S = b$ and $\hat{x}_j = 0\ \forall\, j \notin S$ **then**
3: compute solution $\hat{w}$ to $A_S^\top w = \operatorname{sign}(\hat{x}_S)$
4: **if** $\|A^\top \hat{w}\|_\infty = 1$ **then**
5: return "success"

Note that if x^* is the unique optimum, we have $|S^*| \leq m$ and A_{S^*} has full rank. Thus, it suffices to correctly anticipate S^* and solve $A_{S^*}\hat{x}_{S^*} = b$, $\hat{x}_i = 0$ for all $i \notin S^*$. It then follows that $\hat{x} = x^*$ and, in particular, $\operatorname{sign}(\hat{x}_{S^*}) = \operatorname{sign}(x^*_{S^*})$. If x^* is not unique, or if the above reasoning is applied not to S^* but to some arbitrary $S \supset S^*$ inducing a full-rank submatrix A_S, the hope is that $\operatorname{sign}(\hat{x}_{S^*}) = \operatorname{sign}(x^*_{S^*})$ will nevertheless hold and optimality can still be proven. This yields the *Exact Optimality Check (EOC)* summarized in Algorithm 3.1.

Theorem 3.4. *If Algorithm 3.1 is successful then $\hat{x}$ is an optimal solution for (P_1).*

Proof. If the point $\hat{x}$ exists, it is obviously feasible. If $\hat{w}$ satisfies $A_S^\top \hat{w} = \operatorname{sign}(\hat{x}_S)$ and $(A^\top \hat{w})_j \in [-1, 1]$ for all $j \notin S$ (i.e., j such that $\hat{x}_j = 0$), we have $A^\top \hat{w} \in \partial\|\hat{x}\|_1$. For Step 4, note that since $b \neq 0$, $\|h\|_\infty = 1$ for every $h \in \partial\|x\|_1$ of any feasible point x. The result hence follows from Lemma 3.1. □

There are different ways to select the candidate support S in Step 1 of Algorithm 3.1, the easiest being a hard-thresholding operation:

$$S^k_\delta := \{\, j \in [n] \,:\, |x^k_j| > \delta \,\} \tag{3.1}$$

for some $\delta \geq 0$. An alternative is based on ℓ_1-norm concentration: For a fixed number $c \in (0, 1]$, choose S as

$$S^k_c := \arg\min \left\{ |J| \,:\, J \subseteq [n],\ \sum_{j \in J} |x^k_j| \geq c\,\|x^k\|_1 \right\}. \tag{3.2}$$

Of course, (3.1) and (3.2) are in a sense equivalent: For each δ there exists a c such that the two sets S^k_δ and S^k_c coincide, and vice versa. However, a straightforward implementation of (3.2) requires sorting the $|x^k_j|$'s, yielding an $\mathcal{O}(n \log n)$ running time, whereas (3.1) can be implemented in $\mathcal{O}(n)$. On the other hand, (3.1) will only become effective when the entries outside of the optimal support are small enough already and if δ is not too large (to avoid cutting off entries in the optimal support). In contrast, (3.2) is less dependent on the magnitudes of optimal nonzero entries.

Algorithm 3.2 HEURISTIC OPTIMALITY CHECK (HOC) for (P_1)

Input: matrix A, right hand side vector $b \neq 0$, vector x

1: deduce candidate (approx.) support S from x
2: compute approximate solution $\hat{w}$ to $A_S^\top w = \operatorname{sign}(x_S)$
3: **if** $\|A^\top \hat{w}\|_\infty \approx 1$ **then**
4: **if** $\hat{x}$ exists with $A_S \hat{x}_S \approx b$ and $\hat{x}_j = 0 \; \forall j \notin S$ **then**
5: **if** $(\|\hat{x}\|_1 + b^\top(-\hat{w}))/\|\hat{x}\|_1 \approx 0$ **then**
6: return "success"

3.1.2 Practical Considerations

EOC possibly involves a high computational cost when used as a frequently evaluated stopping criterion in an ℓ_1-solver. We now describe how it can be turned into an efficient and practically useful device, the *Heuristic Optimality Check (HOC)*. The result is Algorithm 3.2, which differs from Algorithm 3.1 in the following aspects.

In Step 2 of Algorithm 3.2, we wish to obtain a solution to $A_S^\top w = \operatorname{sign}(x_S)$ (if possible). To that end, we use the pseudo-inverse $(A_S^\top)^\dagger = A_S(A_S^\top A_S)^{-1}$ to calculate $\hat{w} = (A_S^\top)^\dagger \operatorname{sign}(x_S)$. In fact, we can avoid the explicit calculation of $(A_S^\top)^\dagger$ by obtaining a solution $\hat{v}$ to $A_S^\top A_S \, v = \operatorname{sign}(x_S)$ via the method of conjugate gradients (CG) [133] (cf. Section 1.3; see [197] for a convergence argument in case A_S is rank-deficient) and then taking $\hat{w} = A_S \hat{v}$.

We observed that if S equals the optimal support S^* (or a superset of S^* that induces a full-rank submatrix A_S), then approximating $\hat{v}$ by computing only a few CG iterations is enough to identify $\hat{w}$ with $\|A^\top \hat{w}\|_\infty \approx 1$ (here, as well as in all other comparisons, we used a tolerance value of 10^{-6}). Hence, the computational overhead for HOC in case the support approximations are incorrect can be reduced by limiting the number of CG iterations (we use at most 20). For this reason we first compute $\hat{w}$ and then $\hat{x}$.

The next step is to check whether indeed $\|A^\top \hat{w}\|_\infty \approx 1$. If this is the case, we try to compute $\hat{x}$ with $\hat{x}_i = 0$ for $i \notin S$ and $A_S \hat{x}_S = b$. If $\hat{x}$ exists, $\operatorname{sign}(\hat{x}_S)$ might differ from $\operatorname{sign}(x_S)$ (in particular, $\hat{x}_S$ may contain zeros where x_S does not), and we cannot directly apply Lemma 3.1. Instead, we use strong duality, i.e., (cf. Lemma 1.5)

$$\min\{\, \|x\|_1 \,:\, Ax = b \,\} \;=\; \max\{\, -b^\top y \,:\, \|A^\top y\|_\infty \leq 1 \,\},$$

and conclude that $(\hat{x}, -\hat{w})$ forms an optimal primal-dual pair if the duality gap $\|\hat{x}\|_1 - b^\top \hat{w}$ is (approximately) 0. The empirical results suggest that this optimality check is robust: In our computational experiments, if HOC was successful, it always

obtained the (unique) optimal point (i.e., it made no false-positive "success" claims).

Let us now give a few remarks on the integration of HOC into ℓ_1-solvers. First, the solver type should be taken into account when choosing a support approximation scheme: In methods usually needing many (cheap) iterations, (3.2) offers a more dynamic scheme, which—albeit being more expensive—may recognize the optimal support far earlier than the hard thresholding scheme (3.1). Examples where (3.2) proved very effective are ISAL1 and SPGL1, see Sections 4.6 and 3.2.5, respectively. However, SPGL1 maintains a different type of approximate support anyway (called "active set"), so suitable support approximations are immediately available and were found to work just as well.

On the other hand, for algorithms with typically only relatively few (but expensive) iterations, (3.1) may be more adequate. An example where HOC using this variant improved performance is the interior-point method ℓ_1-Magic, see Section 3.2.3. In this case, we applied (3.1) with a different δ in each iteration, namely, $\delta^k := \|x^k\|_1/10^6$. This scheme was also used (at constant iteration intervals) in YALL1. Finally, in ℓ_1-Homotopy, we used the inherent support build by the algorithm and additional hard-thresholding (3.1) with $\delta = 10^{-9}$.

To limit the overall computational effort in either case, we only execute HOC if the (approximate) support S has changed. Since there always exists an optimal solution with at most m nonzeros, we do not run HOC as long as $|S| > m$ (naturally, if a priori sparsity bounds were known, these could be used instead of m). Moreover, we run HOC only every R iterations. The choice of this HOC frequency R is nontrivial: Lower values for R increase the overhead induced by HOC, but may also lead to early termination if HOC is successful. Thus, one can clearly expect a certain trade-off between the overhead by running HOC and the speed-up it (hopefully) yields. The goal should thus be to try to maximize HOC efficiency while at the same time keeping the overhead low for at least those instances where HOC is not successful.

Since in second-order methods, iterations are relatively expensive already and change the iterate point quite significantly, it makes sense to try HOC in every iteration, i.e., to set $R = 1$. Thus, we did so for ℓ_1-Magic and SolveBP/PDCO. For the first-order solvers, we benchmarked R by choosing $\lfloor \alpha\, m \rfloor$ (the nearest integer no larger than $\alpha\, m$), with $\alpha \in \{\frac{1}{1000}, \frac{1}{500}, \frac{1}{200}, \frac{1}{100}, \frac{1}{50}, \frac{1}{20}, \frac{1}{10}\}$, which led to the best balance between speed-up and accuracy improvement (over our whole test set), respectively. (We used fractions of the row number m for these tests since this is the bounding dimension of the systems solved within HOC.) Thus, we set R to $\lfloor m/500 \rfloor$ in ℓ_1-Homotopy, to $\lfloor m/100 \rfloor$ in SPGL1 and ISAL1, and to $\lfloor m/10 \rfloor$ in YALL1.

Note that for YALL1, we actually did not rigorously benchmark R, because the HOC success rate is lower. Instead, we tried the same value that seemed best for the other first-order methods with a typically high number of iterations (i.e., SPGL1

and ISAL1), but found that very comparable results could be achieved using the value that leads to the least-possible overhead among our choices of R. Thus, the latter seemed preferable. Moreover, the choices were not entirely unambiguous: For SPGL1, with $R = \lfloor m/500 \rfloor$, HOC success occurred slightly more often, but the mean running times were unsatisfactory. Similarly, ISAL1 with $R = \lfloor m/200 \rfloor$ solved more instances but the running time improvement was notably less.

Remark 3.5. It should be noted that HOC can also be applied if A is only available as an implicit operator, as is often the case in large-scale practical applications. Then, it may require considerable effort to extract columns indexed by some $S \subset [n]$ to form A_S explicitly. However, this can be avoided by instead computing matrix-vector products involving A_S using A and an appropriate projection or embedding of the respective vector: For $x \in \mathbb{R}^n$, $A_S x_S = A\,\mathcal{P}_S(x)$, where $\mathcal{P}_S(x)$ is the projection onto the set of vectors in $\mathbb{R}^n$ whose support is a subset of, or equal to, S (i.e., the projection operation keeps the entries of x indexed by S and sets all others to zero); for $z \in \mathbb{R}^{|S|}$, $A_S z = A\,\mathcal{E}_S(z)$, where $\mathcal{E}_S(z) \in \mathbb{R}^n$ denotes the straightforward embedding of z into the aforementioned set in $\mathbb{R}^n$ (i.e., $(\mathcal{E}_S(z))_S = z$ and $(\mathcal{E}_S(z))_{[n]\setminus S} = 0$). Clearly, products involving $A_S^\top$ can be rewritten similarly as $A_S^\top y = (A^\top y)_S$ and, in particular, the CG method can still be utilized to solve the occurring equation systems.

3.1.3 HOC Success Guarantees

The Basis Pursuit problem (P_1) became important for CS in conjuncture with various conditions that ensure ℓ_0-ℓ_1-equivalence, cf. Chapters 1 and 2. Thus, a question of particular interest regarding HOC is whether it is possible to guarantee that it will work correctly under such sparse recovery conditions (provided the optimal support was approximated sufficiently well). As we show below for the SRCs based on the ERC and the mutual coherence (see Sections 2.2 and 2.1, respectively), this is indeed sometimes possible.

Theorem 3.6. *Let the support S^* of an optimal solution x^* of (P_1) obey the ERC (2.3). Then $w^* = -(A_{S^*}^\top)^\dagger \operatorname{sign}(x^*_{S^*})$ is an optimal dual solution. Hence, in exact arithmetic, HOC applied to (A, b, x), using $\hat{w} = (A_S^\top)^\dagger \operatorname{sign}(x_S)$, returns "success" with $\hat{x} = x^*$, if $\operatorname{sign}(x_{S^*}) = \operatorname{sign}(x^*_{S^*})$ and either the support was correctly estimated ($S = S^*$), or S contains S^*, obeys the ERC, and A_S has full column rank.*

Proof. Let $S = S^*$ and $\operatorname{sign}(x_{S^*}) = \operatorname{sign}(x^*_{S^*})$. For an arbitrary $j \notin S^*$, we have

$$|A_j^\top (A_{S^*}^\top)^\dagger \operatorname{sign}(x^*_{S^*})| \;\le\; \|A_j^\top (A_{S^*}^\top)^\dagger\|_1 \, \|\operatorname{sign}(x^*_{S^*})\|_\infty \;\le\; \operatorname{erc}(A, S^*) \;<\; 1. \quad (3.3)$$

Hence, $w^* = -(A_{S^*}^\top)^\dagger \operatorname{sign}(x^*_{S^*}) = -\hat{w}$ fulfills the optimality condition in Lemma 3.1. Moreover, $\hat{x}$ is feasible by construction and hence is optimal. Finally, $\hat{x}$ equals x^* since the ERC guarantees uniqueness of the solutions.

Now assume $\operatorname{sign}(x_{S^*}) = \operatorname{sign}(x^*_{S^*})$, $S \supset S^*$ with $\operatorname{rank}(A_S) = |S| \le m$, and that S obeys the ERC. Then, analogously to (3.3), one can see that $\hat{w} = (A_S^\top)^\dagger \operatorname{sign}(x_S)$ yields the dual feasible solution $-\hat{w}$. Since A_S has full column rank, $\hat{x}$ is the unique solution to $A_S \hat{x}_S = b$ with zero entries at positions $j \notin S$. Then $\hat{x}$ must also have zeros at $j \in S \setminus S^*$ (since x^* with support $S^* \subset S$ is primal feasible). Hence, since S^* obeys the ERC, $\hat{x} = x^*$. Moreover, because a subgradient at $\hat{x}$ can assume any value in $[-1, 1]$ for components in which $\hat{x}$ is 0, $A^\top \hat{w} \in \partial \|\hat{x}\|_1$. Thus, as the duality gap vanishes ($\|\hat{x}\|_1 - b^\top \hat{w} = 0$), HOC returns "success". □

Moreover, from the observation that sufficient sparsity with respect to mutual coherence implies the ERC on all correspondingly small supports (see, e.g., [114, Theorem 3]) we can immediately deduce the following result; cf. (2.2).

Corollary 3.7. *Let $\mu(A)$ be the mutual coherence of A and let x^* be an optimal solution of (P_1) with $\|x^*\|_0 < \frac{1}{2}\big(1 + \frac{1}{\mu(A)}\big)$. If HOC is used with the correct support and exact calculations, then it returns "success" if and only if the outcome $\hat{x}$ equals x^*.*

Numerical results supporting the claimed usefulness of the HOC will be presented later, in Section 3.4, where we turn to the computational comparison of ℓ_1-solvers. (Moreover, extensions to the denoising problems (P_1^δ) and (QP_λ) are discussed in Section 3.6.) Our MATLAB code for HOC can be obtained from `http://wwwopt.mathematik.tu-darmstadt.de/spear`.

3.2 Algorithms for Exact ℓ_1-Minimization

Given the great variety of ℓ_1-solvers, we had to restrict our attention to a certain subset of these methods. The solvers were chosen according to the following guidelines.

- Only *exact general-purpose* solvers for ℓ_1-minimization (P_1) are considered, i.e., algorithms that are theoretically guaranteed to be capable of solving ar-

bitrary instances of (P_1) to optimality. This choice rules out probabilistic or heuristical methods. It also excludes, for example, Orthogonal Matching Pursuit (OMP), since it only solves (P_1) if the solution is sufficiently sparse (see, e.g., [83, 207, 241]) and hence is not a general-purpose ℓ_1-solver.

- We restrict ourselves to algorithms with readily available implementations (all but two in MATLAB), and made our own code ISAL1 publicly accessible as well.

As a consequence, we leave out, e.g., NESTA [199, 19]. Although in theory it can solve (P_1) exactly, the implementation (provided by the authors of [19] at `http://www-stat.stanford.edu/~candes/nesta/`) is designed for instances in which the matrix A is a partial isometry (i.e., $AA^\top = I$). To be fair, the NESTA implementation can handle general matrices, but then it (currently) requires a full singular value decomposition of A; this adds a considerable amount of computational effort to the method which renders it noncompetitive with the other considered ℓ_1-solvers for larger instances. Moreover, we did not include the "Bregman iteration" [262] (the available implementation uses an outdated subproblem solver) or the linearized Bregman iteration [42], since it solves a slightly different problem with an additional penalty term in the objective, see [261].

In the following, we provide an overview of the algorithms and their respective implementations that we included in the computational comparison.

3.2.1 ISAL1

ISAL1 is a specialization of the general subgradient method ISA (which we develop in Chapter 4) and is discussed in detail in Section 4.6. It employs dynamic step sizes and adaptive approximate (instead of the usual exact) projections onto the solution set of $Ax = b$ to speed up iterations. Preliminary numerical experiments in [175, Section 5.2] had shown that ISAL1 has the potential to be a successful and competitive algorithm for CS sparse recovery via Basis Pursuit. The source code is available at `http://wwwopt.mathematik.tu-darmstadt.de/spear` (below, we used version 0.91).

Remark 3.8. It is worth mentioning that we also performed some experiments with bundle methods (cf., e.g., [134]), which usually do not require extensive parameter tuning as simpler subgradient methods do (including ISAL1). More precisely, we implemented code for solving (P_1) in the bundle framework extension to the GNU Scientific Library [119, 139, 152] as well as the ConicBundle package [130]. Both frameworks are C++/C-based and designed for unconstrained nonsmooth convex optimization problems. Using a basis B, i.e., a subset of m column indices such that

the $(m \times m)$ matrix A_B has full rank m, we can rewrite (P_1) in such a form (see also Lemma 3.11 on page 98):

$$\min \|x_N\|_1 + \|A_B^{-1} A_N x_N - A_B^{-1} b\|_1.$$

In our implementations, a basis B was obtained greedily by traversing the column indices in increasing order and adding them to B if this increases the rank of the associated submatrix, until $|B| = m$. (While this approach always works, it is not clear if there is a more sensible way of choosing a basis to make the above reformulation in some sense advantageous. However, it should be noted that every optimal solution has a basis representation, due to the fact that (P_1) can be seen as a linear program.)

Despite many theoretical advantages of bundle methods over simple subgradient methods, and therefore quite surprisingly, our experiments with the bundle codes were very discouraging—often, the programs did not converge to a solution for hours, on instances that ISAL1 managed to solve within seconds. Thus, we refrained from further investigating bundle methods as means to tackle (P_1).

3.2.2 The Homotopy Method

The homotopy method [204, 181] (see also [258]) employs auxiliary problems of the form

$$\min \tfrac{1}{2}\|Ax - b\|_2^2 + \lambda \|x\|_1, \qquad (\mathrm{QP}_\lambda)$$

with a nonnegative parameter λ. More precisely, denoting by x_λ^* a solution of (QP_λ) for some $\lambda \geq 0$, it was shown that the solution path $\chi : \lambda \mapsto x_\lambda^*$ following the change in λ is piecewise linear [204]. Moreover, for $\lambda \geq \|A^\top b\|_\infty$, we have $x_\lambda^* = 0$, and for $\lambda \to 0$, x_λ^* converges to a solution of (P_1). The homotopy method starts with a large λ (and $x_\lambda^* = 0$) and then identifies the "breakpoints" of χ (which correspond to changes in the support of the x_λ^*'s). This yields a decreasing sequence of λ's so that x_λ^* approaches the optimum as $\lambda \to 0$.

The homotopy method provably solves (P_1), whereas its well-known variant LARS [98] is only a heuristic for (P_1), since it leaves out a critical algorithmic step (namely that indices may leave the support of x_λ^* again after they were added in some previous iteration). However, if the solution is sufficiently sparse—i.e., in a favorable situation from the Compressed Sensing perspective—both methods are equivalent (see [93]) and only need as many iterations as the (then unique) optimal solution of (P_1) has nonzero entries. (This is called the *k-step solution property*; other sufficient conditions besides a certain solution sparsity—for instance, as in (2.2) [93]—can

also ensure this property, see [95].) Note that in the worst-case, the number of iterations of the homotopy method is exponential in n (as for the simplex method from linear programming with virtually all deterministic pivot rules, see, e.g., [155]), reflecting the possible traversal of all sign-patterns [180].

We used the implementation ℓ_1-*Homotopy* from `http://users.ece.gatech.edu/~sasif/homotopy` (version 1.0); cf. [9].

3.2.3 ℓ_1-Magic

The ℓ_1-Magic algorithm is a specialization of the primal-dual interior-point method from [38] to a linear programming reformulation of (P_1). The LP considered here is

$$\min \mathbb{1}^\top u \quad \text{s.t.} \quad -u \le x \le u,\ Ax = b. \tag{3.4}$$

Implicitly, $u \ge 0$ holds, and the objective effectively minimizes the absolute values of x. Notably, and unlike typical interior-point methods known from linear (or convex) programming, ℓ_1-Magic solves intermediate equation systems with a conjugate gradient method, thereby avoiding some computationally expensive matrix inversions. A more detailed description is provided within the user's guide of the ℓ_1-Magic implementation; we used version 1.11, which can be found at `http://www.l1-magic.org`.

Remark 3.9. In fact, if ℓ_1-Magic is not supplied with functions returning matrix-vector products with A or $A^\top$ instead of A itself, the implementation does not use CG but MATLAB's own linear equation solver `linsolve`. In particular, specific options are set to specialize `linsolve` to symmetric positive definite matrices. However, preliminary tests showed that this sometimes leads to abnormal program abortion (if the s.p.d. requirement is violated for numerical reasons). Similar numerical trouble with this solver has been reported previously in [245]. Therefore, to ensure that ℓ_1-Magic runs until regular termination on all test instances, we disabled these special options. (Note that this does not change the underlying algorithm itself, only the way the implementation realizes it.) For sparse matrices A, where the method can profit from using only products with A or $A^\top$ (avoiding explicit construction of dense matrices of the form $A_S^\top A_S$, $S \in [n]$), we provided ℓ_1-Magic with corresponding matrix-vector product callback functions, so that CG is used. For dense A, the ℓ_1-Magic variant with CG was even slower than the one with disabled `linsolve` options.

3.2.4 SolveBP/PDCO

The SolveBP function comes as a part of SparseLab (http://sparselab.stanford.edu) and solves (P_1) by applying PDCO (http://www.stanford.edu/group/SOL/software/pdco.html), a primal-dual log-barrier method for convex objectives, to the equivalent linear program

$$\min \mathbb{1}^\top x^+ + \mathbb{1}^\top x^- \quad \text{s.t.} \quad Ax^+ - Ax^- = b,\ x^+ \geq 0,\ x^- \geq 0 \tag{3.5}$$

arising from the standard split of variables $x = x^+ - x^-$ with $x^+ := \max\{0, x\}$ and $x^- := \max\{0, -x\}$. Like ℓ_1-Magic, SolveBP only needs callback functions for multiplication with A and $A^\top$ to perform all necessary computations involving matrices. We used version 2.1 of SparseLab.

3.2.5 SPGL1

The SPGL1 algorithm, introduced in [245], solves (P_1) via a sequence of LASSO problems with suitably chosen parameters τ, i.e.,

$$\min \|Ax - b\|_2 \quad \text{s.t.} \quad \|x\|_1 \leq \tau. \tag{LS_τ}$$

More precisely, an inexact Newton-method is applied to find the value of τ for which the solutions of (LS_τ) and (P_1) coincide; the LASSO subproblems are solved efficiently by a specialization of the spectral projected gradient method (see [29]).

We used version 1.7 of SPGL1, available at http://www.cs.ubc.ca/labs/scl/spgl1, which was proven to quickly compute (or approximate) solutions to (P_1) in various experiments already, see, e.g., [19, 245, 259].

3.2.6 YALL1

The YALL1 software package (http://yall1.blogs.rice.edu) contains an implementation of an alternating direction method (ADM) tailored to (P_1). The paper [259] describes both a primal-based ADM, which corresponds to an inexact augmented Lagrangean algorithm for (P_1), and a dual-based ADM. Two variants of the latter algorithm are implemented in YALL1, with slightly different iterate updates for the cases $AA^\top = I$ and $AA^\top \neq I$. (While the implementation is arguably designed for the first case, the second one is handled without severely limiting steps

such as computing a full SVD as in the above-mentioned NESTA code.) We used version 1.3 of YALL1.

3.2.7 CPLEX

As already pointed out, (P_1) is essentially a linear program, see (3.4) and (3.5). Thus, as a reference, we also solve (P_1) with the dual simplex method of the commercial LP solver CPLEX [140] (version 12.4.0.1), applied to (3.5). (This form has fewer primal constraints—excluding variable bounds—and therefore fewer dual variables than (3.4). Simplex-based LP solvers are expected to be more efficient in this situation, cf., e.g., [143, Section 3.1]. We also experimented with the primal simplex and barrier algorithms, but found the dual simplex approach to perform better.)

Note that while CPLEX is accessed via a compiled C-library, all of the implementations listed above are MATLAB programs (although the projections onto the ℓ_1-ball in SPGL1 are performed by an external C-program, embedded in the MATLAB code via MEX-functions). Since (at user level) MATLAB is an interpreted language, there will be a certain bias to the advantage of CPLEX in terms of running times. On the other hand, we use a C-program that sets up the linear programs from the raw data files via the SCIP LP-interface (version 3.0.0, available within the SCIP optimization suite at `http://scip.zib.de`) and then calls the dual simplex algorithm of CPLEX. This LP construction step is included in the time measurements for CPLEX.

3.2.8 SoPlex

Because CPLEX is proprietary, it is also of interest to include a noncommercial LP solver. Since for our CPLEX wrapper code, we used the interface from SCIP, we consider the solver SoPlex (version 1.7.0, cf. [257]) that comes with the SCIP optimization suite. The C-program setting up the linear programs from data files is the same one we used for CPLEX; the LP construction is again included in the time measurements, and, as with CPLEX, we use the dual simplex method.

3.3 Test Set Description

To perform computational experiments, we built a test set consisting of 100 matrices A of different types (random or structured matrices and concatenations thereof)

Table 3.1. Matrix constructions and corresponding abbreviations.

name	dim.	matrix construction/type
BAND	$m \times m$	band matrix with bandwidth 5, entries drawn uniformly at random from $(0, 1)$; the upper right and lower left corner of the matrix are filled such that we obtain a circulant zero pattern
BIN	$m \times n$	binary matrix, entries drawn uniformly from $\{0, 1\}$
BINB	$m \times m$	binary full-rank square matrix, see BIN
BLROW	$m \times m$	block diagonal matrix with a full row-block at the bottom; the square diagonal blocks have dimension randomly chosen from $\{5, \ldots, 10\}$, the row-block spans 5 rows; the last diagonal block may thus be smaller than 5×5; entries are drawn uniformly at random from $(0, 1)$
CONV	$m \times m$	convolution matrix for a convolution with a Gaussian kernel with variance $1/2$ (truncated such that we obtain a banded matrix with bandwidth 7)
GAUSS	$m \times m$	entries are i.i.d. Gaussian $\mathcal{N}(0, 1)$
HAAR	$m \times m$	Haar matrix (requires $m = 2^r$, $r \in \mathbb{N}$)
HAD	$m \times m$	Hadamard matrix (requires $m = 2^r$, $r \in \mathbb{N}$)
ID	$m \times m$	identity matrix
INT	$m \times n$	integer matrix, entries drawn uniformly at random from $\{-10, -9, \ldots, 9, 10\}$
PHAD	$m \times n$	partial Hadamard matrix, consisting of m randomly chosen rows of the $n \times n$ HAD matrix
PRST	$m \times n$	partial real sinusoid transform matrix, consisting of m randomly chosen rows of the $n \times n$ RST matrix
ROB	$m \times m$	random orthonormal basis matrix
RSE	$m \times n$	random sign ensemble, entries are drawn from a Bernoulli ± 1 distribution
RST	$m \times m$	real sinusoid (Fourier) transform matrix
TER	$m \times n$	ternary matrix, entries are drawn uniformly from $\{-1, 0, +1\}$
URP	$m \times n$	uniform random projection matrix, generated by taking m random rows of an $n \times n$ random orthogonal matrix
USE	$m \times n$	uniform spherical ensemble, columns are m-vectors uniformly distributed on the sphere $\mathcal{S}^{m-1}$

and four or six right hand side vectors b per matrix. The dimensions and specific constructions of the matrices are summarized in Tables 3.1 and 3.2. After construction, the columns of each matrix were normalized to unit Euclidean length. Whenever a column appeared twice in a matrix, we randomly added single entries in one such column and renormalized, until all columns were unique.

Table 3.2. The $m \times n$ matrices in our test set: Matrices with $m \geq 2048$ are sparse, those with $m \leq 1024$ are dense (but may contain sparse subdictionaries); concatenations of matrix types are listed in square brackets in order of appearance.

m	n	matrix constructions
512	1024	BIN, INT, PHAD, PRST, RSE, TER, URP, USE, [HAAR,ID], [HAAR,RST], [HAD,ID], [ROB,RST]
	1536	BIN, INT, PHAD, PRST, RSE, TER, URP, USE, [BAND,GAUSS,RST], [BINB,ID,HAD], [HAAR,ID,RST], [HAAR,ROB,RST]
	2048	BIN, INT, PHAD, PRST, RSE, TER, URP, USE, [BAND,BINB,HAD,ID], [BAND,BLROW,HAD,ID], [BINB,CONV,HAAR,ROB], [HAAR,ID,ROB,RST]
	4096	[ID,HAAR,ROB,RST,BAND,BINB,BLROW,HAD]
1024	2048	as for 512×1024
	3072	as for 512×1536
	4096	as for 512×2048
	8192	as for 512×4096
2048	4096	[BINB,BLROW], [BINB,CONV], [CONV,HAAR], [HAAR,ID]
	6144	[BAND,BINB,HAAR], [BINB,HAAR,ID], [BLROW,CONV,ID], [CONV,HAAR,ID]
	8192	[BAND,BINB,CONV,ID], [BAND,BLROW,CONV,ID], [BINB,BLROW,HAAR,ID], [BINB,CONV,HAAR,ID]
	12288	[ID,HAAR,BAND,BINB,BLROW,CONV]
8192	16384	as for 2048×4096
	24576	as for 2048×6144
	32768	as for 2048×8192
	49152	[BAND,BINB,BLROW,CONV,HAAR,ID]

For each matrix A, we build two sets of right hand side vectors, from solution vectors with high dynamic range[3] and low dynamic range, respectively. For either dynamic range, the first two vectors b were each constructed using a vector x whose support satisfies the Exact Recovery Condition (ERC) [239] and thus is the unique optimal solution to the instance of (P_1) specified by matrix A and vector $b := Ax$ (cf. Section 2.2). More precisely, the supports of each such pair of vectors per matrix

[3]The dynamic range is the ratio of the largest and the smallest nonzero magnitudes of the vector entries.

are constructed as follows:

1. For each support size k (starting with $k = 1$), take a random k-element subset $S \subseteq [n]$, and check whether the ERC (2.3) holds, i.e., if

$$\operatorname{erc}(A, S) \;=\; \max_{j \notin S} \|(A_S^\top A_S)^{-1} A_S^\top A_j\|_1 \;=\; \max_{j \notin S} \|A_j^\top (A_S^\top)^\dagger\|_1 \;<\; 1.$$

   If for the current k a support S satisfying (2.3) was found, k is increased by 1 and the process is repeated. We stop as soon as 25 trials for some k failed.

2. For the second vector, we tried to enlarge supports by adding an index j^* to S that (for the current k) yields the maximum in $\operatorname{erc}(A, S)$: If the extended support $S \cup \{j^*\}$ also satisfies the ERC, we update S to $S \cup \{j^*\}$, increase k by 1, and repeat this procedure as long as possible. When eventually no further indices can be added without leading to a violation of the ERC, we continue our search with random supports of size $k = |S| + 1$. For a given matrix, we stop the search routine as soon as it failed for 25 random supports at some size k or if a time limit of one day was exceeded.

For these two supports per matrix A and dynamic range type, we then define x by fixing its nonzeros at the locations specified by the respective supports, with random signs and magnitudes. For the high dynamic range (HDR) test set, the magnitudes are 10^{5y} with the y's chosen uniformly at random from $(0, 1)$. For the low dynamic range (LDR), we use entries drawn uniformly at random from $(0, 1)$. (Note that this does not necessarily guarantee vectors with high or low dynamic range; however, it usually is the case: Over 80% of the HDR instance dynamic ranges exceed 10^4 and more than 95% of the LDR instance dynamic ranges are below 10^2.) All respective right hand sides are then obtained via $b := Ax$.

If the support of an optimal solution x^* satisfies the ERC, x^* is the unique optimum of both (P_1), and if $\|x^*\|_0$ is sufficiently small, also of (P_0) (see [239]). Thus, since most solutions constructed this way are rather sparse, most of the above-described 400 instances likely represent a favorable situation in Compressed Sensing, i.e., recoverability of sparse solutions via ℓ_1-minimization. As we wish to assess solver performance outside of this so-called ℓ_0-ℓ_1-equivalence as well, we constructed a third set of right hand sides for the first (smaller) 74 matrices as follows (for the other matrices, the construction took unreasonably long):

Given A, we start with a support size k of $m/10$, rounded to the nearest integer. Then, we take a random k-element subset $S \subseteq \{1, \ldots, n\}$ and create a vector x (having support S) as described above (with HDR or LDR, respectively). Next, we try to find a vector w such that $A^\top w \in \partial \|x\|_1$, using the alternating projections scheme included in L1TestPack (see [173]), available at `http://wwwopt.mathematik.tu-darmstadt.de/spear`. If such a w was found, x

will be an optimal solution to the (P_1) instance given by A and $b := Ax$; see Lemma 3.1. Moreover, uniqueness can be verified by [125, Theorem 4.7]: Denoting $S' := \{ i : |(A^\top w)_i| = 1 \}$, x is the unique optimum if $A_{S'}$ has full rank. If no w was found, we repeat this procedure up to 5 times before decreasing k by 1. We iterate the whole scheme until a unique solution was successfully created. Note that it is still possible that a support created this way obeys the ERC; however, we verified that this is not the case for any of the supports generated by this construction.

In total, this results in a test set of 548 instances—274 each with high (HDR) or low (LDR) dynamic range solutions. Note that the repeated verifications of the ERC required a significant amount of time, as did the calculations within the second approach: Overall, the construction of the test set took more than a week of computation time.

In general, a larger number of rows allows for larger supports, as can be seen in Figure 3.1(a). Note also that for the larger instances, the number of nonzeros generally has a larger variance.

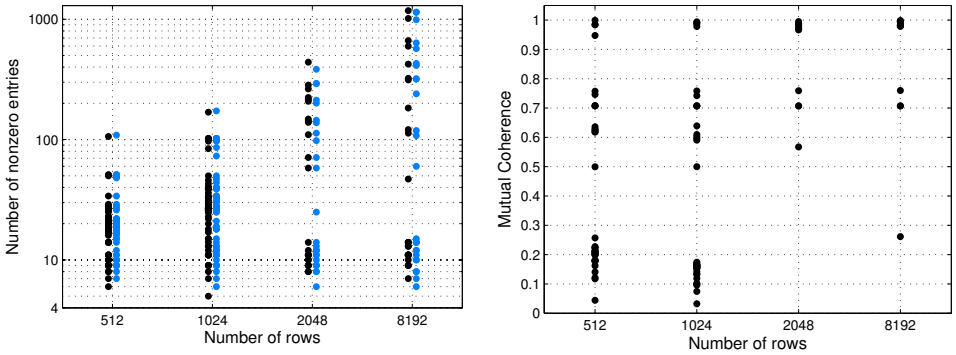
Moreover, the test set exhibits both incoherent and highly coherent matrices. The mutual coherence of a matrix A,

$$\mu(A) = \max_{1 \le j \ne k \le n} \frac{|A_j^\top A_k|}{\|A_j\|_2 \, \|A_k\|_2},$$

is an indicator of dependency among the matrix columns and hence, in a sense, for the difficulty of corresponding instances. In general, small $\mu(A)$ yields better recoverability of solutions to (P_0) via ℓ_1-minimization and other approaches, cf. Section 2.1. Moreover, we saw in Section 3.1.3 that HOC is successful under certain incoherence conditions.

The distribution of the mutual coherences of our test matrices is depicted in Figure 3.1(b). The 25 matrices with coherence larger than 0.95 are used in 114 of the 548 instances (57 each for the HDR and LDR parts); 258 instances (129 HDR, 129 LDR) have a matrix with $\mu < 0.25$. The sparse matrices generally have higher coherence (average 0.88, median 0.98) than the dense ones (average 0.40, median 0.21).

We remark that *none* of our test instances fulfill the incoherence SRC from Theorem 2.1; hence, ℓ_0-ℓ_1-equivalence is never guaranteed by means of the mutual coherence. (However, as mentioned above, it may still offer an idea of the difficulty of an instance by reflecting how "similar" the matrix columns are and hence, how hard to distinguish their respective contributions in a minimum ℓ_1-norm linear combination $b = Ax$ could be.)



(a) Number of nonzeros in the solutions plotted (log-scale) against the number of matrix rows in the resp. instances (black: HDR, blue: LDR).

(b) Mutual coherences plotted against the resp. number of matrix rows.

Figure 3.1. Test set properties: Solution sparsities and mutual coherences.

3.4 Computational Solver Comparison

In the following we present computational results for the ℓ_1-solvers and test set described above. The calculations (single-thread, no parallelization) were performed on a 64bit Ubuntu Linux system with a 3.6 GHz AMD FX-8150 eight-core CPU (8MB cache) and 8GB RAM, using MATLAB R2012a (7.14.0).

Since the solvers use a variety of different stopping criteria and tolerance parameters, we decided to assess them in a "black box" manner. Thus, we keep the default values of all algorithmic parameters. Required input parameters were chosen with respect to the goal of an exact solution of (P_1): The (main) tolerance parameter in YALL1 was set to 10^{-6}, and for ℓ_1-Homotopy we set the final regularization parameter to 0 and the iteration limit to $10n$ (this was never reached in our experiments). Moreover, ℓ_1-Magic does not include a default starting point (unlike the other solvers), so we used the projection of the origin onto the constraint set (i.e., $x^0 = A^\top(AA^\top)^{-1}b$); the additional computation times are included in the corresponding solution times stated below.

To eliminate the possible influence of on-screen output on the running times, we disabled any such output in all solvers, either by setting a corresponding option or by removing the respective lines of code from the programs. Moreover, for each instance we report the average time (geometric mean) over three runs.

In the following, we define an instance to be *solved* by an algorithm that produces

solution $\bar{x}$ if

$$\|\bar{x} - x^*\|_2 \leq 10^{-6}, \tag{3.6}$$

where x^* is the exact optimum (which we know from our test set construction). This ensures $|\,\|\bar{x}\|_1 - \|x^*\|_1\,| \leq 10^{-6}$ and $\|A\bar{x} - b\|_\infty \leq \|A\|_2\, 10^{-6}$. Note that because of the (perhaps less common) choice of considering the absolute difference $\|\bar{x} - x^*\|_2$ instead of a relative criterion, by the latter inequality, (3.6) immediately also bounds the feasibility violation. The spectral norms of the matrices in our test set are around 18 on average (median circa 10, maximum about 64); empirically, evaluating the results of all solvers over the whole test set, the quotients $\|A\bar{x} - b\|_\infty / \|\bar{x} - x^*\|_2$ are usually even smaller (very often below 1, all below 5, with a single exception at roughly 9). Thus, for the upcoming results the "solved" status implies that feasibility violation is at most of the order 10^{-6}.

Due to the different natures of the diverse stopping criteria and corresponding parameters employed by the solvers, it cannot be expected that all resulting solutions will satisfy the above condition. This must of course be taken into account when evaluating the computational results. To that end, we define a solution $\bar{x}$ produced by an algorithm to be *acceptable* if

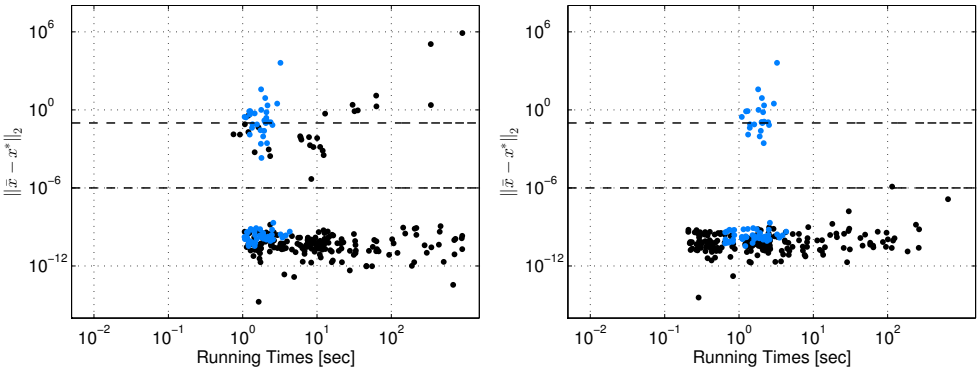
$$10^{-6} < \|\bar{x} - x^*\|_2 \leq 10^{-1}.$$

Note that this definition excludes the solutions of instances that are considered solved. If the upper bound is violated, we consider the obtained point *unacceptable*.

It should be noted that we did not distinguish between the various possible reasons why an algorithm terminated. In particular, the returned point may be the last iterate if an algorithm stopped due to numerical problems. Therefore, should an unacceptable solution be obtained, it is not necessarily implied that the algorithm claims this point to be optimal (within its own tolerances).

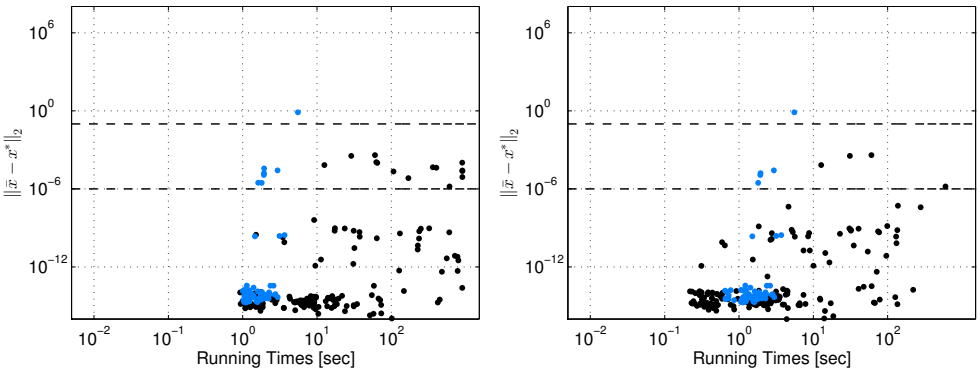
3.4.1 Numerical Results

For all eight solvers we illustrate the performance on the whole test set of 548 instances in Figures 3.3–3.9, in which we plot running time (in seconds) against distance to the (unique) optimum. For clarity, the results for the high and low dynamic range test set parts (HDR, LDR) are shown separately. Moreover, for the solvers into which we integrated HOC (i.e., ℓ_1-Homotopy, ISAL1, ℓ_1-Magic, SolveBP/PDCO, SPGL1, and YALL1), we present two figures each per different test set part, one without and one with HOC. The different construction methods of



(a) HDR test set, without HOC

(b) HDR test set, with HOC

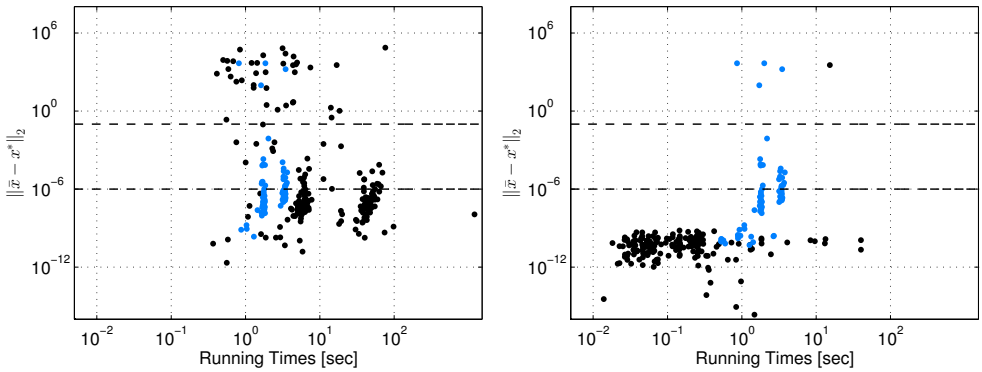


(c) LDR test set, without HOC

(d) LDR test set, with HOC

Figure 3.2. Results for ISAL1. The plots show running times versus distance to optimum in loglog scale. Black and blue dots mark the ERC-based and non-ERC-based instances, respectively.

the instances (ERC-based, non-ERC-based) are distinguishable by different colors in the plots. Note also that all figures are in double-logarithmic scale, and that they share the same axes limits (thus, they are directly comparable). Furthermore, it is worth mentioning that the general appearance, or "shape", of the point clouds hardly changes if we plot relative instead of absolute distances (and adapt the scale appropriately). Thus, the results can be interpreted in some sense independently of the choice of distance criterion.



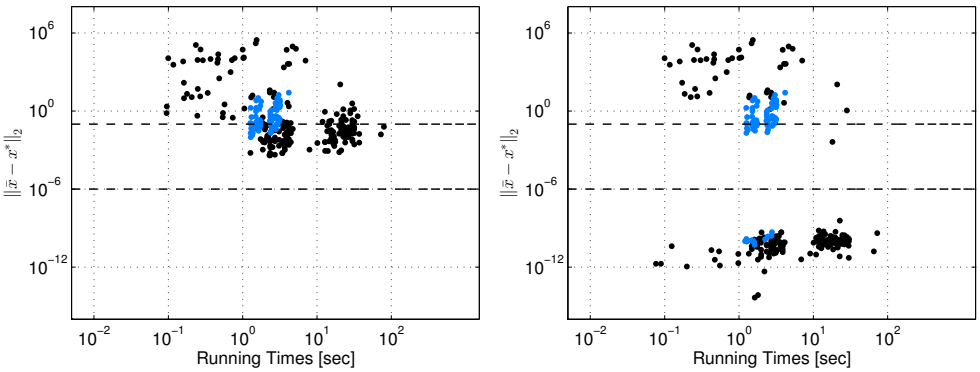
(a) HDR test set, without HOC

(b) HDR test set, with HOC

(c) LDR test set, without HOC

(d) LDR test set, with HOC

Figure 3.3. Results for ℓ_1-Homotopy. The plots show running times versus distance to optimum in loglog scale. Black and blue dots mark the ERC-based and non-ERC-based instances, respectively.



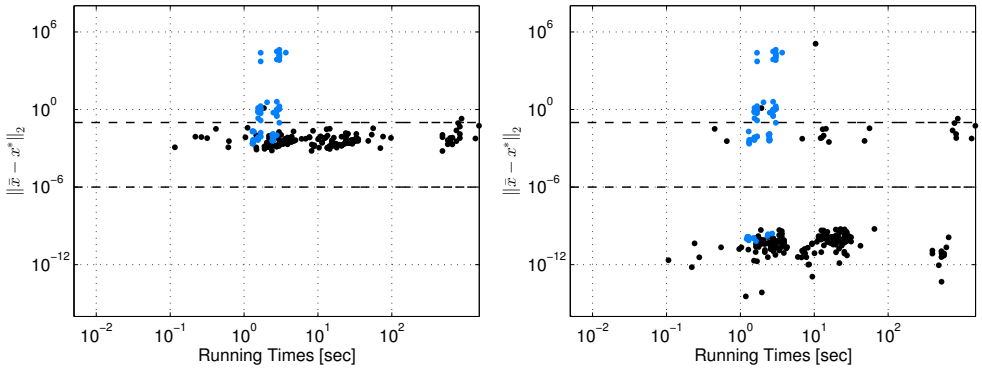
(a) HDR test set, without HOC

(b) HDR test set, with HOC

(c) LDR test set, without HOC

(d) LDR test set, with HOC

Figure 3.4. Results for ℓ_1-Magic. The plots show running times versus distance to optimum in loglog scale. Black and blue dots mark the ERC-based and non-ERC-based instances, respectively.



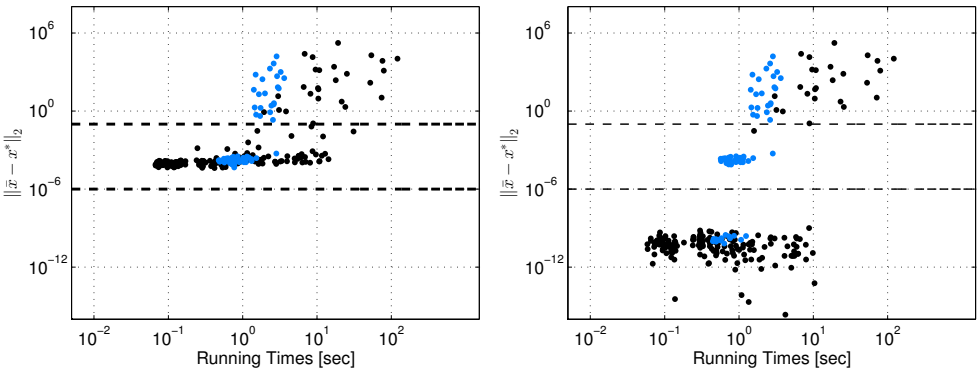
(a) HDR test set, without HOC

(b) HDR test set, with HOC

(c) LDR test set, without HOC

(d) LDR test set, with HOC

Figure 3.5. Results for SolveBP/PDCO. The plots show running times versus distance to optimum in loglog scale. Black and blue dots mark the ERC-based and non-ERC-based instances, respectively.



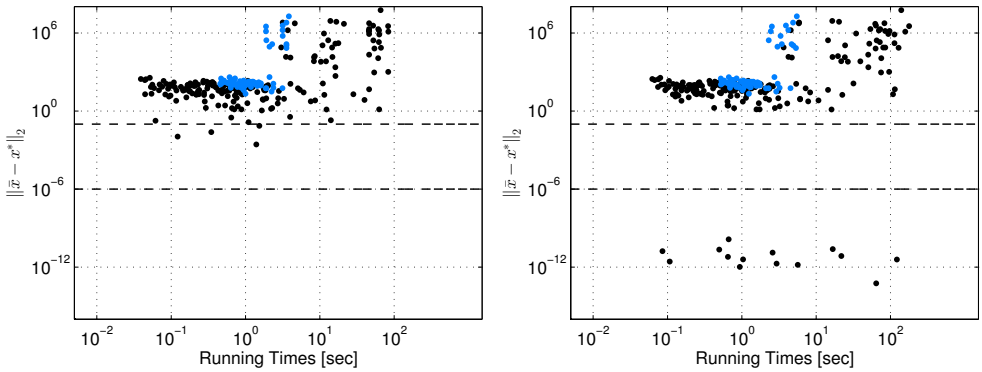
(a) HDR test set, without HOC

(b) HDR test set, with HOC

(c) LDR test set, without HOC

(d) LDR test set, with HOC

Figure 3.6. Results for SPGL1. The plots show running times versus distance to optimum in loglog scale. Black and blue dots mark the ERC-based and non-ERC-based instances, respectively.



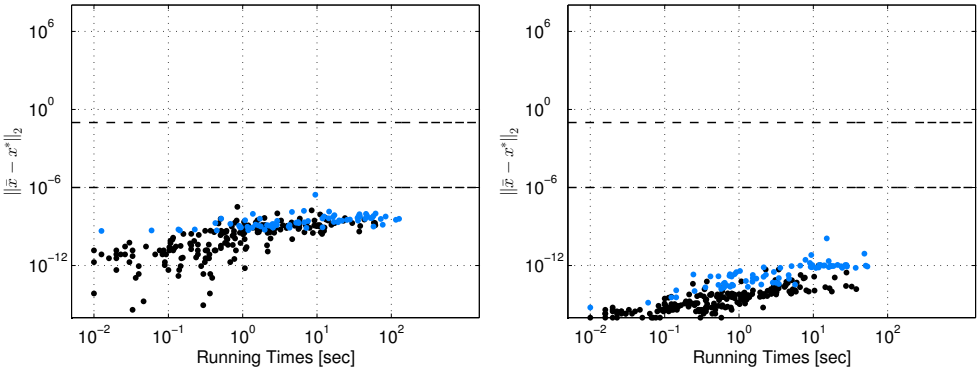
(a) HDR test set, without HOC

(b) HDR test set, with HOC

(c) LDR test set, without HOC

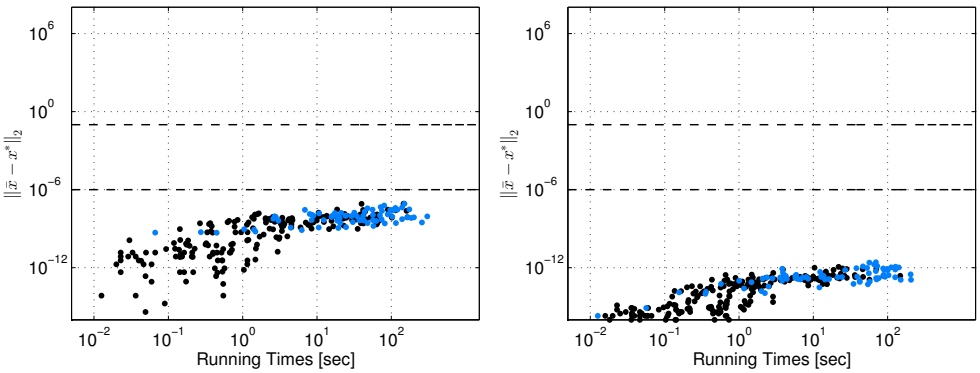
(d) LDR test set, with HOC

Figure 3.7. Results for YALL1. The plots show running times versus distance to optimum in loglog scale. Black and blue dots mark the ERC-based and non-ERC-based instances, respectively.



(a) HDR test set (without HOC) (b) LDR test set (without HOC)

Figure 3.8. Results for CPLEX. The plot shows running times versus distance to optimum in loglog scale. Black and blue dots mark the ERC-based and non-ERC-based instances, respectively.



(a) HDR test set (without HOC) (b) LDR test set (without HOC)

Figure 3.9. Results for SoPlex. The plot shows running times versus distance to optimum in loglog scale. Black and blue dots mark the ERC-based and non-ERC-based instances, respectively.

Table 3.3. Percentage of instances yielding different solutions statuses, per solver. (CPLEX and SoPlex both reached "solved" for *every* instance.)

	% solved			% acceptable			% unacceptable		
Solver	HDR	LDR	all	HDR	LDR	all	HDR	LDR	all
ISAL1	78.1	92.3	85.2	11.3	7.3	9.3	10.6	0.4	5.5
ISAL1 (HOC)	91.2	96.7	94.0	3.6	2.9	3.3	5.1	0.4	2.7
SPGL1	0.0	0.0	0.0	82.5	90.1	86.3	17.5	9.9	13.7
SPGL1 (HOC)	69.0	72.6	70.8	13.9	17.9	15.9	17.2	9.5	13.3
YALL1	0.0	0.0	0.0	1.5	78.1	39.8	98.5	21.9	60.2
YALL1 (HOC)	5.1	64.2	34.7	0.0	18.6	9.3	94.9	17.2	56.0
ℓ_1-Hom.	65.3	83.6	74.5	19.0	6.6	12.8	15.0	9.5	12.2
ℓ_1-Hom. (HOC)	92.3	97.8	95.1	5.8	0.4	3.1	1.8	1.8	1.8
SolveBP	0.0	0.0	0.0	85.0	99.3	92.2	15.0	0.7	7.8
SolveBP (HOC)	70.1	0.0	35.0	14.6	99.3	56.9	15.3	0.7	8.0
ℓ_1-Magic	0.0	15.3	7.7	53.3	82.8	68.1	46.7	1.8	24.3
ℓ_1-Magic (HOC)	62.0	76.6	69.3	5.1	21.5	13.3	32.8	1.8	17.3

Let us start by looking at the results without HOC; the changes induced by integrating HOC into the solvers are discussed in Section 3.4.2 below.

Firstly, we observe that only CPLEX and SoPlex are able to solve *all* instances with respect to the above measure (3.6). It is remarkable that the dual simplex methods from CPLEX and SoPlex perform that well even for the dense instances, since the codes are optimized for sparse instances. Moreover, it should be noted that the time taken by CPLEX or SoPlex to actually solve an instance accounts for a significantly smaller portion of the total time for instances with sparse matrices (roughly 74% on average, median 74%, for both CPLEX and SoPlex) as opposed to dense ones (circa 90% (CPLEX) or 89% (SoPlex) on average, median 96% for both). The overhead arises in the interface to the respective solver and can possibly be improved; however, note that all instances with sparse matrices were solved in well below one second by both solvers.

A large part of the instances was also solved by ISAL1 and, to a slightly lesser extent, by ℓ_1-Homotopy. SPGL1 and SolveBP/PDCO produce acceptable solutions for a majority of instances (but solve none), in accordance with their respective default accuracy settings. The two interior-point solvers, ℓ_1-Magic and SolveBP/PDCO, behave quite differently: Unlike the latter, ℓ_1-Magic reaches acceptable solutions only for about half the HDR instances, but solves several LDR instances. Similarly, YALL1 hardly produces any acceptable solution for the HDR test set part and only for about three quarters of the LDR instances. (We also ran YALL1 with input tolerance parameter 10^{-12} on our test set, but the results were only marginally better

w.r.t. solution quality; the runtimes naturally increased.)

In fact, note that all solvers can handle the LDR instances better than the HDR instances: The number of solved or acceptable instances is notably higher on the LDR test set part, see Table 3.3.

Looking also at the running times (for now, still without HOC), we immediately observe from the figures that SPGL1 and YALL1 are relatively fast, but all other solvers are somewhat comparable in speed, with SolveBP/PDCO and ISAL1 being on the slower end of the spectrum. However, such a cursory analysis is hardly fair, given the different solution accuracy demands the solvers try to satisfy. By default, ISAL1, ℓ_1-Homotopy, CPLEX, and SoPlex aim at high accuracy (i.e., "solved" status), while the other solvers are content with "acceptable" solutions. Thus, we look at these two solver groups separately and compare their running times on the subsets of instances which all four solvers in the respective group solved, or for which they reached at least acceptable solutions, respectively, see Tables 3.4 and 3.5. Since YALL1 performed extremely poorly on the HDR instances (see Table 3.3), we excluded it from Table 3.5 to avoid a strong bias towards LDR instances in the stated running times.

From these tables, we observe that for the smaller instances (having up to 4096 columns), SPGL1 is clearly the fastest algorithm to obtain acceptable solutions; for the large-scale instances (almost all of which have sparse matrices), ℓ_1-Magic performs better than SPGL1, but the LP codes CPLEX and SoPlex are much faster still *and* actually solve all instances. For solving the smaller instances to high accuracy, however, one can do better than these two: While CPLEX still has the lowest average running times (cf. the rows corresponding to $n \leq 4096$ in Table 3.4), it often yields high maximum times for the listed instance groups; SoPlex and ISAL1 seem good noncommercial alternatives: SoPlex—like CPLEX—solves all instances, and ISAL1 solves many, typically requiring lower running time than SoPlex and ℓ_1-Homotopy (and smaller maximum times even than CPLEX). Moreover, ℓ_1-Homotopy is much faster than ISAL1 on large-scale problems, but solves considerably less instances. Similarly, SolveBP/PDCO produces the most acceptable solutions but is much slower than the other algorithms in Table 3.5. Finally, it should be mentioned that YALL1 is comparable to SPGL1 on the LDR test set part, although it achieved acceptable solutions fewer times and is slightly slower.

We note in particular that ℓ_1-Homotopy exhibits moderate to large running times and produces over 12% unacceptable solutions. This is perhaps surprising, since many instances have very sparse solutions, so one could expect the k-step solution property (cf. [93]) to hold; then, ℓ_1-Homotopy should obtain the true optimum very quickly. We will therefore discuss the behavior of ℓ_1-Homotopy, as well as possible improvements of the implementation at hand, in more detail in Section 3.4.3 below.

Table 3.4. Geometric means of the running times (in seconds) of the algorithms (*without* HOC) aiming at "solved" solution status, for instances that all four could solve, grouped by number n of columns.

n	used/of		ℓ_1-Hom.	ISAL1	CPLEX	SoPlex
1024	56/72	solved	63	64	72	72
		avg. t	1.70	1.35	0.37	0.89
		max t	6.18	2.93	2.56	13.02
1536	60/72	solved	66	66	72	72
		avg. t	2.20	1.63	0.75	1.94
		max t	7.06	5.09	5.56	41.76
2048	101/144	solved	120	121	144	144
		avg. t	3.42	3.26	1.59	4.53
		max t	50.41	14.40	19.17	89.95
3072	53/72	solved	60	61	72	72
		avg. t	6.75	6.72	5.26	13.87
		max t	55.17	25.04	44.43	150.00
4096	64/94	solved	73	82	94	94
		avg. t	6.32	9.98	3.22	7.96
		max t	74.30	70.82	126.96	303.95
6144	5/16	solved	6	12	16	16
		avg. t	2.02	31.61	0.03	0.04
		max t	19.95	37.27	0.03	0.05
8192	6/22	solved	6	16	22	22
		avg. t	3.08	63.32	0.27	0.39
		max t	19.22	73.11	2.13	2.87
12288	2/4	solved	2	4	4	4
		avg. t	1.48	123.14	0.05	0.09
		max t	2.03	147.76	0.05	0.10
16384	6/16	solved	6	14	16	16
		avg. t	18.73	194.33	0.21	0.32
		max t	1204.31	237.66	0.29	0.42
24576	1/16	solved	1	13	16	16
		avg. t	5.86	460.46	0.30	0.44
		max t	5.86	460.46	0.30	0.44
32768	3/16	solved	4	14	16	16
		avg. t	5.96	857.45	0.36	0.57
		max t	7.45	900.03	0.36	0.57
49152	0/4	solved	1	0	4	4
		avg. t	–	–	–	–
		max t	–	–	–	–

Table 3.5. Geometric means of the running times (in seconds) of the algorithms (*without* HOC) aiming at "acceptable" solution status, for instances that all three could reach at least acceptable solutions, grouped by number n of columns.

n	used/of		ℓ_1-Magic	SPGL1	SolveBP
1024	63/72	$\leq$ acceptable	65	68	72
		avg. t	1.52	0.16	1.28
		max t	2.42	1.63	2.32
1536	58/72	$\leq$ acceptable	62	68	67
		avg. t	2.27	0.23	1.76
		max t	3.94	1.49	3.80
2048	112/144	$\leq$ acceptable	122	131	134
		avg. t	5.07	0.38	4.07
		max t	17.64	6.03	17.44
3072	54/72	$\leq$ acceptable	57	66	61
		avg. t	11.96	0.70	9.40
		max t	29.43	4.68	28.21
4096	57/94	$\leq$ acceptable	64	81	81
		avg. t	9.42	0.96	9.03
		max t	37.00	10.82	35.12
6144	7/16	$\leq$ acceptable	8	12	16
		avg. t	0.37	0.75	1.64
		max t	1.12	2.94	5.77
8192	9/22	$\leq$ acceptable	13	12	21
		avg. t	5.16	3.11	8.68
		max t	79.64	12.86	76.06
12288	1/4	$\leq$ acceptable	2	2	4
		avg. t	0.84	1.99	5.13
		max t	0.84	1.99	5.13
16384	8/16	$\leq$ acceptable	8	14	16
		avg. t	2.82	2.40	64.57
		max t	9.20	23.01	552.15
24576	4/16	$\leq$ acceptable	7	8	15
		avg. t	2.49	3.76	297.78
		max t	5.72	5.00	485.01
32768	6/16	$\leq$ acceptable	6	9	15
		avg. t	6.63	7.26	113.33
		max t	29.88	33.77	397.44
49152	1/4	$\leq$ acceptable	1	2	3
		avg. t	7.39	8.89	310.30
		max t	7.39	8.89	310.30

Additionally, the Tables 3.4 and 3.5 show that the running times generally increase with the number of columns, i.e., with the dimension of the solution vector. Thus, the points on the right in the point clouds in Figures 3.3–3.9 usually belong to larger instances. (Note that this also explains why the non-ERC-based instances appear relatively central in many plots.)

Finally, it should also be noted that all solvers except CPLEX and SoPlex would benefit from fast matrix-vector multiplications (e.g., for the real sinusoid transform RST). This is especially true for SPGL1, YALL1, and ISAL1, which use only a small number of matrix-vector multiplications per iteration.

As we will discuss in the next part, the picture we get for the running time and solution accuracy behavior changes quite significantly when we reconsider the algorithms, this time *with* HOC integrated into them.

3.4.2 Impact of the Heuristic Optimality Check

The benefit of HOC is clearly visible for ISAL1, ℓ_1-Homotopy, ℓ_1-Magic, SolveBP/PDCO, and SPGL1: In all cases the point clouds "moved down left", although the speed-up is less distinct for the interior-point solvers ℓ_1-Magic and SolveBP/PDCO than for the other three (first-order) solvers. Moreover, it becomes apparent from Table 3.3 and from comparing the two pairs of plots ((a)/(b) and (c)/(d)) in Figures 3.2–3.6, respectively, that by using HOC, one can often drastically improve the accuracy of the obtained solutions, especially on instances for which only acceptable solutions were reached without HOC (i.e., solutions corresponding to markers lying between the horizontal dashed lines). A specific example is shown in Figure 3.10, where one clearly sees how HOC allows for jumping to the optimum (up to numerical precision). Note that the slightly better accuracy of the solution produced by HOC in ISAL1 (left picture) is due to numerical issues; here, HOC successfully computed x^* (to machine precision) from an overestimated support.

An exception to the overall success of integrating HOC is YALL1, cf. Figure 3.7. Here, the performance only improved for very few HDR instances but for about 64% of the LDR instances, see Table 3.3. In YALL1, the (approximate) supports typically vary often and significantly, implying that an early identification of the optimal support is often impossible. Moreover, YALL1 produces a notable amount of acceptable solutions only on the LDR test set part; as we already observed, HOC seems particularly successful in improving solutions from that region. Thus, including HOC in YALL1 is relatively often unsuccessful (at least for HDR instances), but of course introduces overhead.

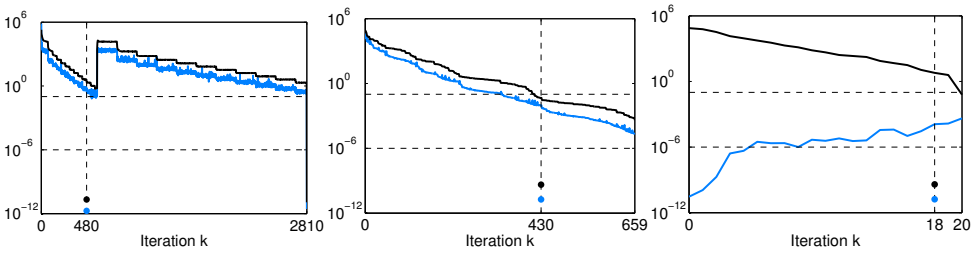


Figure 3.10. Impact of HOC in the solvers ISAL1 (left picture), SPGL1 (middle), and ℓ_1-Magic (right) for the HDR instance consisting of the 1024×8192 matrix A and b constructed with the second ERC-based variant. The curves trace the distance of the current iterates to the known optimum ($\|x^k - x^*\|_2$, black) and the current feasibility violation ($\|Ax^k - b\|_\infty$, blue) per iteration k, in logarithmic scale. The dots indicate the corresponding measures after HOC was successful, at the corresponding iteration.

Further details on the impact of HOC can be found in Table 3.6: The first row group shows how many of the instances (with respect to the whole test set, HDR only, or LDR only, respectively) that were solved by the original algorithm were solved *faster* when employing HOC. The second group of rows gives the numbers of instances that were solved *only* when HOC was used. It is important to note that in most cases listed in this second row, the variant employing HOC was not only able to solve the problem, but also required less time than the unmodified algorithm needed to reach an inferior solution. For instance, of the 380 instances that ℓ_1-Magic with HOC could solve, 379 times the early termination due to HOC success led to a smaller running time compared to ℓ_1-Magic without HOC (which only reached solved status on 42 instances).

The next two row groups of Table 3.6 give the percentages of improvements (in the sense defined by the first two row groups) achievable by including HOC, with respect to the two methods used for constructing the corresponding instances, cf. Section 3.3. Note that the HOC success rate is considerably higher for the ERC-based test set than for the non-ERC part; a theoretical explanation for this was provided by Theorem 3.6 in Section 3.1.3, see also Remark 3.10 at the end of this subsection.

The fifth group of three rows shows the average relative speed-up that HOC yields for each solver, over the whole test set or the HDR and LDR parts separately, respectively. The *average relative speed-up* is defined as

$$\frac{1}{\#\text{ instances}} \sum_i \frac{t_i^{\text{w/o HOC}} - t_i^{\text{w/ HOC}}}{t_i^{\text{w/o HOC}}};$$

Table 3.6. Impact of HOC in the solvers.

	ℓ_1-Hom.	ISAL1	ℓ_1-Magic	SPGL1	SolveBP	YALL1
solved faster	329/408	386/467	41/42	0/0	0/0	0/0
(HDR)	(148/179)	(186/214)	(0/0)	(0/0)	(0/0)	(0/0)
(LDR)	(181/229)	(200/253)	(41/42)	(0/0)	(0/0)	(0/0)
solved extra	113	49	338	388	192	190
(HDR)	(74)	(37)	(170)	(189)	(192)	(14)
(LDR)	(39)	(12)	(168)	(199)	(0)	(176)
% imp. ERC	99.5	98.2	85.2	87.8	45.0	44.8
(HDR)	(99.5)	(99.5)	(78.0)	(87.0)	(90.0)	(7.0)
(LDR)	(99.5)	(97.0)	(92.5)	(88.5)	(0.0)	(82.5)
% imp. non-ERC	29.7	28.4	25.7	25.0	8.1	7.4
(HDR)	(31.1)	(32.4)	(18.9)	(20.3)	(16.2)	(0.0)
(LDR)	(28.4)	(24.3)	(32.4)	(29.7)	(0.0)	(14.9)
% avg. rel. speed-up	67.9	57.1	10.4	15.4	4.8	−25.8
(HDR)	(68.9)	(58.8)	(8.6)	(8.7)	(11.0)	(−50.9)
(LDR)	(67.0)	(55.3)	(12.1)	(22.0)	(−1.5)	(−0.8)
# faster	444	438	413	365	240	141
% speed-up if faster	84.6	71.9	14.0	24.1	12.8	50.3
# slower	104	110	135	183	308	407
% overhead if slower	2.9	1.8	0.8	1.9	1.4	26.3

it therefore incorporates the running time changes induced by HOC in both directions, i.e., not only speed-ups are considered but also overheads. The last four rows in Table 3.6 yield the number of instances (out of 548 in the whole test set) for which the algorithm variant with HOC was faster than the one without it, and the relative speed-up obtained on these instances, followed by the number of instances where HOC slowed down the respective solver and the corresponding relative overhead on those instances.

From Table 3.6, we see that large parts of instances that a solver could also solve without HOC are solved faster when HOC is employed; the actual speed-ups on instances where the variant with HOC was faster are most impressive for ℓ_1-Homotopy (about 85% faster) and ISAL1 (about 72%). As already observed earlier from Figures 3.4 and 3.5, the speed-ups are not as pronounced for the interior-point methods ℓ_1-Magic and SolveBP/PDCO, but still quite significant. SPGL1 and YALL1 are very fast without HOC already, so it does not come as a surprise that the speed-up rate is not as high as for the other first-order methods. In fact, for YALL1, HOC is mostly ineffective on the HDR test set part and apparently also too often slows it down on the LDR part, which results in a "negative speed-up", i.e., an overhead. This can also be observed for SolveBP/PDCO on the LDR part:

On average, HOC introduces a relative overhead of 1.5% here, while on the HDR instances there is a relative speed-up of 11%, which combines to a total average relative speed-up of almost 5%. Except for YALL1, the last two rows of the table show that the actual overheads occurring in case HOC slows down the algorithm are rather small (between 0.8 and 2.9%). Combined with the much higher speed-up rates on instances where HOC decreased the running time, we end up with average relative speed-ups for each solver (except YALL1) that are quite substantial.

It is somewhat surprising that in SolveBP/PDCO, HOC is not once successful on the LDR test set part. Possibly, a different way to obtain the approximate supports used for HOC could resolve this issue, which did not seem to be a problem in any other solver. Moreover, it is worth mentioning that SPGL1 with HOC frequency $R = \lfloor m/500 \rfloor$ could solve 16 more instances (mostly from the non-ERC-based test set parts) than with $R = \lfloor m/100 \rfloor$; however, then, we end up with a relative overhead of almost 30% as opposed to the average relative speed-up of 15.4% otherwise. Since spending more time can naturally lead to more accurate solutions (without even employing HOC), the choice $R = \lfloor m/100 \rfloor$ seemed more sensible to us.

Since with HOC, many algorithms reach "solved" solution status quite often, we should reconsider which algorithm is "the best" (a tag that could arguably have been affixed to CPLEX or SoPlex by the preceding discussion in Section 3.4.1). To this end, we again compare running times on instances which all solvers could now solve, see Table 3.7. Here, we leave out SolveBP/PDCO and YALL1 due to their problems with LDR or HDR instances, respectively. Regarding the others, we know from Table 3.3 that *with* HOC, SPGL1 and ℓ_1-Magic solve about 70% of all instances, ℓ_1-Homotopy about 95%, and ISAL1 94%. (Recall that SoPlex and CPLEX both solve all instances anyway.)

The results summarized in Table 3.7 show that by including HOC, both ℓ_1-Homotopy and SPGL1 are now clearly faster than ISAL1, which—although it also benefits from a large speed-up due to HOC—is the slowest solver on the large-scale (sparse-matrix) instances with 6144 columns or more. On the smaller instances (up to column size 4096), ℓ_1-Homotopy, SPGL1, and ISAL1 all beat the LP solvers—even the commercial CPLEX—in terms of the average running time. The interior-point code ℓ_1-Magic is not competitive: On the small instances, all other codes are much faster[4], and while the code clearly works better on sparse instances (i.e., the larger-scale ones), it still is the second-worst freely available algorithm considered here, both in terms of running time and number of solved instances. Similarly, although still significantly faster than ℓ_1-Magic and SoPlex (and in some cases also CPLEX), ISAL1 is always slower on the smaller instances than ℓ_1-Homotopy and SPGL1.

[4] The runtime differences are large enough to expect that ℓ_1-Magic will not "catch up" even with the `linsolve` options enabled (provided it then terminates with a solution and does not abort with an error, cf. Remark 3.9), though we have not tested this.

Table 3.7. Geometric means of the running times (in seconds) of selected algorithms *with* HOC, for instances that all could solve, grouped by number n of columns. (CPLEX and SoPlex do not employ HOC but still solve all instances.)

n	used/of		ℓ_1-Hom.	ISAL1	ℓ_1-Magic	SPGL1	CPLEX	SoPlex
1024	61/72	solved	71	70	61	62	72	72
		avg. t	0.07	0.35	1.31	0.12	0.34	0.78
		max t	1.16	1.81	2.01	1.08	2.56	15.66
1536	51/72	solved	69	68	54	53	72	72
		avg. t	0.05	0.44	2.06	0.13	0.58	1.17
		max t	1.20	1.35	3.10	1.17	4.55	34.49
2048	103/144	solved	139	134	108	107	144	144
		avg. t	0.11	1.02	5.03	0.26	1.36	3.27
		max t	1.26	2.59	14.69	4.09	17.24	90.14
3072	48/72	solved	65	65	52	50	72	72
		avg. t	0.19	2.36	14.84	0.52	3.95	9.33
		max t	1.69	12.26	26.67	6.60	43.36	113.40
4096	59/94	solved	87	88	64	65	94	94
		avg. t	0.18	3.48	9.42	0.58	1.64	3.54
		max t	1.01	30.98	30.03	7.92	60.89	204.03
6144	8/16	solved	16	15	8	11	16	16
		avg. t	0.07	6.90	0.23	0.36	0.02	0.03
		max t	0.44	25.63	0.82	0.97	0.03	0.05
8192	9/22	solved	20	20	10	11	22	22
		avg. t	0.15	15.07	4.07	1.17	0.21	0.31
		max t	0.42	67.41	72.54	8.77	2.17	2.96
12288	0/4	solved	4	4	2	1	4	4
		avg. t	–	–	–	–	–	–
		max t	–	–	–	–	–	–
16384	5/16	solved	14	16	8	11	16	16
		avg. t	0.41	21.41	1.09	0.59	0.12	0.20
		max t	1.85	32.05	6.96	1.59	0.26	0.40
24576	6/16	solved	16	16	7	8	16	16
		avg. t	0.29	68.31	1.43	2.79	0.17	0.27
		max t	0.35	185.28	4.24	6.18	0.31	0.46
32768	4/16	solved	16	15	5	7	16	16
		avg. t	0.63	122.13	4.53	3.85	0.35	0.56
		max t	1.75	135.06	8.42	8.16	0.36	0.56
49152	1/4	solved	4	4	1	2	4	4
		avg. t	0.65	220.72	6.27	4.79	0.44	0.68
		max t	0.65	220.72	6.27	4.79	0.44	0.68

From the way the running times vary with the number of columns, Table 3.7 also shows that the running time of ℓ_1-Homotopy (with HOC) mostly seems to depend on the solution sparsity; the increase with larger column number is not nearly as

significant as for the other solvers. Clearly, this is an advantage for the Compressed Sensing scenario, where ℓ_1-minimization is employed to recover *sparse* solutions to (large-scale) systems $Ax = b$.

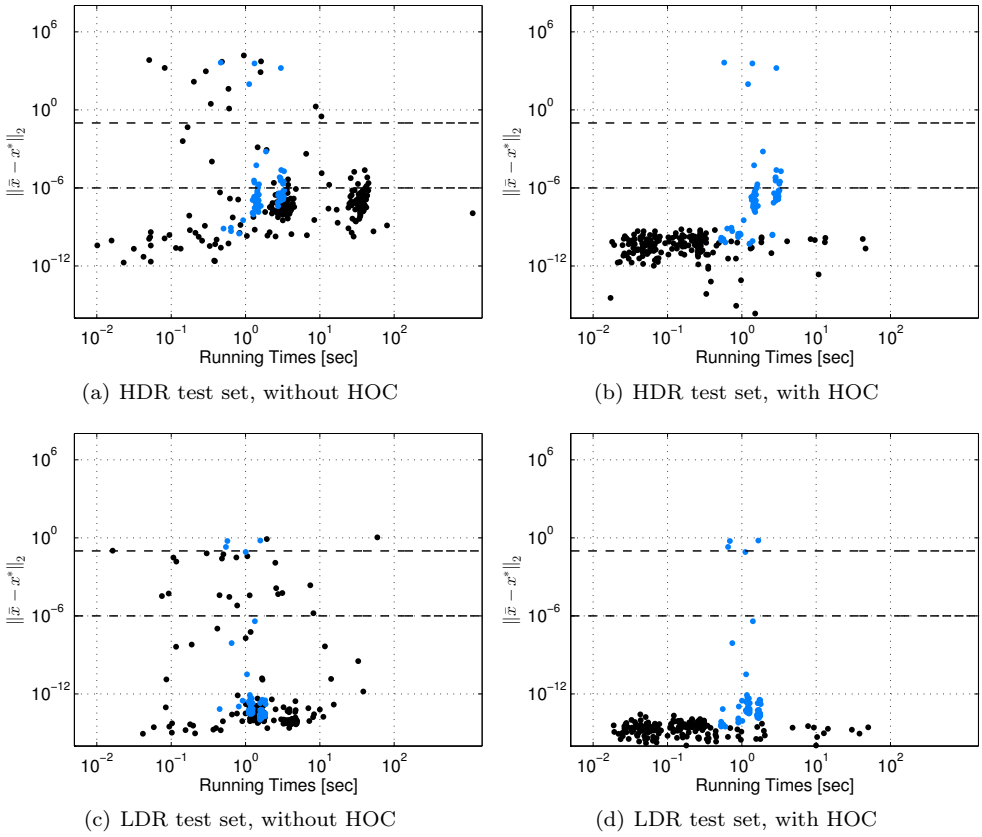
To summarize, for the test instances with at most 8192 columns, ℓ_1-Homotopy with HOC integrated into it solves almost all instances, and does so much faster than all noncommercial alternatives and, in most cases, also faster than CPLEX. On the larger instances, which mostly have sparse matrices, CPLEX solves all instances and is the fastest algorithm. However, both SoPlex and ℓ_1-Homotopy (with HOC) are only fractions of a second slower, solving all or almost all instances, respectively, and therefore offer the best freely available codes in this regime.

It is noteworthy that this situation does not change when we only ask for acceptable solutions: ℓ_1-Homotopy with HOC is faster than all other solvers for instances up to column size 8192 (the next fastest are—usually, but not always—SPGL1 and ISAL1); beyond that size, CPLEX is fastest, followed by SoPlex and then again ℓ_1-Homotopy with HOC. Both ℓ_1-Magic and SolveBP/PDCO turn out to not be competitive even with HOC (although ℓ_1-Magic ends up in fourth place on the large-scale instances).

Remark 3.10. As mentioned earlier, the observation that HOC is more often successful on instances constructed using the ERC can be explained by Theorem 3.6. Moreover, note that the ERC guarantees the existence of $\epsilon > 0$ such that the magnitude of $(A^\top w^*)_j$ (recall $w^* = -(A_{S^*}^\top)^\dagger \operatorname{sign}(x^*_{S^*})$) is smaller than $(1 - \epsilon)$ for j outside of the support of x^*, and hence, some inaccuracy in the calculation of w^* is tolerated. Generally however, there is no guarantee that w^* yields the desired properties. Therefore, the approach taken by HOC (i.e., working with $\hat{w} \approx w^*$) does not need to be effective, as is reflected by the lower success rates on the second part of the test set, where the ERC does not hold. Nevertheless, even without a theoretical justification of the choice of $\hat{w}$, HOC still works for a significant number of these instances as well.

3.4.3 The Behavior of ℓ_1-Homotopy

As mentioned above, the unconvincing behavior of ℓ_1-Homotopy is somewhat unexpected. One reason could be a large coherence among dictionary columns leading to (many) more breakpoints encountered along the homotopy solution path. However, ℓ_1-Homotopy produced unacceptable solutions for only 39 of the 114 instances with highly coherent dictionaries (HDR: 26/57, LDR: 13/57), while 52 of the remaining 75 instances (HDR: 20/31, LDR: 32/44) were solved (in the sense defined



(a) HDR test set, without HOC

(b) HDR test set, with HOC

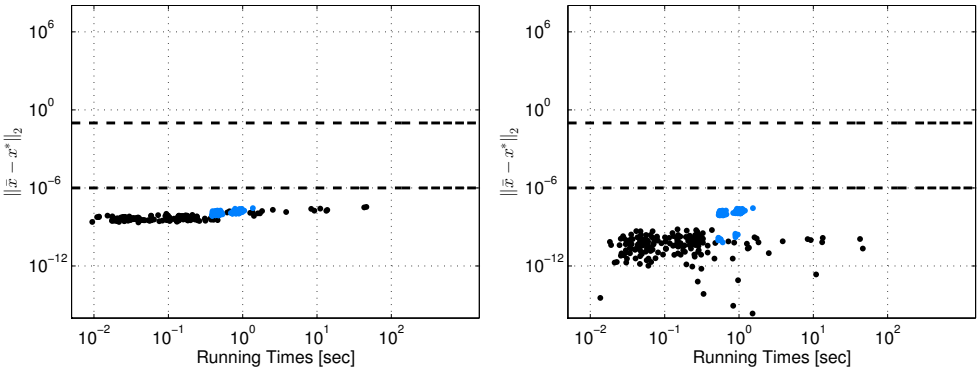
(c) LDR test set, without HOC

(d) LDR test set, with HOC

Figure 3.11. Results for ℓ_1-Homotopy, modified to terminate as soon as a problem is recognized. The plots show running times versus distance to optimum in loglog scale. Black and blue dots mark the ERC-based and non-ERC-based instances, respectively.

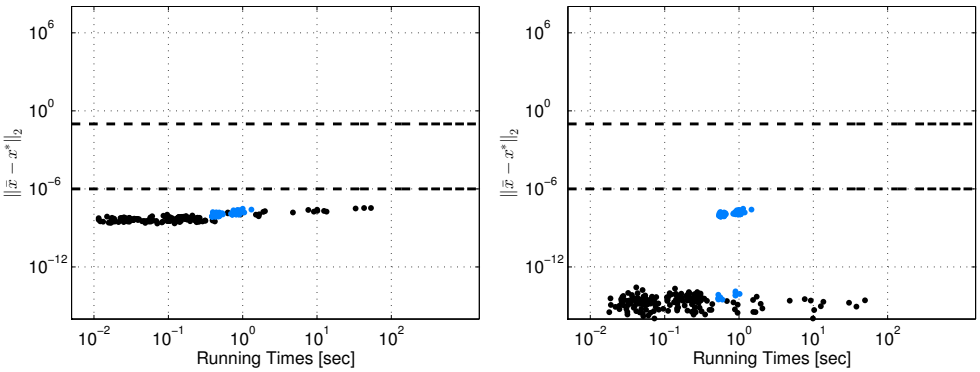
by (3.6)). Thus, this intuitive explanation actually cannot be verified.

The implementation identifies two problematic cases ("sign mismatches" and "degeneracy"), but ignores them for up to 500 more iterations. Terminating the algorithm as soon as one of these cases occurs yields the results depicted in Figure 3.11. Compared to Figure 3.3, the improvement is clear, even without incorporating HOC (now only 13 of the 114 highly coherent instances yield unacceptable solutions (HDR: 11, LDR: 2), and 101 (HDR: 46, LDR: 55) are solved.) Nevertheless, the general impression remains the same: One would expect the homotopy method to perform better and, in particular, not to deliver unacceptable results. Thus, we suspect



(a) HDR test set, without HOC

(b) HDR test set, with HOC



(c) LDR test set, without HOC

(d) LDR test set, with HOC

Figure 3.12. Results for ℓ_1-Homotopy with final regularization parameter 10^{-9}. The plots show running times versus distance to optimum in loglog scale. Black and blue dots mark the ERC-based and non-ERC-based instances, respectively.

numerical issues to be the reason for this unexpectedly bad performance.

Indeed, theory only guarantees convergence of the homotopy method to a BP solution if the final regularization parameter is set to 0. However, if we use 10^{-9} instead of 0, we get a completely different picture, see Figure 3.12 and compare with Figures 3.3 and 3.11. It turns out that ℓ_1-Homotopy with final regularization parameter 10^{-9}—although technically not an *exact* ℓ_1-solver—solves all instances with high accuracy and is also the fastest algorithm in the vast majority of cases. It is noteworthy that, in this case, the final results are actually worse if we terminate as soon as the implementation detects a problematic case. Moreover, HOC apparently

does not significantly improve the general performance and does not convincingly justify its induced running time overhead, although it is small.

3.4.4 Conclusions

Our experiments do not indicate a clear winner among the various exact ℓ_1-solvers. While CPLEX and SoPlex were the only ones to solve all instances to within distance 10^{-6} of the true optimum, they have the potential drawback of requiring the matrices in explicit form, while all other algorithms could also handle matrices given implicitly via matrix-vector product oracles. Moreover, CPLEX is proprietary, while all other implementations are freely available.

Currently, only our ISAL1 implementation contains the Heuristic Optimality Check (HOC) by default. However, the results clearly indicate that this is a good idea for all solvers (with the possible exception of YALL1), producing more accurate solutions and typically reducing the running times at the same time. Considering the various implementations as they actually can be downloaded to date, ISAL1 (containing HOC) may be a good alternative to CPLEX and SoPlex when small- to medium-scale problems are to be solved. However, once HOC is also integrated into the other solvers, ℓ_1-Homotopy becomes the best code on such instances, followed closely by SPGL1. For large-scale instances with sparse matrices, the two LP solvers fare best; however, ℓ_1-Homotopy with HOC is very close behind.

The average performance of ℓ_1-Homotopy could be improved significantly by a slight modification of the code and by integrating HOC. Deviating from theoretical requirements, ℓ_1-Homotopy with final regularization parameter 10^{-9} instead of 0 in fact turned out to be the fastest reliable solver. With this parameter choice, using HOC does not really seem beneficial (the accuracy is slightly improved while small overheads are often introduced). On the other hand, we also observed from a few experiments, that using an even larger regularization parameter (e.g., 10^{-4}) still allowed for extracting the optimal support from the point produced by ℓ_1-Homotopy by a simple hard-thresholding scheme like (3.1); a theoretical justification is given by the results in [240, 176] on exact support recovery. This suggests that it may, in principle, be worth integrating HOC into ℓ_1-Homotopy. In particular, this might hold true for a variant in which the homotopy path is not followed exactly but only approximately, see [180], which might avoid some numerical problems of the homotopy method that apparently occur when the "kinks" in the homotopy path become hard to distinguish as the distances between them get close to machine precision.

3.5 Equivalence of Basis Pursuit and Linear Programming

It is well-known that the Basis Pursuit problem (P_1) can be reformulated as a linear program, see (3.4) and (3.5). In fact, the results in Section 3.4 show that this approach yields some of the fastest and most reliable methods in practice. In this section, we show that (for rational data) the converse holds as well: Every (bounded, feasible) LP can be transformed into an equivalent ℓ_1-minimization problem.

We will call two problems *polynomially equivalent*, if any instance of one problem can be reduced to an instance of the other (and vice versa) in polynomial time and space. The reductions we employ here are of this type; in particular, the encoding lengths of the constructed instances are polynomially bounded by those of the respective input instances.

3.5.1 Related Work

An equivalence between a linear program and a (seemingly) different ℓ_1-minimization problem has been noted before: In [15], the problem of minimizing the ℓ_1-norm violation of an overdetermined system (also known as *least absolute value (LAV) regression*, see, e.g., [86] and the references therein),

$$\min \|M\xi - f\|_1, \tag{3.7}$$

was considered (here, $M \in \mathbb{R}^{p\times n}$ with $p > n$ and $f \in \mathbb{R}^p$), and it was shown that any bounded feasible linear program can be rewritten into the form of the dual of (3.7),

$$\min f^\top y \quad \text{s.t.} \quad M^\top y = 0,\ -\mathbb{1} \le y \le \mathbb{1}.$$

The reduction in [15] is very similar in spirit to ours (see Section 3.5.3 below); in particular, it involves a certain "Big-M" argument and dualization. However, [15] merely sketches the procedure without giving precise values for several constants needed in the transformation, only stating that they need to be "suitably large". In our reduction, we will derive the corresponding values explicitly. In particular, we will see that the constants can be chosen such that their encoding lengths depend polynomially on that of the LP.

It is noteworthy that (3.7) is actually equivalent to a problem of the form (P_1). This was observed before, e.g., in [50], where (3.7) was considered in the context of restoring code words that are corrupted by sparse errors. We provide a different

proof below, for completeness of our discussion and because parts of it are used in the results to follow.

First, note that (P_1) can be recast as an unconstrained optimization problem.

Lemma 3.11. *The Basis Pursuit problem* (P_1) *is polynomially equivalent to*

$$\min \|z\|_1 + \|d - Dz\|_1, \tag{$P_1^\dagger$}$$

where D is a matrix, d is a vector and minimization is performed over z.

Proof. Let $A \in \mathbb{R}^{m\times n}$ with $\operatorname{rank}(A) = m < n$ and $b \in \mathbb{R}^m$ be an instance for the Basis Pursuit problem $\min\{\, \|x\|_1 : Ax = b \,\}$. We make use of a basis decomposition (or variable reduction) technique as known from the simplex algorithm for linear programs: Let $B \subseteq [n]$ with $|B| = m$ be an arbitrary *basis*, i.e., such that A_B is regular (square and full-rank), and let $N := [n] \setminus B$. Clearly, some basis B can be found in polynomial time, by Gaussian elimination. Then, we can rewrite

$$Ax = b \quad \Leftrightarrow \quad A_B x_B + A_N x_N = b \quad \Leftrightarrow \quad x_B = A_B^{-1} b - A_B^{-1} A_N x_N.$$

Obviously, $x_N = 0$ always yields a feasible solution (along with $x_B = A_B^{-1} b$) for (P_1); in fact, $x_N \in \mathcal{N}(A_N)$ suffices. Let

$$z := x_N, \quad D := A_B^{-1} A_N, \quad \text{and } d := A_B^{-1} b. \tag{3.8}$$

Thus, the constraint $Ax = b$ is incorporated into the definitions of z, D and d (it holds implicitly), whence (P_1) is equivalent to a problem in form ($P_1^\dagger$):

$$\begin{aligned} (P_1) \quad &= \quad \min\{\, \|x_N\|_1 + \|x_B\|_1 : A_B x_B + A_N x_N = b \,\} \\ &= \quad \min \|z\|_1 + \|d - Dz\|_1. \end{aligned} \tag{3.9}$$

Optimal solutions x^* and z^* of (P_1) and (3.9), respectively, determine each other via (3.8), with respect to the given, fixed B and N.

Conversely, for an instance $D \in \mathbb{R}^{m\times p}$, $d \in \mathbb{R}^m$ of ($P_1^\dagger$), an equivalent instance of (P_1) is given by $A = (I, D)$ and $b = d$. The correspondence between optimal solutions z^* and x^* is again described by the relations (3.8), using $B = [m]$ and $N = \{m+1, \ldots, m+p\}$; i.e., $z^* = x_N^*$, and $(d - Dz^*) = x_B^*$ holds implicitly. □

It is easily observed that ($P_1^\dagger$) can be stated in the form (3.7):

$$\min \|z\|_1 + \|d - Dz\|_1 \quad = \quad \min \left\| \begin{pmatrix} I \\ D \end{pmatrix} z - \begin{pmatrix} 0 \\ d \end{pmatrix} \right\|_1. \tag{3.10}$$

Indeed, if D is $m \times p$ then $(I^\top, D^\top)^\top$ is $(m+p) \times p$, and hence the system whose ℓ_1-norm violation is minimized in (3.10) is overdetermined, as intended for (3.7). Combined with the proof of Lemma 3.11, this shows the first direction of the following result.

Proposition 3.12. *The Basis Pursuit problem* (P_1) *is polynomially equivalent to the LAV regression problem* (3.7).

Proof. It remains to show that we can bring any instance of (3.7) into the form ($\mathrm{P}_1^\dagger$). Given $M \in \mathbb{R}^{p \times q}$ with $p \geq q$ and $f \in \mathbb{R}^q$, assume w.l.o.g. that $\operatorname{rank}(M) = q$ and that the first q rows of M are linearly independent. With the variable substitution $z = M_{[q],[q]}\xi - f_{[q]}$, we obtain:

$$\begin{aligned} \min \|M\xi - f\|_1 &= \min \left\| \begin{pmatrix} M_{[q],[q]} \\ M_{\{q+1,\dots,p\},[q]} \end{pmatrix} \xi - \begin{pmatrix} f_{[q]} \\ f_{\{q+1,\dots,p\}} \end{pmatrix} \right\|_1 \\ &= \min \left\| \begin{pmatrix} Iz - 0 \\ Dz - d \end{pmatrix} \right\|_1 = \min \|z\|_1 + \|d - Dz\|_1, \end{aligned}$$

where $D = M_{\{q+1,\dots,p\},[q]} M_{[q],[q]}^{-1}$ and $d = f_{\{q+1,\dots,p\}} - Df_{[q]}$. Note that the resulting (P_1) instance $A = (I, D)$, $b = d$ (cf. the proof of Lemma 3.11) has $A \in \mathbb{R}^{(p-q)\times p}$, so the system $Ax = b$ is indeed underdetermined, as we require for (P_1). □

By these results, one could reduce a given linear program to (P_1) also by using the transformation sketched in [15], dualizing, and then applying Proposition 3.12. However, the reduction we will present below avoids this intermediate step involving (3.7), and builds on Lemma 3.11 directly.

3.5.2 Preliminaries

In the following, we give further auxiliary results that will be employed in the announced reduction.

Lemma 3.13. *The dual problem of* ($\mathrm{P}_1^\dagger$) *reads*

$$\max -d^\top q \quad \text{s.t.} \quad \|q\|_\infty \leq 1,\ \|D^\top q\|_\infty \leq 1. \tag{$\mathrm{D}_1^\dagger$}$$

Proof. Recall from the proof of Lemma 3.11 that ($\mathrm{P}_1^\dagger$) can be written as the Basis Pursuit problem $\min\{\, \|x\|_1 \,:\, (I, D)x = d \,\}$. By Lemma 1.5, the dual of ($\mathrm{P}_1^\dagger$) is therefore given by $\max\{\, -d^\top q \,:\, \|q^\top (I, D)\|_\infty \leq 1 \,\}$, which corresponds exactly to ($\mathrm{D}_1^\dagger$). □

Remark 3.14. Note that strong duality holds for ($P_1^\dagger$) and ($D_1^\dagger$), i.e., since both are (clearly) bounded and always have feasible points, they each have an optimal solution and the corresponding (finite) objective function values coincide. Thus, ($P_1^\dagger$) and ($D_1^\dagger$) are in fact polynomially equivalent, and by Lemma 3.11, the same holds for (P_1) and ($D_1^\dagger$).

The next lemma gathers some results about encoding lengths.

Lemma 3.15.

(i) *For $r, s \in \mathbb{Q}$, it holds that*

$$\langle rs \rangle \;\leq\; \langle r \rangle + \langle s \rangle. \tag{3.11}$$

More generally, for $A \in \mathbb{Q}^{m\times n}$ and $B \in \mathbb{Q}^{n\times p}$, it holds that

$$\langle AB \rangle \;\leq\; p\,\langle A \rangle + m\,\langle B \rangle. \tag{3.12}$$

(ii) *For all $r_1, r_2, \ldots, r_n \in \mathbb{Q}$, it holds that*

$$\langle r_1 + r_2 + \cdots + r_n \rangle \;\leq\; 2\left(\langle r_1 \rangle + \langle r_2 \rangle + \cdots + \langle r_n \rangle\right). \tag{3.13}$$

(iii) *For any nonsingular $A \in \mathbb{Q}^{n\times n}$, it holds that*

$$\langle A^{-1} \rangle \;\leq\; 4n^2\,\langle A \rangle. \tag{3.14}$$

(iv) *Given $A \in \mathbb{Q}^{m\times n}$ and $b \in \mathbb{Q}^m$, for every vertex $v = (v_1, \ldots, v_n)^\top$ of a polyhedron in any one of the forms $\{\, x \,:\, Ax \leq b \,\}$, $\{\, x \,:\, Ax = b \,\}$ or $\{\, x \,:\, Ax \leq b,\, x \geq 0 \,\}$, it holds that*

$$|v_j| \;\leq\; 2^{2\,\langle A\rangle + \langle b\rangle - 2n^2} \qquad \text{for all } j \in [n]. \tag{3.15}$$

(v) *For any positive integer $z \in \mathbb{N}$, it holds that*

$$\langle z \rangle \;\leq\; 3 + \log_2(z) \quad \text{and} \quad \langle 2^z \rangle \;\leq\; 2 + z. \tag{3.16}$$

Moreover, for $z \in \mathbb{Z}$ with $|z| > 1$ and $0 \neq r \in \mathbb{Q}$, it holds that

$$\langle 1/z \rangle = 2 + \langle z \rangle \quad \text{and} \quad \langle 1/r \rangle = \langle r \rangle. \tag{3.17}$$

(vi) *For a vector $u \in \mathbb{Q}^n$, it holds that*

$$\langle \|u\|_1 \rangle \;\leq\; 2\,\langle u \rangle. \tag{3.18}$$

Proof. The first three statements can be found in [128, pp. 30f]; the fourth appears in [97, Satz 6.6]. Moreover, $\langle z\rangle = 1+\lceil\log_2(|z|+1)\rceil \leq 1+(1+\lfloor\log_2(2z)\rfloor) \leq 3+\log_2(z)$ and $\langle 2^z\rangle = 1+\lceil\log_2(|2^z|+1)\rceil \leq 1+\lceil\log_2(2^{z+1})\rceil = 2+z$, for any $z \in \mathbb{N}$, which proves (3.16). Regarding (3.17), recall that $\langle r\rangle = \langle s\rangle + \langle t\rangle$ for mutually prime integers s and t such that $r = s/t$; thus, $1/r = t/s$ and therefore $\langle 1/r\rangle = \langle r\rangle$. Furthermore, for $z \in \mathbb{Z}$ with $|z| > 1$, $\langle 1/z\rangle = \langle 1\rangle + \langle z\rangle = 2 + \langle z\rangle$ (for $z \in \{\pm 1\}$, $\langle 1/z\rangle = \langle z\rangle$). Finally, the last statement follows directly from (3.13), since $\|u\|_1 = \sum_{j=1}^n |u_j|$ (and $\langle |r|\rangle = \langle r\rangle$ for $r \in \mathbb{Q}$). □

3.5.3 The Reduction

Consider a linear program in standard inequality form:

$$\max c^\top x \quad \text{s.t.} \quad Ax \leq b,\ x \geq 0, \tag{LP}$$

with rational data $A \in \mathbb{Q}^{m\times n}$, $b \in \mathbb{Q}^m$ and $c \in \mathbb{Q}^n$. Throughout, we assume w.l.o.g. that A does not contain any all-zero rows or columns.

Additionally, we will make the following assumptions. The first two are fairly natural, while the third allows for a better presentation of the main proof and is, in fact, not necessary (see Proposition 3.19 later).

Assumption 3.16.

(i) The inequality system of (LP) describes a *nonempty* polyhedron,

(ii) the objective function $c^\top x$ remains *bounded* over this polyhedron, and

(iii) for all $i = 1, \ldots, m$,

$$b_i \ \neq\ \frac{K}{2}\left(\left(\sum_{j=1}^n a_{ij}\right) - \|a_i^\top\|_1\right) \ =\ \frac{K}{2}\left(a_i^\top \mathbb{1} - \|a_i^\top\|_1\right),$$

where $K := 2^{2\,\langle A\rangle + \langle b\rangle - 2n^2} > 0$ is the constant from (3.15).

Theorem 3.17. *Under Assumption 3.16, the linear program* (LP) *is polynomially equivalent to the Basis Pursuit problem* (P_1).

Proof. We already know that (P_1) can be reformulated as an LP, see (3.4) or (3.5); the same clearly holds for ($D_1^\dagger$), since its constraints can be restated as $-\mathbb{1} \leq q \leq \mathbb{1}$, $-\mathbb{1} \leq D^\top q \leq \mathbb{1}$. Moreover, it is well-known that every linear program can be stated in the above standard inequality form (LP). Thus, because (P_1) and ($D_1^\dagger$)

are polynomially equivalent by Remark 3.14, it remains to show that (LP) can be transformed into an instance of $(D_1^\dagger)$ (under Assumption 3.16).

By Assumption 3.16, the optimum of (LP) is finite and therefore attained at a vertex of $\{\, x \,:\, Ax \le b,\, x \ge 0 \,\} \neq \emptyset$. Thus, we can restrict our attention to a bounded polyhedron (i.e., a polytope) by explicitly including a constraint of the sort (3.15) for each variable; see also [206, Theorem 2.2]. Note that this does not cut off any vertices, since these constraints are *implied* by the data. Since in (LP), $x \ge 0$ and hence $|x_j| = x_j$ for all $j \in [n]$, it follows from Lemma 3.15(iv) that (LP) is equivalent to

$$\max\, c^\top x \quad \text{s.t.} \quad Ax \le b,\ 0 \le x \le K\mathbb{1}, \tag{3.19}$$

with $K := 2^{2\langle A\rangle + \langle b\rangle - 2n^2}$ as in Assumption 3.16(iii). Substituting x by $\frac{K}{2}(\mathbb{1} + y)$, i.e., with $y := \frac{2}{K}x - \mathbb{1}$, we see that (3.19) is equivalent to

$$\max\, c^\top \left(\tfrac{K}{2}\mathbb{1} + \tfrac{K}{2}y\right) \quad \text{s.t.} \quad A\left(\tfrac{K}{2}\mathbb{1} + \tfrac{K}{2}y\right) \le b,\ 0 \le \tfrac{K}{2}\mathbb{1} + \tfrac{K}{2}y \le K\mathbb{1}.$$

Omitting the constant terms and factors in the objective, and denoting $r := \frac{2}{K}b - A\mathbb{1}$, this can be rewritten as

$$\begin{aligned} & \max\, c^\top y \quad \text{s.t.} \quad Ay \le r,\ -\mathbb{1} \le y \le \mathbb{1} \\ = \quad & \max\, c^\top y \quad \text{s.t.} \quad Ay \le r,\ \|y\|_\infty \le 1. \end{aligned} \tag{3.20}$$

Now observe that $-\mathbb{1} \le y \le \mathbb{1}$ implies that $-\|a_i^\top\|_1 \le a_i^\top y \le \|a_i^\top\|_1$ for all $i \in [m]$. Together with the constraint $Ay \le r$, this yields

$$-\|a_i^\top\|_1 \le a_i^\top y \le \min\{r_i, \|a_i^\top\|_1\} \qquad \text{for all } i \in [m]. \tag{3.21}$$

Note that, if $r_i > \|a_i^\top\|_1$ for some $i \in [m]$, then the constraint $a_i^\top y \le r_i$ is in fact redundant and can be omitted. Thus, we can assume w.l.o.g. that we always have $\min\{r_i, \|a_i^\top\|_1\} = r_i$. Moreover, by Assumption 3.16(i), $r_i < -\|a_i^\top\|_1$ is impossible (because it would imply inconsistency of the constraint system), and by Assumption 3.16(iii), we know that $r_i \neq -\|a_i^\top\|_1$. Also, because A contains no zero rows, we have $\|a_i^\top\|_1 \neq 0^\top$ for all $i \in [m]$. Therefore,

$$(3.21) \quad \Leftrightarrow \quad -\|a_i^\top\|_1 \le a_i^\top y \le r_i \quad \Leftrightarrow \quad -1 \le \tilde{a}_i^\top y + \delta_i \le 1, \tag{3.22}$$

where

$$\tilde{a}_i^\top := \frac{2}{\|a_i^\top\|_1 + r_i} a_i^\top \quad \text{and} \quad \delta_i := \frac{\|a_i^\top\|_1 - r_i}{\|a_i^\top\|_1 + r_i}. \tag{3.23}$$

In particular, $\delta_i \ge 0$, and if $r_i = \|a_i^\top\|_1$ then $\delta_i = 0$ and $\tilde{a}_i^\top = (1/\|a_i^\top\|_1) a_i^\top$.

Thus, we can replace the constraint system $Ay \le r$ by box constraints of the

form (3.22). Letting $\tilde{A}$ be the matrix consisting of rows $\tilde{a}_i^\top$ and δ the vector containing the numbers δ_i (as defined in (3.23)), this shows that (3.20) is equivalent to

$$\begin{aligned} & \max c^\top y \quad \text{s.t.} \quad -\mathbb{1} \le \tilde{A}y + \delta \le \mathbb{1},\ \|y\|_\infty \le 1 \\ = \quad & \max c^\top y \quad \text{s.t.} \quad \|\tilde{A}y + \delta\|_\infty \le 1,\ \|y\|_\infty \le 1. \end{aligned} \tag{3.24}$$

The only difference between (3.24) and the desired form ($\mathrm{D}_1^\dagger$) is the presence of the constant "shifts" δ_i in the constraint system. We can eliminate these shifts by a sort of *lifting* procedure, which we describe in the following.

First, consider a single box constraint

$$-1 \ \le\ \tilde{a}_i^\top y + \delta_i \ \le\ 1 \quad \Leftrightarrow \quad -1 - \delta_i \ \le\ \tilde{a}_i^\top y \ \le\ 1 - \delta_i. \tag{3.25}$$

Our goal is to transform the shift δ_i into the coefficient of a new variable $z \in [-1, 1]$, thereby obtaining the desired unit box constraint form. To that end, suppose we assign an objective function coefficient $\mathcal{M}$ to z. If $\mathcal{M}$ is sufficiently large (hence the name "Big-M" for this kind of argument), setting $z = 1$ is superior to any other choice and implicitly enforces the original constraint (3.25) on y. In fact, it is easy to see that we need only one additional variable z to transform all constraints (3.25), i.e., we replace δ by δz. Then, if $z = 1$, the original bounds are restored, and an optimal solution of (3.24) is indeed in one-to-one correspondence with the y-part of an optimal solution for

$$\max c^\top y + \mathcal{M} z \quad \text{s.t.} \quad \left\| \left(\tilde{A},\, \delta\right) \begin{pmatrix} y \\ z \end{pmatrix} \right\|_\infty \le 1,\ \left\| \begin{pmatrix} y \\ z \end{pmatrix} \right\|_\infty \le 1. \tag{3.26}$$

Note that with respect to any *fixed* value of z, replacing the constant shift δ_i by $\delta_i z$ can be regarded as changing the i-th box constraint bounds on $\tilde{A}y$. Let $\hat{A}$ and $\hat{b}$ denote the matrix and vector such that $\hat{A}y \le \hat{b}$ is equivalent to the constraints in (3.24); i.e., $\hat{A}$ is the concatenation of $\tilde{A}$, $-\tilde{A}$, I and $-I$, and $\hat{b}$ contains $\mathbb{1} - \delta$, $\mathbb{1} + \delta$, $\mathbb{1}$ and $\mathbb{1}$ (placed above each other in the given order to form $\hat{A}$ and $\hat{b}$, respectively). Replacing δ_i by $\delta_i z$ (for some fixed z) changes only the two entries in $\hat{b}$ that correspond to (3.25); namely, from $1 \pm \delta_i$ to $1 \pm \delta_i z$. Thus, considering all bound changes (w.r.t. all rows (3.25)) simultaneously, we replace $\mathbb{1} \pm \delta$ by $\mathbb{1} \pm \delta z$, respectively.

Concerning the resulting possible variation in the LP objective due to changing the right hand side $\hat{b}$ to some $\hat{b}'$, the following estimate is known [224, p. 126]:

$$\left|\ \max\{\, c^\top y \,:\, \hat{A}y \le \hat{b}\,\} - \max\{\, c^\top y \,:\, \hat{A}y \le \hat{b}'\,\}\ \right| \ \le\ n\,\beta\,\|c\|_1\,\|\hat{b} - \hat{b}'\|_\infty,$$

where β is an upper bound on the absolute values of all entries in the inverses of all regular submatrices B of $\hat{A}$. (This result is only applicable if both LPs occurring in the above expression are finite, which is, however, clear in our case, since y is bounded.) From the above discussion, in our case, we have for any $z \in [-1, 1]$ that

$$\|\hat{b} - \hat{b}'\|_\infty \;=\; \max_{i\in[m]} \big\{\, |1 - \delta_i - (1 - \delta_i z)|,\; |1 + \delta_i - (1 + \delta_i z)| \,\big\} \;=\; \max_{i\in[m]} \{\, \delta_i \,\}(1 - z);$$

recall also that each $\delta_i \geq 0$. Moreover, it is known (see [128, Lemma 1.3.3]) that for all $r \in \mathbb{Q}$, $|r| \leq 2^{\langle r\rangle - 1} - 1$. With this, we obtain an upper bound on $|(B^{-1})_{ij}|$:

$$\beta \;:=\; 2^{4n^2\langle \hat{A}\rangle} \;\geq\; 2^{4n^2\langle B\rangle} \overset{(3.14)}{\geq} 2^{\langle B^{-1}\rangle} \;\geq\; 2^{\langle |(B^{-1})_{ij}|\rangle - 1} - 1 \;\geq\; |(B^{-1})_{ij}|.$$

To summarize, due to implicit bound changes for $\tilde{A}y$ caused by introducing the variable $z \in [-1, 1]$, the objective value contribution of y can *increase* (with respect to the optimal value of (3.24)) by at most

$$n\,\beta\,\|c\|_1 \max_{i\in[m]}\{\, \delta_i \,\}(1 - z) \;=\; \underbrace{2^{4n^2\langle \hat{A}\rangle}\, n\,\|c\|_1 \max_{i\in[m]}\{\, \delta_i \,\}}_{=:\,M}\,(1 - z).$$

Now, choosing any value $\mathcal{M} > M$ as the objective function coefficient for z will suffice, i.e., it guarantees that for the problem (3.26), any optimal point $((y^*)^\top, z^*)^\top$ has $z^* = 1$ and y^* is an optimal solution of (3.24). To see this, suppose $\mathcal{M} = \mathcal{M}_\epsilon = M + \epsilon$ for some $\epsilon > 0$. Then, by the above discussion and since $-1 \leq z^* \leq 1$ (and hence, $M(1 - z^*) \geq 0$), an upper bound for the objective value in any feasible point $(y^\top, z)^\top$ of (3.26) is given by

$$c^\top y + \mathcal{M}_\epsilon z \;\leq\; c^\top y^* + M(1 - z^*) + (M + \epsilon)z^* \;=\; c^\top y^* + M + \epsilon\, z^* \;\leq\; c^\top y^* + \mathcal{M}_\epsilon.$$

Here, the last inequality holds with equality if and only if $z^* = 1$, in which case this upper bound is sharp: The optimal objective of (3.26) then is precisely $c^\top y^* + \mathcal{M}_\epsilon$, and since $z^* = 1$, y^* solves (3.24), as argued above. (Conversely, if y^* is an optimal solution of (3.24), then $((y^*)^\top, 1)^\top$ is feasible for (3.26) by construction, and for $\mathcal{M} > M$, we saw that no other value than $z^* = 1$ can possibly be optimal in (3.26), which entails that y^* is the y-part of an optimal solution of (3.26).)

Thus, (3.24) can be restated *equivalently* as (3.26), which is clearly an instance of ($D_1^\dagger$). It remains to note that with (say) $\mathcal{M} := M + 1$, all transformations described in this proof can clearly be performed in polynomial time, and the encoding length of (3.26) is polynomially bounded by that of (LP); see Section 3.5.4 below for more details. □

Remark 3.18. Clearly, if for some $j \in [n]$, the linear program contains variable bounds $\ell_j \le x_j \le u_j$, these do not have to be included into the main inequality constraint $Ax \le b$ but could be used directly, instead of the artificial construction $0 \le x_j \le K$.

The following result shows that the assumptions under which we proved Theorem 3.17 are, in a sense, not essential.

Proposition 3.19. *Theorem 3.17 remains valid even without Assumption 3.16(iii). Moreover, with $\tilde{K} \ge K$ replacing K in the construction of the instance (3.26):*

(i) *(LP) is infeasible if and only if $b_i < \tilde{K}(a_i^\top \mathbb{1} - \|a_i^\top\|_1)/2$ for some $i \in [m]$ or if $z^* < 1$ holds for every optimal solution $((y^*)^\top, z^*)^\top$ of (3.26).*

(ii) *(LP) is unbounded if and only if (3.26) constructed with $\tilde{K} > K$ has an optimal solution $((y^*)^\top, 1)^\top$ with $y_i^* = 1$ for some $i \in [m]$.*

Proof. We start by considering the consequences of omitting Assumption 3.16(iii), which deals with the case $r_i = -\|a_i^\top\|_1$ that might be encountered during the transformation process. Clearly, with $K = 2^{2\langle A\rangle + \langle b\rangle - 2n^2}$,

$$r_i \;=\; -\|a_i^\top\|_1 \qquad \Leftrightarrow \qquad b_i \;=\; \frac{K}{2}\Big(a_i^\top \mathbb{1} - \|a_i^\top\|_1\Big). \tag{3.27}$$

First, note that we can w.l.o.g. exclude the simultaneous occurrence of $b_i = 0$ and (3.27): In this case, the latter equality holds if and only if $a_i^\top \mathbb{1} = \|a_i^\top\|_1$, which implies, in particular, that $a_{ij} \ge 0$ for all $j \in [n]$. But then $a_i^\top x \le b_i = 0$ with $x \ge 0$ forces $x_j = 0$ for all j with $a_{ij} \ne 0$, in every feasible solution of (LP). Therefore, these variables and row i can be eliminated from the LP a priori.

Thus, suppose that (3.27) holds with $b_i \ne 0$ (which would imply $a_i^\top \ne 0$, but recall that this was already assumed throughout). The equality (3.27) is "fragile" with respect to scaling, since it depends on the encoding length of the data.

To exploit this, assume w.l.o.g. that $a_i^\top \in \mathbb{Z}^m$ and $b_i \in \mathbb{Z}$. This can always be achieved, in polynomial time, by scaling with the least common denominator L of the entries. Since such scaling may both reduce or increase the encoding lengths of entries a_{ij} and b_i, possibly $\langle a_i^\top\rangle + \langle b_i\rangle = \langle La_i^\top\rangle + \langle Lb_i\rangle$, in which case K remains the same and (3.27) still holds with $a_i^\top$ and b_i replaced by $La_i^\top$ and Lb_i, respectively.

Let $c \coloneqq \min\{\, |b_i|, \min\{\, |a_{ij}| \,:\, j \in [n], a_{ij} \ne 0 \,\}\,\}$ and $\alpha \coloneqq (2c+2)/c$ (note that $c \ge 1$ because b_i is a nonzero integer—and $a_i^\top$ contains at least one—and that consequently, $2 < \alpha \le 4$). If we scale the inequality $a_i^\top x \le b_i$ by α and update K accordingly, (3.27) can no longer hold with respect to the scaled data. To see this,

assume to the contrary that

$$\alpha b_i \;=\; 2^{2\langle A_{[m]\setminus\{i\},[n]}\rangle+2\langle\alpha a_i^\top\rangle+\langle b_{[m]\setminus\{i\}}\rangle+\langle\alpha b_i\rangle-2n^2-1}(\alpha a_i^\top \mathbb{1} - \|\alpha a_i^\top\|_1),$$

which is equivalent to

$$b_i \;=\; 2^{2\langle A_{[m]\setminus\{i\},[n]}\rangle+2\langle\alpha a_i^\top\rangle+\langle b_{[m]\setminus\{i\}}\rangle+\langle\alpha b_i\rangle-2n^2-1}(a_i^\top \mathbb{1} - \|a_i^\top\|_1).$$

Equating this with the identity for b_i from (3.27), we obtain (after cancelations)

$$2^{2\langle a_i^\top\rangle+\langle b_i\rangle} = 2^{2\langle\alpha a_i^\top\rangle+\langle\alpha b_i\rangle} \quad\Leftrightarrow\quad 2\langle a_i^\top\rangle + \langle b_i\rangle = 2\langle\alpha a_i^\top\rangle + \langle\alpha b_i\rangle. \tag{3.28}$$

However, this is a contradiction: Assume w.l.o.g. that $c = |b_i|$. Since $\alpha > 2$ and $a_i^\top \in \mathbb{Z}^n$, it holds that $\langle\alpha a_i^\top\rangle \geq \langle a_i^\top\rangle$. Moreover,

$$\alpha > \frac{2c+1}{c} \quad\Leftrightarrow\quad 2(c+1) < \alpha c + 1 \quad\Leftrightarrow\quad \log_2(c+1) + 1 < \log_2(\alpha c + 1),$$

and since we also have $\log_2(c+1)+1 \geq \lfloor\log_2(c+1)\rfloor + 1 \geq \lceil\log_2(c+1)\rceil$ and $\log_2(\alpha c+1) \leq \lceil\log_2(\alpha c+1)\rceil$, this implies that

$$\langle c\rangle \;=\; 1 + \lceil\log_2(c+1)\rceil \;<\; 1 + \lceil\log_2(\alpha c+1)\rceil \;=\; \langle\alpha c\rangle.$$

Therefore,

$$2\langle\alpha a_i^\top\rangle + \langle\alpha b_i\rangle \;\geq\; 2\langle a_i^\top\rangle + \langle\alpha c\rangle \;>\; 2\langle a_i^\top\rangle + \langle c\rangle \;=\; 2\langle a_i^\top\rangle + \langle b_i\rangle,$$

so indeed, (3.28) cannot hold. This shows that the case $r_i = -\|a_i^\top\|_1$ can always be avoided by appropriate scaling of the i-th constraint $a_i^\top x \leq b_i$.

Thus, we conclude that Assumption 3.16(iii) is in fact not needed for the proof of Theorem 3.17 and can be dropped without loss of generality. Moreover, note that replacing K by $\tilde{K} = \gamma K$ with $\gamma \geq 1$ does not change anything: The case $b_i = 0$ is independent of K, and γ cancels out in (3.28).

Now consider statement (i). Let $P := \{\, x : Ax \leq b,\ x \geq 0 \,\}$. If (LP) is infeasible (i.e., $P = \emptyset$), possibly $r_i < -\|a_i^\top\|_1$ for some i, which is easily detected during the construction of the (3.26) instance (equivalently, $b_i < \frac{K}{2}(a_i^\top \mathbb{1} - \|a_i^\top\|_1)$, which could be checked a priori). Thus, for the remainder of this paragraph, suppose "trivial" infeasibility $r_i < -\|a_i^\top\|_1$ does not occur, but still it holds that $P = \emptyset$. Note that then, (3.26) can always be successfully constructed and is, in fact, always feasible (cf. Remark 3.14). Since the preceding intermediate transformed problem (3.24) is infeasible if and only if (LP) is, it is clear that the feasibility of (3.26) is directly caused by introducing z and relaxing $z = 1$ (which yields direct correspondence

of (3.26) and (3.24)) to $z \in [-1, 1]$. Thus, we have $P = \emptyset$ if and only if the set of points $(y^\top, z)^\top$ that satisfy the constraints of (3.26) with $z = 1$ is empty. In particular, by construction of $\mathcal{M}$, if and only if every *optimal* solution $((y^*)^\top, z^*)^\top$ of (3.26) has $z^* < 1$, then the original (LP) is infeasible. Therefore, $P = \emptyset$ can be detected (a posteriori) by inspecting the optimal solution of (3.26), whenever (3.26) is constructible. It remains to note that clearly, using any $\tilde{K} \geq K$ instead of K does not change the above argumentation, which concludes the proof of statement (i).

Finally, we turn to statement (ii): Suppose we replaced K by some $\tilde{K} > K$ in the proof of Theorem 3.17, and that $P \neq \emptyset$. Then, if (LP) is bounded, $x_j \leq K$ holds *implicitly* for all vertices x (in particular, optimal solutions) by Lemma 3.15(iv), and therefore $x_j \leq \tilde{K}$ as well (in fact, $x_j < \tilde{K}$). On the other hand, if and only if $c^\top x$ can become arbitrarily large, then for at least one $j_* \in [n]$, x_{j_*} can be increased infinitely in (LP). Since adding the inequalities $x_j \leq \tilde{K}$, $j \in [n]$, makes the problem bounded, in any optimal solution $\tilde{x}$ of (3.19), $\tilde{x}_{j_*}$ will instead reach its upper bound, i.e., $\tilde{x}_{j_*} = \tilde{K}$. This can be expressed in terms of $y = (2/\tilde{K})x - \mathbb{1}$ as well: In an optimal solution $((y^*)^\top, 1)^\top$ of the transformed problem (3.26) we have $y_j^* = 2\tilde{K}/\tilde{K} - 1 = 1$, for some $j \in [n]$, if and only if the original problem (LP) is unbounded. Thus, like nontrivial infeasibility, unboundedness of (LP) can be detected a posteriori by inspecting the optimal solution of (3.26). □

3.5.4 Detailed Complexity Analysis

We now examine how the encoding length of the problem changes in the reduction from (LP) to a BP instance. The transformation is clearly problematic in practice since, in particular, $\mathcal{M}$ is extremely large (although its encoding length is polynomially related to that of the input instance). Nevertheless, a careful analysis can yield further insights into the relationship between the two problems, for instance, regarding asymptotical (worst-case) running time bounds.

For the reduction from Theorem 3.17, we obtain the following:

Theorem 3.20. *Let the linear program*

$$\max c^\top x \quad \text{s.t.} \quad Ax \leq b,\ x \geq 0 \tag{LP}$$

with $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$, satisfy Assumption 3.16 and let

$$\min \|x\|_1 \quad \text{s.t.} \quad \left(I, \begin{pmatrix} \tilde{A}^\top \\ \delta^\top \end{pmatrix}\right) x = \begin{pmatrix} c \\ \mathcal{M} \end{pmatrix} \tag{3.29}$$

be the Basis Pursuit problem obtained by transforming (LP) *as shown in the proof of Theorem 3.17 (cf. Lemma 3.11 and Remark 3.14). Then*

$$\langle\,(3.29)\,\rangle \in \mathcal{O}\big(nm(\langle A\rangle + \langle b\rangle) + \langle c\rangle + n^3 m\big) \subseteq \mathcal{O}\big(nm\langle\,(\mathrm{LP})\,\rangle + n^3 m\big). \tag{3.30}$$

Proof. The encoding length of the (LP) data is

$$\langle\,(\mathrm{LP})\,\rangle \;=\; \langle A\rangle + \langle b\rangle + \langle c\rangle + n\langle 0\rangle \;=\; \langle A\rangle + \langle b\rangle + \langle c\rangle + n.$$

Having transformed (LP) into (3.26), we obtain (3.29) by dualizing (3.26) and applying Lemma 3.11. Let $D^\top \coloneqq (\tilde{A}, \delta) \in \mathbb{Q}^{m\times(n+1)}$ and $d^\top \coloneqq (c^\top, \mathcal{M})$, and note that $I = I_{n+1}$. Therefore, we have

$$\begin{aligned}\langle\,(3.29)\,\rangle &= \langle (I, D)\rangle + \langle d\rangle \;=\; ((n+1)\langle 1\rangle + (n^2+n)\langle 0\rangle) + \langle\tilde{A}\rangle + \langle\delta\rangle + \langle c\rangle + \langle\mathcal{M}\rangle \\ &= \langle\tilde{A}\rangle + \langle\delta\rangle + \langle c\rangle + \langle\mathcal{M}\rangle + n^2 + 3n + 2. \end{aligned}\tag{3.31}$$

In the following, we derive bounds on the encoding lengths of $\tilde{A}$, δ and $\mathcal{M}$.

We start with $\tilde{A}$: Recall that

$$\langle\tilde{A}\rangle \;=\; \sum_{i=1}^{m}\langle\tilde{a}_i^\top\rangle,$$

and that, for any $i \in [m]$,

$$\tilde{a}_i^\top = \begin{cases} \frac{1}{\|a_i^\top\|_1}\, a_i^\top, & \text{if } \min\{r_i, \|a_i^\top\|_1\} = \|a_i^\top\|_1, \\ \frac{2}{\|a_i^\top\|_1 + r_i}\, a_i^\top, & \text{if } \min\{r_i, \|a_i^\top\|_1\} = r_i \notin \{\|a_i^\top\|_1, -\|a_i^\top\|_1\}. \end{cases}$$

For the first case we obtain

$$\begin{aligned}\langle\tilde{a}_i^\top\rangle \;&=\; \left\langle \frac{1}{\|a_i^\top\|_1}\, a_i^\top \right\rangle \overset{(3.12)}{\leq}\; n\left\langle \frac{1}{\|a_i^\top\|_1}\right\rangle + \langle a_i^\top\rangle \\ &\overset{(3.17)}{\leq}\; 2n + n\langle\|a_i^\top\|_1\rangle + \langle a_i^\top\rangle \overset{(3.18)}{\leq}\; (2n+1)\langle a_i^\top\rangle + 2n. \end{aligned}\tag{3.32}$$

Now consider the second case. For $r_i = \frac{2}{K} b_i - a_i^\top \mathbb{1}$, we derive

$$\begin{aligned}\langle r_i\rangle \;&=\; \left\langle \frac{2}{K} b_i - a_i^\top \mathbb{1}\right\rangle \;=\; \left\langle 2^{1+2n^2-2\langle A\rangle - \langle b\rangle} b_i - \sum_{j=1}^{n} a_{ij}\right\rangle \\ &\overset{(3.13),(3.11)}{\leq}\; 2\left(\langle 2\rangle + \left\langle 2^{2n^2}\right\rangle + \left\langle \frac{1}{2^{2\langle A\rangle}}\right\rangle + \left\langle \frac{1}{2^{\langle b\rangle}}\right\rangle + \langle b_i\rangle + \left\langle \sum_{j=1}^{n} a_{ij}\right\rangle\right) \end{aligned}$$

$$\overset{(3.13),(3.17)}{\leq} 2\left(3+\left\langle 2^{2n^2}\right\rangle+2+\left\langle 2^{2\langle A\rangle}\right\rangle+2+\left\langle 2^{\langle b\rangle}\right\rangle+\langle b_i\rangle+2\sum_{j=1}^{n}\langle a_{ij}\rangle\right)$$
$$\overset{(3.16)}{\leq} 14+2(2+2n^2)+2(2+2\langle A\rangle)+2(2+\langle b\rangle)+2\langle b_i\rangle+4\langle a_i^\top\rangle$$
$$= 4\langle A\rangle+2\langle b\rangle+4\langle a_i^\top\rangle+2\langle b_i\rangle+4n^2+26. \tag{3.33}$$

Thus, we obtain

$$\langle \tilde{a}_i^\top\rangle = \left\langle \frac{2}{\|a_i^\top\|_1+r_i}\,a_i^\top\right\rangle \overset{(3.12),(3.11)}{\leq} n\left(\langle 2\rangle+\left\langle\frac{1}{\|a_i^\top\|_1+r_i}\right\rangle\right)+\langle a_i^\top\rangle$$
$$\overset{(3.17)}{\leq} 3n+2n+n\langle\|a_i^\top\|_1+r_i\rangle+\langle a_i^\top\rangle$$
$$\overset{(3.13),(3.18)}{\leq} (4n+1)\langle a_i^\top\rangle+2n\langle r_i\rangle+5n$$
$$\overset{(3.33)}{\leq} (4n+1)\langle a_i^\top\rangle+5n+2n\left(4\langle A\rangle+2\langle b\rangle+4\langle a_i^\top\rangle+2\langle b_i\rangle+4n^2+26\right)$$
$$= 8n\langle A\rangle+4n\langle b\rangle+(12n+1)\langle a_i^\top\rangle+4n\langle b_i\rangle+8n^3+52n. \tag{3.34}$$

Clearly, the bound (3.34) is larger than the one from the first case, (3.32). Thus, we estimate the encoding length of $\tilde{A}$ as

$$\langle\tilde{A}\rangle \overset{(3.34)}{\leq} 8nm\langle A\rangle+4nm\langle b\rangle+(12n+1)\sum_{i=1}^{m}\langle a_i^\top\rangle+4n\sum_{i=1}^{m}\langle b_i\rangle+8n^3m+52nm$$
$$= (8nm+12n+1)\langle A\rangle+(4nm+4n)\langle b\rangle+8n^3m+52nm. \tag{3.35}$$

Moreover, we obtain (for all $i\in[m]$)

$$\langle\delta_i\rangle = \left\langle\frac{\|a_i^\top\|_1-r_i}{\|a_i^\top\|_1+r_i}\right\rangle \overset{(3.11)}{\leq} \langle\|a_i^\top\|_1-r_i\rangle+\left\langle\frac{1}{\|a_i^\top\|_1+r_i}\right\rangle$$
$$\overset{(3.13),(3.17)}{\leq} 2\langle\|a_i^\top\|_1\rangle+2\langle r_i\rangle+2\big(\langle\|a_i^\top\|_1\rangle+\langle r_i\rangle\big)+2 \overset{(3.18)}{\leq} 8\langle a_i^\top\rangle+4\langle r_i\rangle+2$$
$$\overset{(3.33)}{\leq} 16\langle A\rangle+8\langle b\rangle+24\langle a_i^\top\rangle+8\langle b_i\rangle+16n^2+106, \tag{3.36}$$

and therefore

$$\langle\delta\rangle = \sum_{i=1}^{m}\langle\delta_i\rangle \overset{(3.36)}{\leq} 16m\langle A\rangle+8m\langle b\rangle+24\sum_{i=1}^{m}\langle a_i^\top\rangle+8\sum_{i=1}^{m}\langle b_i\rangle+16n^2m+106m$$
$$= (16m+24)\langle A\rangle+(8m+8)\langle b\rangle+16n^2m+106m. \tag{3.37}$$

Finally, it remains to bound $\langle \mathcal{M} \rangle$. Recall that

$$\mathcal{M} \;\coloneqq\; 2^{4n^2\langle \hat{A}\rangle}\, n\, \|c\|_1 \max_{i\in[m]}\{\,\delta_i\,\} + 1,$$

where $\hat{A} = (\tilde{A}^\top, -\tilde{A}^\top, I_n, -I_n)^\top$. By (3.35), it holds that

$$\begin{aligned}\langle \hat{A} \rangle \;&=\; \langle \tilde{A}\rangle + \langle -\tilde{A}\rangle + \langle I_n\rangle + \langle -I_n\rangle \;=\; 2\langle \tilde{A}\rangle + 2\langle I_n\rangle \\ &\leq (16nm + 24n + 2)\langle A\rangle + (8nm + 8n)\langle b\rangle + 16n^3m + 2n^2 + 104nm + 2n.\end{aligned} \tag{3.38}$$

Thus, we obtain

$$\begin{aligned}
\langle \mathcal{M} \rangle \;&\leq\; \left\langle 2^{4n^2\langle \hat{A}\rangle}\, n\, \|c\|_1 \max_{i\in[m]}\{\,\delta_i\,\} + 1 \right\rangle \\
&\overset{(3.11),(3.13)}{\leq}\; 2\left(\left\langle 2^{4n^2}\right\rangle + \left\langle 2^{\langle \hat{A}\rangle}\right\rangle + \langle n\rangle + \langle \|c\|_1\rangle + \left\langle \max_{i\in[m]}\{\,\delta_i\,\}\right\rangle + \langle 1\rangle\right) \\
&\overset{(3.16),(3.18)}{\leq}\; 2\left(2 + 4n^2 + 2 + \langle \hat{A}\rangle + \langle n\rangle + 2\langle c\rangle + \langle \delta\rangle + 2\right) \\
&\overset{(3.16)}{\leq}\; 2\langle \hat{A}\rangle + 2\langle \delta\rangle + 4\langle c\rangle + 8n^2 + 6 + 2\log_2(n) + 12 \\
&\overset{(3.37),(3.38)}{\leq}\; (32nm + 48n + 4)\langle A\rangle + (16nm + 16n)\langle b\rangle \\
&\qquad + 32n^3m + 4n^2 + 208nm + 4n \\
&\qquad + (32m + 48)\langle A\rangle + (16m + 16)\langle b\rangle + 32n^2m + 212m \\
&\qquad + 4\langle c\rangle + 8n^2 + 2\log_2(n) + 18 \\
&=\; (32nm + 48n + 32m + 52)\langle A\rangle \\
&\qquad + (16nm + 16n + 16m + 16)\langle b\rangle + 4\langle c\rangle \\
&\qquad + 32n^3m + 32n^2m + 12n^2 + 208nm + 4n \\
&\qquad + 212m + 2\log_2(n) + 18.
\end{aligned} \tag{3.39}$$

With (3.35), (3.37) and (3.39), we obtain from (3.31) the following upper bound for the encoding length of the ℓ_1-minimization problem (3.29) obtained by transforming the linear program (LP):

$$\begin{aligned}\langle\, (3.29)\,\rangle \;&=\; \langle \tilde{A}\rangle + \langle \delta\rangle + \langle \mathcal{M}\rangle + \langle c\rangle + n^2 + 3n + 2 \\ &\leq\; (8nm + 12n + 1)\langle A\rangle + (4nm + 4n)\langle b\rangle + 8n^3m + 52nm \\ &\qquad + (16m + 24)\langle A\rangle + (8m + 8)\langle b\rangle + 16n^2m + 106m \\ &\qquad + (32nm + 48n + 32m + 52)\langle A\rangle + (16nm + 16n + 16m + 16)\langle b\rangle\end{aligned}$$

$$\begin{aligned}
&+ 4\langle c\rangle + 32n^3m + 32n^2m + 12n^2 + 208nm \\
&+ 4n + 212m + 2\log_2(n) + 18 + \langle c\rangle + n^2 + 3n + 2 \\
&= (40nm + 60n + 48m + 77)\langle A\rangle + (20nm + 20n + 24m + 24)\langle b\rangle + 5\langle c\rangle \\
&\quad + 40n^3m + 48n^2m + 13n^2 + 260nm + 7n + 2\log_2(n) + 318m + 20.
\end{aligned}$$

Thus, we obtain the claimed asymptotic bound (3.30):

$$\begin{aligned}
\langle\,(3.29)\,\rangle \;&\in\; \mathcal{O}\big(nm(\langle A\rangle + \langle b\rangle) + \langle c\rangle + n^3m\big) \\
&\subseteq\; \mathcal{O}\big(nm(\langle A\rangle + \langle b\rangle + \langle c\rangle + n) + n^3m\big) \;=\; \mathcal{O}\big(nm\langle\,(\mathrm{LP})\,\rangle + n^3m\big).
\end{aligned}$$

This concludes the proof. □

The $\mathcal{O}(n^3m)$ term in the above estimate cannot be dropped in general, because, for instance, if $\langle A\rangle, \langle b\rangle, \langle c\rangle \in \mathcal{O}(n^2)$ then $\mathcal{O}(nm\langle\,(\mathrm{LP})\,\rangle + n^3m) \subseteq \mathcal{O}(n^3m)$. This situation is indeed possible: Consider $A \in \{0, \pm 1\}^{m\times n}$ with just two nonzero entries per column (e.g., the incidence matrix of a directed graph), and let b and c be ternary as well. Then, $\langle b\rangle, \langle c\rangle \in \mathcal{O}(\langle A\rangle)$, and

$$\langle A\rangle \;=\; 2n\langle 1\rangle + (nm - 2n)\langle 0\rangle \;=\; nm + 2n \;\in\; \mathcal{O}(nm).$$

Thus, whenever $n \geq m$, $\langle A\rangle \in \mathcal{O}(n^2)$ and $nm\langle A\rangle \in \mathcal{O}(n^2m^2) \subseteq \mathcal{O}(n^3m)$.

On the other hand, if $\langle A\rangle + \langle b\rangle \in \Omega(n^2)$, then $n^3m \in \mathcal{O}(nm(\langle A\rangle + \langle b\rangle))$ and consequently,

$$\langle\,(3.29)\,\rangle \;\in\; \mathcal{O}\big(nm(\langle A\rangle + \langle b\rangle) + \langle c\rangle\big) \;\subseteq\; \mathcal{O}\big(nm\langle\,(\mathrm{LP})\,\rangle\big). \tag{3.40}$$

Recall that a linear program can be solved in $\mathcal{O}(n^3\langle\,(\mathrm{LP})\,\rangle)$ arithmetic operations[5] by interior-point methods, see, e.g., [122, 244, 260, 26] (assuming $m \leq n$). Therefore, to improve this asymptotic bound by means of our reduction, an ℓ_1-solver would be required to need less than $\mathcal{O}((n^2/m)\langle\,(3.29)\,\rangle)$ arithmetic operations (assuming that the running time of constructing the instance (3.29) from (LP) is insignificant in comparison, which holds true under Assumption 3.16 at least).

Typically, (first-order) ℓ_1-solvers rely on matrix-vector multiplication (at least one per iteration), whose cost dominates the overall iteration complexity. With the $(n+1)\times(n+m+1)$ matrix from the instance (3.29), one such multiplication requires

[5] The bound $\mathcal{O}(n^3\langle\,(\mathrm{LP})\,\rangle)$ pertains to reaching an approximately optimal solution, using $\mathcal{O}(\langle\,(\mathrm{LP})\,\rangle)$-bit precision arithmetic, that is sufficiently close to an exact optimum to allow crossing over to an optimal basic solution (see, e.g., [187, 30] or [244, Section 9]). It holds if the LP data is *integer*, which, given rational data, can be assumed w.l.o.g. (otherwise, appropriate scaling achieves it); thus, in the discussion to follow, we implicitly make this assumption (as well as $m \leq n$, cf. Remark 3.21).

$(n+1)m+n+1 \in \mathcal{O}(nm)$ elementary operations involving matrix and vector entries (note that multiplication with I only needs $n+1$ operations instead of $(n+1)^2$ for general, fully dense $(n+1) \times (n+1)$ matrices). Thus, let us assume the iteration complexity of an ℓ_1-solver (applied to (3.29)) to be bounded generally as $\mathcal{O}(nm)$. Consequently, the general bound $\mathcal{O}(n^3\langle\,(\text{LP})\,\rangle)$ for linear programming could be improved (based on (3.40)) if the applied ℓ_1-solver took only $\mathcal{O}(((n/m^2)\langle\,(3.29)\,\rangle)^{1-\epsilon})$ iterations[6] to solve (3.29), for some constant $\epsilon \in (0,1]$:

$$\mathcal{O}\Big(\big(\tfrac{n}{m^2}\langle\,(3.29)\,\rangle\big)^{1-\epsilon} \cdot \mathcal{O}(nm)\Big) \;\subseteq\; \mathcal{O}\Big(nm\big(\tfrac{n^2}{m}\langle\,(\text{LP})\,\rangle\big)^{1-\epsilon}\Big) \;\subset\; \mathcal{O}\big(n^3\langle\,(\text{LP})\,\rangle\big).$$

To the best of our knowledge, no algorithms for (P_1) are known that meet these specific requirements in general. Regarding, in particular, the ℓ_1-solvers listed in Section 3.2, the Homotopy method disqualifies due to its exponential worst-case complexity, and ℓ_1-Magic and SolveBP/PDCO are themselves interior-point methods applied to LP reformulations of (P_1) (admittedly, this specialization might allow to refine the respective complexity analyses w.r.t. the general case; we are, however, not aware of any detailed investigation of this aspect). For SPGL1, YALL1 and ISAL1, convergence to the optimum is known. Note that, since (3.29) can itself be written as an LP, one does not need to perform an infinite number of iterations when applying these methods. Instead, similarly to interior-point LP methods, we could terminate once a sufficiently accurate solution is reached, and then cross-over to an optimal basic solution (provided dual information is available, cf. [187, 30]). However, it is not clear from the existing convergence results whether the precision required for cross-over could be reached within $\mathcal{O}(((n/m^2)\langle\,(3.29)\,\rangle)^{1-\epsilon})$ iterations by any of the aforementioned algorithms.

A method with known estimates on the iteration number required to reach a certain accuracy is FISTA [18]. Although not directly applicable to (P_1), FISTA can solve problem (QP_λ), i.e., $\min \frac{1}{2}\|Ax-b\|_2^2 + \lambda\|x\|_1$, whose optimal solution x_λ^* converges to that of (P_1) as $\lambda \to 0$ (cf. Section 3.2.2). However, no a priori upper bound on λ is known that guarantees a prescribed precision (w.r.t. solving BP), see [245, 113].

Furthermore, recall that the Homotopy method sometimes exhibits the "k-step solution property", i.e., it needs only k iterations if the (unique) optimal Basis Pursuit solution has k nonzero entries and k is sufficiently small. If this was the case for the instance (3.29), then (LP) could be solved in $\mathcal{O}(nmk)$ time (plus that needed for the reduction) by applying the Homotopy method to (3.29), cf. [93]. Unfortunately, this situation can also hardly be guaranteed a priori, since neither

[6] Clearly, it would suffice to have $\mathcal{O}(n^{1-\epsilon_1}m^{-2-\epsilon_2}\langle\,(3.29)\,\rangle^{1-\epsilon_3})$ with at least one $\epsilon_i > 0$. For simplicity, we give the argument using just one $\epsilon \in (0,1]$ for all terms, and also neglect other subtleties such as possible finite-precision arithmetic requirements.

does one usually have prior knowledge of the solution sparsity for BP problems, nor is it clear what properties of the (LP) solution or instance a certain sparsity of solutions of (3.29) possibly translates into.

Thus, to summarize the above discussion, the reduction from a linear program to a Basis Pursuit instance seems unlikely to lead to better asymptotical running time bounds for linear programming in general. Nevertheless, it shows that, in principle, one could indeed solve arbitrary LPs with any BP-solver.

Remark 3.21. The interior-point methods from, e.g., [122, 244, 260], require the original LP to be in standard equality form $\min\{\, c^\top x \,:\, Ax = b,\, x \geq 0 \,\}$, for which assuming $m \leq n$ does not lead to a loss of generality (additional equations can easily be eliminated, or shown to induce infeasibility, a priori, via Gaussian elimination, cf. [128]). Therefore, a running time bound like $\mathcal{O}(((m+n)n^2 + (m+n)^{1.5}n))L)$ (see [244]) is usually shortened to $\mathcal{O}(n^3 L)$, where L is the encoding length of the LP in equality form, not the inequality form (LP). Clearly, it holds that $\mathcal{O}(L) = \mathcal{O}(\langle\langle \text{(LP)} \rangle\rangle)$ for the equivalent equality and inequality forms of any linear program.

However, if we are given an LP in equality form, the corresponding inequality form (LP) can of course have more rows than columns, since each equality is turned into a pair of inequalities; generally, we could then end up with $n \leq m \leq 2n$. We do not give a detailed discussion for this case, since arguments completely analogous to those above can be derived, along the same lines, incorporating estimates for the general linear programming running time that involve both m and n.

Remark 3.22. Due to the strong duality between (P_1) and ($\text{D}_1^\dagger$) (cf. Remark 3.14 and Lemma 3.11), the optimal objective function values opt_{BP} and opt_{LP} of (3.29) and (LP), respectively, are related as

$$c^\top x^* \;=\; \text{opt}_{LP} \;=\; \tfrac{K}{2}(\text{opt}_{BP} - \mathcal{M} + c^\top \mathbb{1}).$$

In contrast, an optimal *solution* x^* for (LP) itself can, in general, not be obtained directly (in closed form) from an optimal solution η^* for (3.29). This is because our reduction involves dualization, and so we first need to construct from η^* an optimal *dual* solution $((y^*)^\top, 1)^\top$, i.e., one for (3.26). Then, $x^* = \frac{K}{2}(y^* + \mathbb{1})$ optimally solves (LP), see the beginning of the proof of Theorem 3.17.

In some situations, $y^* = -\big((I,\, D)_{S^*}^\top\big)^\dagger \operatorname{sign}(\eta^*_{S^*})$, where $S^* = \operatorname{supp}(\eta^*)$; e.g., if η^* is sufficiently sparse (cf. Section 3.1.3). Also, many ℓ_1-solvers are of a primal-dual type, i.e., they also (perhaps implicitly) compute a dual optimal solution as well; examples are SPGL1 and YALL1, cf. Sections 3.2.5 and 3.2.6, respectively.

In general, however, from a theoretical LP-perspective on (P_1) and ($\text{D}_1^\dagger$), computing y^* from a given primal-optimal solution η^* alone may essentially be as hard

as solving the problem (3.29) (written as an LP) from scratch, cf. [187]. Although this shows that (LP) and (P_1) are polynomially equivalent also in the sense that respective optimal solutions (not only objective values) can be obtained from each other, it seems undesirable, given that we originally started with an LP in the first place. Nevertheless, in practice (or average-case scenarios), one can still expect the knowledge of η^* and the corresponding optimal objective value to allow for obtaining a dual optimal solution y^* more efficiently than by a complete resolve of the problem. For instance, an LP cross-over method relying only on an (approximate) primal-optimal solution is described and demonstrated to be practically efficient in [30]. Moreover, one could employ any algorithm for convex feasibility problems (see, e.g., [16]) to tackle the problem of finding a point satisfying the dual constraints together with the prescribed (optimal) objective function value.

3.6 Excursion into Basis Pursuit Denoising

So far, we have solely investigated Basis Pursuit in the noisefree case, i.e., (P_1). In the following, we will instead consider the (usually more realistic) situation in which the measurements are contaminated by noise and the pure BP model becomes arguably less useful. In particular, motivated by the success of the heuristic optimality check (HOC) in improving solution speed and quality of BP-solvers, we will focus on HOC variants for the noise-aware ℓ_1-minimization problems

$$\min \|x\|_1 \quad \text{s.t.} \quad \|Ax - b\|_2 \leq \delta \tag{P_1^δ}$$

and

$$\min \frac{1}{2}\|Ax - b\|_2^2 + \lambda \|x\|_1. \tag{QP_λ}$$

Throughout this section, we will assume that $\|b\|_2 > \delta$, which rules out the trivial all-zero solution to (P_1^δ). Moreover, we shall assume that in an optimal solution x^*, the support S^* of x^* yields a full-rank submatrix A_{S^*}, i.e., $\operatorname{rank}(A_{S^*}) = |S^*| \leq m$. (Note that—at least in a sparse recovery context—this rank assumption is not particularly restrictive, since it is part of necessary and sufficient conditions for uniqueness of optimal solutions of many ℓ_1-minimization problems, including (P_1^δ) and (QP_λ); see [265].)

The HOC code for (P_1^δ) and (QP_λ) is available from the same webpage as the one for (P_1) (i.e., `http://wwwopt.mathematik.tu-darmstadt.de/spear`).

3.6.1 HOC for BP Denoising

Let us start with the standard constrained form of Basis Pursuit Denoising, i.e., (P_1^δ). Emulating the derivation of HOC for (P_1) (cf. Section 3.1), we first regard the optimality conditions for (P_1^δ), which we gather in the following lemma. (This result is well-known—see, e.g., [173]—so we only roughly sketch the proof.)

Lemma 3.23. *A point $x^* \neq 0$ is optimal for the Basis Pursuit Denoising problem (P_1^δ) if and only if $\|Ax^* - b\|_2 = \delta$ and there exists some $\mu^* > 0$ such that $-\mu^* A^\top (Ax^* - b) \in \partial \|x^*\|_1$.*

Proof. Clearly, $x^* \neq 0$ can only be optimal if $\|b\|_2 > \delta$, and then, in particular, x^* lies on the boundary of the feasible set $X_\delta := \{\, x : \|Ax - b\|_2 \leq \delta \,\}$, i.e., $\|Ax^* - b\|_2 = \delta$. The latter follows from the complementarity condition $\mu^*(\|Ax^* - b\|_2 - \delta) = 0$ (cf., e.g., [221, Theorem 3.34]), since $\mu^* > 0$, as shown below. The second condition can be obtained by computing the normal cone to X_δ at x^* and applying Lemma 1.4, or by differentiating the Lagrangean function associated with (P_1^δ); $\mu^* \geq 0$ is a requirement (since μ^* is the Lagrange multiplier linked to an inequality constraint) and it is easily seen that whenever $x^* \neq 0$, $0 \notin \partial \|x^*\|_1$ and therefore, $\mu^* > 0$. □

Moreover, recall from Lemma 1.5 that the dual problem of (P_1^δ) is given by

$$\max\ -b^\top y - \delta \|y\|_2 \quad \text{s.t.} \quad \|A^\top y\|_\infty \leq 1. \tag{D_1^δ}$$

Lemma 3.24. *To an optimal primal solution $x^* \neq 0$ of (P_1^δ) and the corresponding Lagrange multiplier μ^*, a dual optimal solution is found as $y^* = \mu^*(Ax^* - b)$.*

Proof. Since $-\mu^* A^\top (Ax^* - b) \in \partial \|x^*\|_1$ (by Lemma 3.23) and $x^* \neq 0$, it holds that $\|A^\top y^*\|_\infty = \|\mu^* A^\top (Ax^* - b)\|_\infty = 1$. Thus, y^* is dual feasible. Moreover, we have $\|x^*\|_1 + b^\top y^* + \delta \|y^*\|_2 = -\mu^*(Ax^* - b)^\top Ax^* + \mu^*(Ax^* - b)^\top b + \mu^* \delta \|Ax^* - b\|_2 = -\mu^* \|Ax^* - b\|_2^2 + \mu^* \delta^2 = 0$, i.e., the duality gap vanishes, which shows that y^* is optimal. □

The general HOC idea is to construct a primal-dual optimal pair from a given candidate solution x. For (P_1^δ), we need to derive the Lagrange multiplier as well in order to relate primal and dual solutions (via Lemma 3.24).

To that end, note that from Lemma 3.23, an explicit expression for μ^* is immediate: Since $x^* \neq 0$, $\|h\|_\infty = 1$ for any subgradient $h \in \partial \|x^*\|_1$. Thus,

$$\mu^* \;=\; \frac{1}{\|A^\top (Ax^* - b)\|_\infty}. \tag{3.41}$$

However, it should be noted that (3.41) incorporates the actual values of x^*, whereas HOC for (P_1) uses only support and sign information, making the approach robust to deviations in the vector components. Fortunately, we can devise such a "magnitude-independent" HOC scheme for (P_1^δ) as well:

Note that, for an arbitrary (fixed) optimal solution x^* of (P_1^δ), this problem can be equivalently rewritten as the reduced-dimensional problem

$$\min \|z\|_1 \quad \text{s.t.} \quad \|A_{S^*} z - b\|_2 \leq \delta,\ \operatorname{sign}(z) = \operatorname{sign}(x^*_{S^*}), \tag{3.42}$$

Clearly, the optimal solution to this problem is $z^* = x^*_{S^*}$. In particular, since $\operatorname{sign}(z)$ is fixed, $\partial\|z\|_1 = \{\operatorname{sign}(x^*_{S^*})\}$ for any feasible point z of (3.42), and the optimality conditions reduce to

$$-\mu_z^* A_{S^*}^\top (A_{S^*} z^* - b) = \operatorname{sign}(x^*_{S^*}), \quad \|A_{S^*} z^* - b\|_2 = \delta, \quad \text{and} \quad \mu_z^* \geq 0.$$

Since $\operatorname{rank}(A_{S^*}) = |S^*| \leq m$ (by assumption), the first condition yields

$$z^* \;=\; \left(A_{S^*}^\top A_{S^*}\right)^{-1} \left(A_{S^*}^\top b - \frac{1}{\mu_z^*} \operatorname{sign}(x^*_{S^*}) \right).$$

Plugging this into the second condition, some basic calculations (together with the requirement $\mu_z^* \geq 0$) reveal that

$$\mu_z^* \;=\; \sqrt{\frac{\operatorname{sign}(x^*_{S^*})^\top \left(A_{S^*}^\top A_{S^*}\right)^{-1} \operatorname{sign}(x^*_{S^*})}{b^\top A_{S^*} \left(A_{S^*}^\top A_{S^*}\right)^{-1} A_{S^*}^\top b - \|b\|_2^2 + \delta^2}} \;>\; 0.$$

This expression does indeed not contain the actual entries of $x^*_{S^*}$ anymore. Since $A_{S^*}^\top A_{S^*}$ is symmetric positive definite (under our rank assumption), we can avoid building $A_{S^*}^\top A_{S^*}$ (and its inverse) explicitly by computing the unique vectors p^* and q^* such that $A_{S^*}^\top A_{S^*} p^* = \operatorname{sign}(x^*_{S^*})$ and $A_{S^*}^\top A_{S^*} q^* = A_{S^*}^\top b$ by iterative methods like CG, and then μ_z^* as

$$\mu_z^* \;=\; \sqrt{\frac{\operatorname{sign}(x^*_{S^*})^\top p^*}{b^\top A_{S^*} q^* - \|b\|_2^2 + \delta^2}}.$$

In particular, note that $z^* = q^* - (1/\mu_z^*) p^*$.

Replacing x^* and S^* by respective candidates x and S yields our HOC for (P_1^δ), which we summarize in Algorithm 3.3.

The derivation above immediately yields the following result concerning HOC success.

Algorithm 3.3 HEURISTIC OPTIMALITY CHECK (HOC) for (P_1^δ)

Input: matrix A, noise estimate $\delta > 0$, measurement vector b ($\|b\|_2 > \delta$), vector x

1: deduce candidate (approx.) support S from x
2: compute approximate solution $\hat{p}$ to $A_S^\top A_S\, p = \operatorname{sign}(x_S)$
3: compute approximate solution $\hat{q}$ to $A_S^\top A_S\, q = A_S^\top b$
4: define Lagrange multiplier estimate

$$\hat{\mu} \;:=\; \sqrt{\frac{\operatorname{sign}(x_S)^\top \hat{p}}{b^\top A_S\, \hat{q} - \|b\|_2^2 + \delta^2}}$$

5: define primal solution $\hat{x}$ with $\hat{x}_{S^c} := 0$ and $\hat{x}_S := \hat{q} - \frac{1}{\hat{\mu}}\hat{p}$
6: **if** $\|A\hat{x} - b\|_2 \approx \delta$ **then**
7: define dual solution $\hat{y} := \hat{\mu}(A\hat{x} - b)$
8: **if** $\|A^\top \hat{y}\|_\infty \approx 1$ **and** $(\|\hat{x}\|_1 + b^\top \hat{y} + \delta\|\hat{y}\|_2)/\|\hat{x}\|_1 \approx 0$ **then**
9: return "success"

Theorem 3.25. *Let $x^* \neq 0$ denote an optimal solution of (P_1^δ) with support S^* such that $\operatorname{rank}(A_{S^*}) = |S^*|$, and let a candidate solution $x \in \mathbb{R}^n$ be given. Then, under exact arithmetic, HOC (Algorithm 3.3) with input (A, b, δ, x) returns "success" with $\hat{x} = x^*$ if $S = S^*$ and $\operatorname{sign}(x_S) = \operatorname{sign}(x^*_{S^*})$.*

In our implementation of Algorithm 3.3, we use the CG method to approximately solve the equation systems involving $A_S^\top A_S$; we allow a maximum of 25 CG iterations. Moreover, the tolerances for comparisons (Steps 6 and 8) are set to 10^{-3} for primal and dual feasibility, and to 10^{-6} for the relative duality gap evaluation. We allow more flexibility with respect to feasibility because otherwise, the inaccuracy induced by $\hat{p}$ and $\hat{q}$ may apparently prevent correct optimality detection. (Moreover, note that in CS practice, δ is typically only an educated guess at the true noise level, so one might argue that primal feasibility needs not be handled very strictly—especially if a slightly larger violation allows for a sparser solution.) This effect can be alleviated to some degree by performing more CG iterations to compute $\hat{p}$ and, in particular, $\hat{q}$. (The rather quick CG convergence for systems with sign-vector right hand sides observed in case of the original BP-HOC applies to the computation of $\hat{p}$ as well.) However, more CG iterations naturally lead to a larger overhead. Thus, one has to bear in mind a certain trade-off between the (possible) extra running time induced by integrating HOC and the reliability of HOC success claims. The given settings seem to achieve an agreeable balance in this regard, but can possibly be improved.

In fact, the inexactness of computations (in combination with larger tolerances) apparently also makes false-positive "success" declarations possible, in cases in which

$S \subset S^*$ or if S contains most, but not all, indices from S^*, and some from $(S^*)^c$. However, judging from the computational experiments we will discuss below, this is a rare occurrence (with the above-described settings). When $S \supset S^*$, we observed that HOC also sometimes claims success—in such cases, one can usually obtain S^* exactly by an additional thresholding sweep of $\hat{x}$; empirically, the threshold $10^{-6}\,\|\hat{x}\|_1$ seems to work quite well in this respect.

3.6.2 HOC for ℓ_1-Regularized Least-Squares

The problem that was originally called Basis Pursuit Denoising (cf. [58]) reads

$$\min \frac{1}{2}\|Ax-b\|_2^2 + \lambda\,\|x\|_1, \tag{QP_λ}$$

and is often preferred because it is generally easier to solve than the constrained problem (P_1^δ). Another common name for (QP_λ) is essentially a description of its objective: ℓ_1-regularized least-squares; we will use this denomination to distinguish it from (P_1^δ).

The (necessary and sufficient) first-order optimality condition for (QP_λ) looks very similar to that of (P_1^δ) (see, e.g., [173]):

$$-\frac{1}{\lambda}\,A^\top(Ax^*-b) \;\in\; \partial\|x^*\|_1.$$

Note that this actually reveals the implicit relationship between the two problem parameters that must hold for the respective solutions of (QP_λ) and (P_1^δ) to be identical: λ then is the reciprocal value of the optimal Lagrange multiplier μ^* associated with (P_1^δ), which of course depends on δ; see also [245, 256, 173].

From the optimality condition, we have, in particular, that for an optimal solution x^* with support S^*, it holds that

$$(A^\top(Ax^*-b))_{S^*} = A_{S^*}^\top(A_{S^*}x^*_{S^*}-b) = -\lambda\,\mathrm{sign}(x^*_{S^*})$$

and $\|A^\top(Ax^*-b)\|_\infty \leq \lambda$. Moreover, we have the following result (see also, e.g., [259]).

Lemma 3.26. *The dual problem of* (QP_λ) *is*

$$\max -b^\top y - \frac{1}{2}\|y\|_2^2 \quad \text{s.t.} \quad \|A^\top y\|_\infty \leq \lambda,$$

and if x^* is a primal optimal solution, then $y^* = Ax^* - b$ is a dual optimal solution.

Algorithm 3.4 Heuristic Optimality Check (HOC) for (QP_λ)

Input: matrix A, regularization parameter $\lambda > 0$, measurement vector b, vector x

1: deduce candidate (approx.) support S from x
2: define primal solution $\hat{x}$ with $\hat{x}_{S^c} := 0$ and $\hat{x}_S$ an approximate solution to $A_S^\top A_S\, z = A_S^\top b - \lambda \operatorname{sign}(x_S)$.
3: define dual solution $\hat{y} := (A\hat{x} - b)$
4: **if** $\|A^\top \hat{y}\|_\infty \approx \lambda$ **and** $(\|\hat{x}\|_1 + b^\top \hat{y} + \frac{1}{2}\|\hat{y}\|_2^2)/\|\hat{x}\|_1 \approx 0$ **then**
5: return "success"

Proof. We obtain the dual problem via Fenchel-Rockafellar duality (see Lemma 1.3), noting that the conjugate functions of $f(x) = \lambda\|x\|_1$ and $g(x) = \frac{1}{2}\|x - b\|_2^2$ are $f^*(y) = \iota_{\{\, z \,:\, \|z\|_\infty \le \lambda \,\}}(y)$ and $g^*(y) = b^\top y + \frac{1}{2}\|y\|_2^2$, respectively. Let x^* be a (primal-) optimal solution of (QP_λ), and set $y^* := Ax^* - b$. Then, completely analogously to the proof of Lemma 3.24, one can show dual feasibility of y^* and that the duality gap vanishes. □

The fact that λ is a *given* part of any (QP_λ) instance simplifies the HOC procedure, which otherwise is analogous to that for (P_1^δ) (where we first have to compute the associated Lagrange multiplier). Furthermore, (QP_λ) is unconstrained, so we do not need to concern ourselves with primal feasibility. We summarize the approach in Algorithm 3.4.

In our implementation, we again employ at most 25 CG iterations to determine $\hat{x}_S$, and use tolerances 10^{-3} for (dual) feasibility and 10^{-6} for the duality gap check. (The remarks we made about HOC for (P_1^δ) regarding the trade-off between the accuracy of the approximate solution computed by the CG method and the possible overhead induced by HOC—in combination with tolerance settings and possible false-positive "success" declarations—apply similarly to Algorithm 3.4.)

Moreover, we obtain the following basic theoretical success guarantee.

Theorem 3.27. *Let $x^* \neq 0$ denote the optimal solution of (QP_λ) with support S^* such that $\operatorname{rank}(A_{S^*}) = |S^*| \le m$, and let a candidate solution $x \in \mathbb{R}^n$ be given. Then, under exact arithmetic, HOC (Algorithm 3.4) with input (A, b, λ, x) returns "success" with $\hat{x} = x^*$ if $S = S^*$ and $\operatorname{sign}(x_S) = \operatorname{sign}(x_S^*)$.*

Remark 3.28. It is noteworthy that HOC for (QP_λ) bears a strong resemblance to the semismooth Newton method proposed in [127]. Here, we try to prove optimality of the solution of the equation system in Step 2 via duality arguments, whereas in [127, Algorithm 2], this solution is used to compute a new support estimate (or "active set") for the next iteration. The semismooth Newton method is locally superlinearly convergent and the experiments in [127] show that a sudden drop in

the error occurs at some iteration, indicating (near-)optimality of the current active set. This "jump" to a high accuracy solution is precisely what we hope to achieve by using HOC and can indeed be explained along the same lines as our derivation.

The main drawback of the semismooth Newton algorithm is that convergence is only guaranteed if the starting point is sufficiently close to the optimum, which can be hard to ensure a priori However, the method can be embedded in a globally convergent algorithm by multidimensional filter techniques, see [188] for details.

3.6.3 Numerical Experiments

In the following, we demonstrate HOC in the setting of the denoising problems (P_1^δ) and (QP_λ) with a few computational experiments.

There exists a broad variety of algorithms and implementations for (QP_λ) and (although fewer) for (P_1^δ). Here, for simplicity, we will only consider solver packages that already participated in our solver comparison for pure Basis Pursuit, (P_1).

Note that SoPlex is a pure LP solver, and while CPLEX can handle quadratic constraints, our wrapper code currently does not support this feature; therefore, we exclude these solvers here. SolveBP/PDCO and ℓ_1-Homotopy can solve the regularized version (QP_λ), whereas (P_1^δ) can be solved by ℓ_1-Magic, SPGL1 and YALL1; moreover, our own solver ISAL1 can, in principle, be adapted to handle (P_1^δ) as well (details will be given in Section 4.6, see also Section 4.5.3).

In the following, we will disregard YALL1 due to its unsatisfactory performance for pure BP and because a few tentative tests for the denoising variant were similarly disappointing. Moreover, the ISAL1 code for (P_1^δ) is presently still at a prototypical stage: The many algorithmic parameters of ISAL1 were tuned for solving (P_1) via extensive benchmarking—such benchmark tests (in particular, also for the parameters that control the approximate projections) have yet to be performed for the denoising variant in order to achieve some degree of competitiveness. Since here, we do not wish to rigorously compare solvers for BP Denoising but to illustrate HOC in this context, we save this effort for the future, and instead impose some restrictions on our experiments with ISAL1; we will remark more on this aspect later.

3.6.3.1 Experimental Setup

We reused the matrices A and solutions x^* from our BP test set (see Section 3.3) to construct—with L1TestPack, cf. [173]—new right hand side vectors b such that each x^* is the unique optimum for the corresponding (P_1^δ) instance with $\delta = \|Ax^* - b\|_2$. In fact, L1TestPack actually constructs instances for (QP_λ) for

a given λ value, and δ can then be derived as the residual norm w.r.t. the solution x^*. Therefore, conveniently, we can use the same instances also for (QP_λ). Since the construction routines from L1TestPack take an unreasonably long time for the larger instances (for A with 2048 or more rows), we excluded these, which leaves 444 data sets (A, x^*) to begin with.

We constructed two sets of 444 instances each, using $\lambda = 0.01$ for the first and $\lambda = 10$ for the second set. This implies that the values of δ are smaller in the first part (about 0.07 on average, min/max 0.025/0.215, median 0.06) and larger in the second (average 71.09, min/max 24.47/215.03, median 60.49). Thus, in total, we obtain 888 instances (A, b, δ, λ) with known unique (and exactly sparse) optima x^* of (P_1^δ) and (QP_λ), respectively. Recalling the way we originally chose the x^* vectors, note that half of each part with small and large λ exhibits high and low dynamic ranges of the solutions. (However, clearly, the two sets are not independent, since we started the construction from the same original data (A, x^*) and then simply used different λ values.)

As we did for the noiseless setup, we integrated the suitable denoising version of HOC into each solver and assess its potential by comparing results without and with HOC on the given test set. For the interior-point methods ℓ_1-Magic and SolveBP/PDCO, HOC is again executed in every iteration. For ℓ_1-Homotopy and SPGL1, we tried different frequencies for the HOC calls, testing the same intervals as for BP (see Section 3.1.2, page 63); the findings of these tests are reported below. In ISAL1 we kept the default HOC frequency $R = \lfloor m/100 \rfloor$ from the code version for (P_1).

The support estimations for HOC are performed in the same fashion as before, see Section 3.1.2, with one exception: In SPGL1, the active set indices that worked well as approximate supports for (P_1) did not seem to be a sensible choice w.r.t. HOC for (P_1^δ); thus, we estimate the supports by the same hard-thresholding strategy as in ℓ_1-Magic, namely $S \coloneqq \{\, j \in [n] \,:\, |x_j| > 10^{-6}\, \|x\|_1 \,\}$.

Moreover, we again maintain a "black box" approach and only manually set required parameters, leaving all optional ones (in particular, the various tolerance parameters) at their default values. We analyze the computational results similarly to our rigorous solver comparison for (P_1). However, in the denoising context, it would appear somewhat unnatural to apply equally strict high-accuracy standards for feasibility and solution quality as before. Thus, we drop the previous distinction between "solved" and "acceptable" solution, and will be content with any solution $\overline{x}$ that reaches the known optimum to within an ℓ_2-norm (absolute) distance of 0.1 (which corresponds to the upper accuracy bound for solutions to qualify as acceptable in Section 3.4). The main goal of HOC should hence be improving the running time while (at least) achieving an accuracy comparable to what the solver would

reach by itself. (As mentioned earlier, HOC behavior can be steered towards this aim by adequately tuning the CG accuracy and tolerance parameters.)

3.6.3.2 BP Denoising

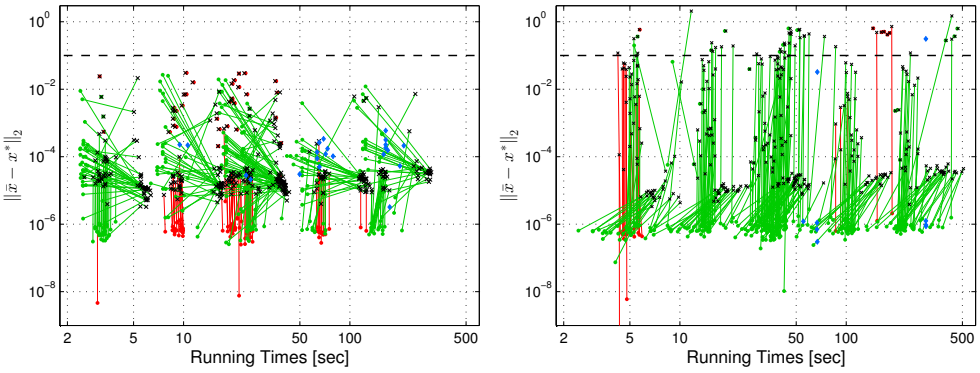
We start with the Basis Pursuit Denoising problem (P_1^δ), for which we tested HOC (Algorithm 3.3) integrated into SPGL1, ℓ_1-Magic, and ISAL1. The corresponding results of our experiments are depicted in Figures 3.13, 3.14 and 3.15, respectively (all running times are geometric means over 3 runs, or 10 runs if the codes were very fast). We will mostly let these pictures guide our tentative analysis of the influence of HOC; some supporting statistics are summarized in Table 3.8. In particular, note that the number of instances used in the experiments are different for each solver (for reasons we give below) and that the figures have different time and accuracy scales. Therefore, the results for the three solvers are not directly comparable to each other, and we focus on the individual effect of integrating HOC.

Starting with ℓ_1-Magic, we first remark (again) that the implementation struggles with numerical issues, cf. Remark 3.9: Like on the (P_1) instances, the algorithm quite often terminates due to errors connected to (at least numerically) incorrect settings w.r.t. symmetry and positive definiteness for MATLAB's `linsolve` function. This issue concerned 62 of the instances constructed with $\lambda = 0.01$ (smaller δ values) and 104 of those with $\lambda = 10$ (and larger δ values), i.e., about 14% and 23% of the respective groups of 444 instances. Here, we kept the original `linsolve` settings and exclude the instances on which ℓ_1-Magic failed. It is however noteworthy that with HOC, 22 more small-δ instances and 9 more with large δ value were solved, i.e., HOC led to early termination (with good approximations of the true solution) before ℓ_1-Magic "crashed".

On the remaining instances, the large number of green lines in Figure 3.13 suggests that HOC success quite often leads to early termination. In particular, note that a significant amount of the green lines has a clear "horizontal component", indicating relevant runtime improvements. Moreover, the red lines all appear almost vertical, which represents small to statistically insignificant overheads. Thus, Figure 3.13 shows that HOC is useful for ℓ_1-Magic. Indeed, the data confirms this impression: On the 382 instances with smaller δ values, integrating HOC into ℓ_1-Magic led to an average relative speed-up of 25.29% (over all instances; the algorithm became faster on 268 instances, on which the average speed-up is actually 36.59%; the mean overhead on the remaining instances is 1.26%). For the 340 larger-δ instances, HOC yields a total speed-up of 17.58%, the version with HOC being faster on 307 instances (average relative speed-up on these: 19.57%) and introducing an average overhead of 0.93% on the others.

Table 3.8. Impact of HOC for (P_1^δ) on the runtimes of ℓ_1-Magic, ISAL1 and SPGL1.

		# instances		# faster		avg. speed-up	
Solver	R	small δ	large δ	small δ	large δ	small δ	large δ
ℓ_1-Magic	1	382	340	268	307	25.29%	17.58%
ISAL1	$\lfloor m/100 \rfloor$	222	222	197	95	41.92%	13.85%
SPGL1	$\lfloor m/20 \rfloor$	444	444	155	33	2.98%	−1.98%
	$\lfloor m/100 \rfloor$	444	444	228	21	−0.09%	−10.18%

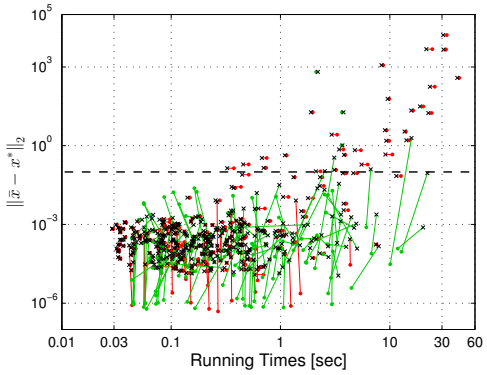


(a) Results on instances with small δ

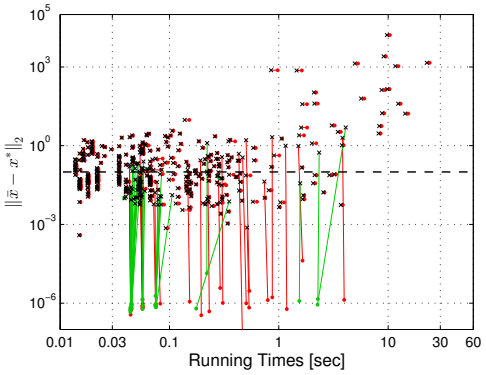
(b) Results on instances with large δ

Figure 3.13. Results of numerical experiments with HOC for (P_1^δ) in ℓ_1-Magic. The plots show running times versus distance to optimum in loglog scale. Black crosses mark the results of ℓ_1-Magic alone and are connected by lines to the corresponding dots marking results obtained with HOC (Algorithm 3.3) integrated into the solver. The colors indicate whether HOC led to a speed-up (green) or an overhead (red), respectively. Blue diamonds mark results obtained only with HOC, where ℓ_1-Magic itself aborted without returning a solution.

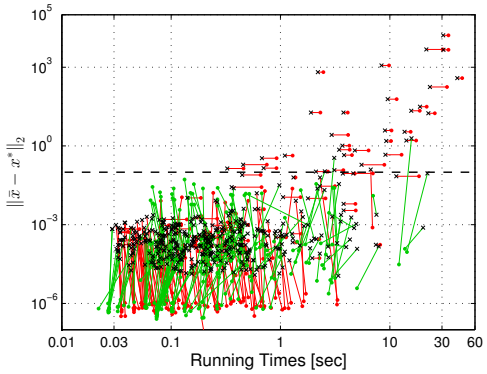
From Figure 3.13, we also see that HOC sometimes yields a less accurate solution on the small-δ instances, but hardly ever for the other test set part. (In fact, note that ℓ_1-Magic itself is already fairly accurate, w.r.t. our acceptance tolerance.) Accuracy improvement occurs more often, and is more pronounced, on the larger-δ test instances. Indeed, the variance of the differences between the distances to x^* obtained without and with HOC, respectively, is of the order 10^{-6} for small-δ problems and about 0.028 for those with large δ values.



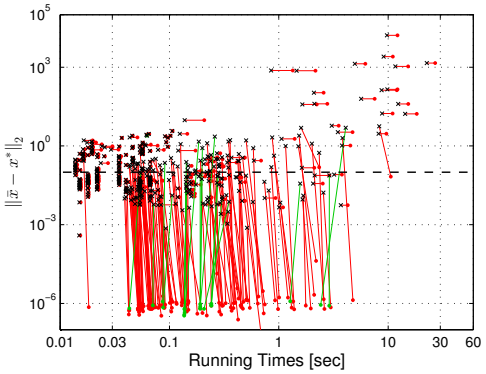
(a) Results on instances with small δ, $R = \lfloor \frac{m}{20} \rfloor$



(b) Results on instances with large δ, $R = \lfloor \frac{m}{20} \rfloor$



(c) Results on instances with small δ, $R = \lfloor \frac{m}{100} \rfloor$



(d) Results on instances with large δ, $R = \lfloor \frac{m}{100} \rfloor$

Figure 3.14. Results of numerical experiments with HOC for (P_1^δ) in SPGL1 (with different HOC frequencies R). The plots show running times versus distance to optimum in loglog scale. Black crosses mark the results of SPGL1 alone and are connected by lines to the corresponding dots marking results obtained with HOC (Algorithm 3.3) integrated into the solver. The colors indicate whether HOC led to a speed-up (green) or an overhead (red), respectively.

Despite the obvious benefit of integrating HOC for (P_1^δ) into ℓ_1-Magic to improve runtimes whilst maintaining comparable accuracy—and leaving aside the error-termination issue—comparing Figures 3.13 and 3.14 immediately shows that ℓ_1-Magic is not competitive with SPGL1; note the different time scales. Indeed, on the (rather small-scale) test set used here, SPGL1 is very fast and reliably produces good approximate solutions, albeit with a few outliers on which it fails.

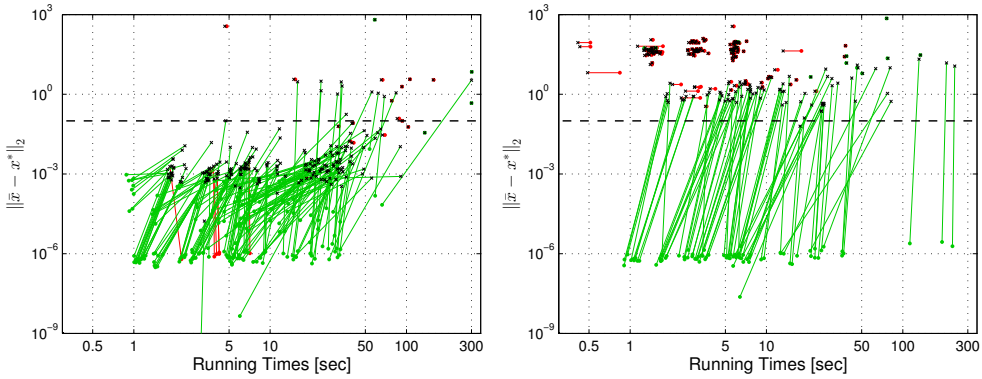
Hence, it comes as no surprise that (on average) HOC almost always introduces an overhead in SPGL1. In fact, an average relative *speed-up* was only achieved on the smaller-δ instances for the three lowest HOC frequencies R. The best choice here turned out to be $R = \lfloor m/20 \rfloor$ (2.98% speed-up). With $R = \lfloor m/100 \rfloor$, the variant with HOC was most often faster than the original code—on 228 of the 444 small-δ instances ($R = \lfloor m/20 \rfloor$: 155)—but gives an overall average overhead of 0.09%. Therefore, we will discuss these two HOC frequencies.

Looking at Figure 3.14(a) and (c), we observe that the distances to the known optima x^* are improved by HOC in many cases, but also sometimes degraded slightly, though all points then remained within our accuracy acceptance tolerance. The variance of the per-instance differences between the distances to x^* for the points computed by SPGL1 without or with HOC, respectively, is around 0.009 for all tested HOC frequencies. The figure also clearly shows that with $R = \lfloor m/20 \rfloor$, the overhead for HOC is naturally much smaller than with $R = \lfloor m/100 \rfloor$, but that more frequent HOC calls specified by the latter choice lead to HOC success in many more cases. Unfortunately, the overall overheads then become dominant.

On the instances with larger δ values, SPGL1 is even faster (but also considerably less accurate). Consequently, here, all HOC frequencies introduce an overhead on average and the number of times an actual improvement was achieved is even less, cf. Table 3.8. Given that the solver is so fast, it is hard to tell whether the time differences are statistically significant at all—both speed-ups and overheads could just be due to normal fluctuations. However, particularly in the regime where SPGL1 is extremely fast (say, below one second), the (mostly red) lines show that the accuracy is significantly improved by HOC (at least with $R = \lfloor m/100 \rfloor$) without sacrificing notable amounts of running time. In fact, they never changed for the worse. Therefore, HOC may arguably still be worth the slight increase in runtime. Moreover, note that the higher frequency of HOC calls ($R = \lfloor m/100 \rfloor$) again dramatically increases the number of instances for which HOC claims success—compare the respective numbers of relatively long lines leading into the high-accuracy region in Figures 3.14(d) and 3.14(b).

Regarding the Denoising-ISAL1 prototype, we only consider instances with 512 rows since its performance seems to degrade strongly the larger the problem dimensions become. Nevertheless, this actually leaves half of the instances—i.e., we still have 222 instances with small and large δ values, respectively—so the experiments should give a viable indication of how HOC influences this solver.

For both test set parts, Figure 3.15 immediately shows that HOC can improve speed and accuracy of ISAL1 significantly: We see lots of (not very steeply sloped) green lines that lead down to very agreeable solutions. The corresponding numbers in Table 3.8 support this impression with quite impressive average speed-ups of



(a) Results on instances with small δ

(b) Results on instances with large δ

Figure 3.15. Results of numerical experiments with HOC for (P_1^δ) in ISAL1. The plots show running times versus distance to optimum in loglog scale. Black crosses mark the results of ISAL1 alone and are connected by lines to the corresponding dots marking results obtained with HOC (Algorithm 3.3) integrated into the solver. The colors indicate whether HOC led to a speed-up (green) or an overhead (red), respectively.

about 42% and 14% on the small- and large-δ test set parts, respectively.

Moreover, it is noteworthy that ISAL1 in its present untuned form (but with HOC), already beats ℓ_1-Magic, as becomes apparent from comparing the scales of Figures 3.15 and 3.13 together with the shapes of the respective point clouds. More precisely, on the respective subsets of the two groups of 222 instances on which the latter solver did not fail to even produce a solution, the overall geometric means of the runtimes are about 12.23 and 5.06 seconds for ISAL1 (with HOC: 6.18 and 3.92) and 11.56 and 17.02 for ℓ_1-Magic (with HOC: 8.31 and 13.72), respectively.

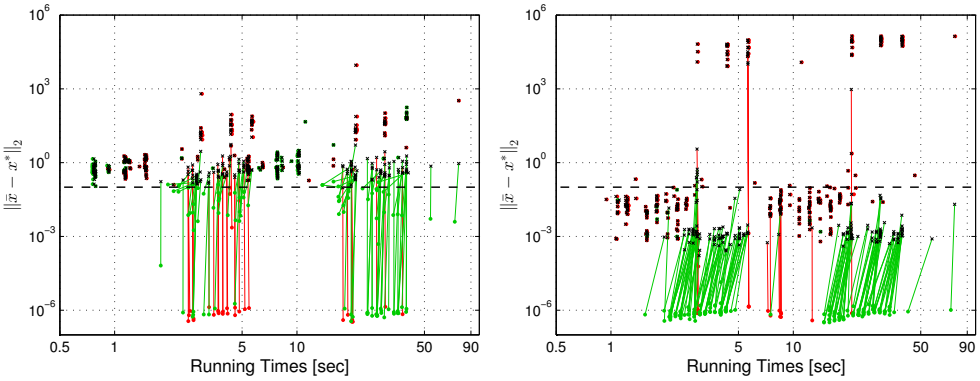
Thus, to summarize, HOC for (P_1^δ) (Algorithm 3.3) has a notable influence on all three tested solvers. Whereas it obviously helps ISAL1 and ℓ_1-Magic, the use within SPGL1 is not quite clear. Here, HOC only sometimes leads to a speed-up but can significantly improve accuracy. Thus, the results here indicate that there might be something to gain by integrating HOC in SPGL1, although many more experiments (preferably large-scale) are necessary before a final answer can be given.

3.6.3.3 ℓ_1-Regularized Least-Squares

For (QP_λ), we integrated HOC (Algorithm 3.4) into ℓ_1-Homotopy and SolveBP/PDCO. For the homotopy method, we again tested the previously specified different HOC frequencies and found that none of them leads to an overall average

Table 3.9. Impact of HOC for (QP$_\lambda$) on the runtimes of ℓ_1-Homotopy and SolveBP/PDCO.

		# instances		# faster		avg. speed-up	
Solver	R	small δ	large δ	small δ	large δ	small δ	large δ
ℓ_1-Homotopy	$\lfloor m/10 \rfloor$	444	444	104	185	−1.93%	−0.06%
SolveBP/PDCO	1	444	444	226	162	1.52%	5.67%



(a) Results on instances with $\lambda = 0.01$

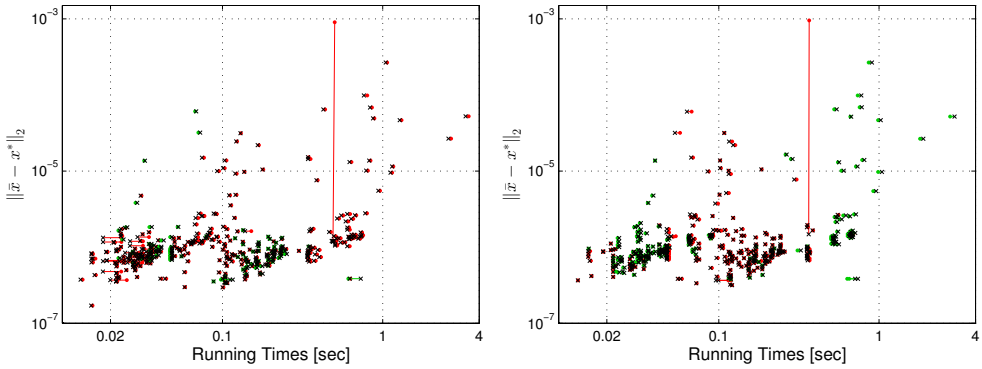
(b) Results on instances with $\lambda = 10$

Figure 3.16. Results of numerical experiments with HOC for (QP$_\lambda$) in SolveBP/PDCO. The plots show running times versus distance to optimum in loglog scale. Black crosses mark the results of SolveBP alone and are connected by lines to the corresponding dots marking results obtained with HOC (Algorithm 3.4) integrated into the solver. The colors indicate whether HOC led to a speed-up (green) or an overhead (red), respectively.

speed-up on the test set considered here. Therefore, we show the results for the experiments with the least frequent number of HOC calls (i.e., $R = \lfloor m/10 \rfloor$). The results for SolveBP/PDCO are depicted in Figure 3.16, those for ℓ_1-Homotopy in Figure 3.17; see also Table 3.9.

Starting with SolveBP, the impression we gain from the two plots in Figure 3.16 is that HOC for (QP$_\lambda$) apparently often leads to higher-accuracy solutions and at the same time slightly reduces the running times. In particular, it can sometimes identify the optimum from a very far-off point, see the long (near-vertical) red lines in Figure 3.16(b) and note the scale on the vertical axis.

For ℓ_1-Homotopy, most claims or indications of becoming faster or slower by



(a) Results on instances with $\lambda = 0.01$

(b) Results on instances with $\lambda = 10$

Figure 3.17. Results of numerical experiments with HOC for (QP_λ) in ℓ_1-Homotopy. The plots show running times versus distance to optimum in loglog scale. Black crosses mark the results of ℓ_1-Homotopy alone and are connected by lines to the corresponding dots marking results obtained with HOC (Algorithm 3.4) integrated into the solver. The colors indicate whether HOC led to a speed-up (green) or an overhead (red), respectively.

using HOC (with $R = \lfloor m/10 \rfloor$) will essentially be meaningless, given how fast the solver is (always below 4 seconds, on all instances). Also, there is no improvement regarding solution accuracy; ℓ_1-Homotopy already is very accurate. The only aspect of Figure 3.17 that stands out is the one long red line, which corresponds to some instance for which HOC—introducing a minor overhead—actually produces a worse approximation of the optimum.

In fact, for the experiments with ℓ_1-Homotopy, a closer inspection reveals that HOC *literally* has no chance to be successful for the majority of instances: The number of homotopy iterations is equal to the optimal solution sparsity on 72% of the test set with $\lambda = 0.01$, and on almost 78% of instances with $\lambda = 10$. In these cases, ℓ_1-Homotopy reconstructs the optimal support adding one index per iteration—then, of course, HOC cannot (correctly) claim success, because all it encounters are subsets of the true optimal support. (Note also that the aforementioned instance corresponding to the long red line in Figure 3.17 demonstrates a case where HOC gives a false-positive answer, albeit still producing a solution with error below 10^{-3}.)

Thus, there seems little hope for HOC to become useful to ℓ_1-Homotopy—at least not on instances with very sparse solutions, and for fairly small-scale problems such as those we considered in the above experiments. On the other hand, SolveBP does profit from HOC.

Combined with the results for (P_1^δ) discussed earlier, more experiments are clearly

needed to investigate whether the appropriate denoising variant of HOC may become (more) useful for ℓ_1-Homotopy or SPGL1, respectively. Nevertheless, the HOC idea may be a useful tool for some algorithms, as the results for ℓ_1-Magic, SolveBP/PDCO and ISAL1 indicate.

CHAPTER 4

ISA Framework for Nonsmooth Convex Optimization

In this chapter, we leave the Compressed Sensing regime for a while and instead focus on general nonsmooth convex constrained optimization problems. We introduce a new algorithmic framework that extends the classical projected subgradient method. Since—unlike in the standard approach—the iterates of our algorithm may become infeasible (due to admitting approximate instead of the usual exact projections), we call our framework *Infeasible-Point Subgradient Algorithm (ISA)*. After investigating convergence properties and extensions of several variants of our algorithm, we eventually return to the CS theme by discussing the application of ISA to the Basis Pursuit problems (P_1) and (P_1^δ) (and some variants closely related to the latter).

This chapter consists largely of a revised (extended and rearranged) version of the paper [175] (joint work by the author with Dirk Lorenz and Marc Pfetsch). We complement these parts (mainly Sections 4.1–4.4) by further results and details, in particular regarding variable target value versions of ISA (Section 4.4.3) and the specialization ISAL1 to ℓ_1-minimization problems (Section 4.6; parts were already presented in [174]). Moreover, we give additional examples of the adaptive approximate projection operator used in the ISA framework (Sections 4.5.2 and 4.5.3) and provide a more thorough treatment of those previously discussed in [175] (see Sections 4.5.1 and 4.5.4).

4.1 Motivation, Scope and Preliminaries

We consider the following general setup: Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ be a convex function (not necessarily differentiable), $\operatorname{dom} f := \{ x : f(x) < \infty \}$, and let $X \subset \operatorname{int}(\operatorname{dom} f) \subseteq \mathbb{R}^n$ be a closed convex set. (Recall that this implies that f is continuous on X.) We are interested in solving the minimization problem

$$\min f(x) \quad \text{s.t.} \quad x \in X. \tag{4.1}$$

Throughout, we will assume (4.1) to have a nonempty set of optima

$$X^* := \operatorname{Argmin}\{ f(x) : x \in X \}.$$

An optimal point will be denoted by x^* and its objective function value $f(x^*)$ by f^*. Similarly, for a sequence (x^k) of points, the corresponding sequence of objective function values will be abbreviated by $(f_k) = (f(x^k))$.

The projected subgradient method [228] is a classical algorithm for solving problem (4.1). One iteration consists of taking a step of size α_k along the negative direction of an arbitrary subgradient h^k of the objective function f at the current point x^k and then computing the next iterate by projection ($\mathcal{P}_X$) onto the feasible set X:

$$x^{k+1} = \mathcal{P}_X(x^k - \alpha_k h^k). \tag{4.2}$$

Over the past decades, numerous extensions and specializations of this scheme have been developed and proven to converge to a minimum (or minimizer, respectively). Well-known disadvantages of the subgradient method are its slow local convergence and the necessity to extensively tune algorithmic parameters in order to obtain practical convergence. On the positive side, subgradient methods involve fast iterations (provided the projections can be performed efficiently) and are easy to implement. In fact, they have been widely used in applications and form one of the most popular algorithms for nonsmooth convex minimization.

The main effort in each iteration of the projected subgradient algorithm usually lies in the computation of the projection $\mathcal{P}_X$. Since the projection is the solution of a (smooth) convex program itself, the required time depends on the structure of X and corresponding specialized algorithms. Examples admitting a fast projection include the case in which X is the nonnegative orthant (simply set all negative entries to zero) or the ℓ_1-norm ball $\{ x : \|x\|_1 \leq r \}$, onto which any $x \in \mathbb{R}^n$ can be projected in $\mathcal{O}(n)$ time [246] (see also [96]). The projection is more involved if X is, for instance, an affine space or a (convex) polyhedron. In such cases, it makes sense to consider replacing the exact projection $\mathcal{P}_X$ by an approximation $\mathcal{P}_X^\varepsilon$.

Here, we will, for a given x, approximate the projected *point* $\mathcal{P}_X(x)$ adaptively up to a desired accuracy ε (i.e., we do not uniformly approximate the projection *operator*). This will be formalized by computing points $\mathcal{P}_X^\varepsilon(x)$ with the property that $\|\mathcal{P}_X^\varepsilon(x) - \mathcal{P}_X(x)\|_2 \leq \varepsilon$ for a given $\varepsilon \geq 0$, see Section 4.1.2 below for a discussion. Algorithmically, the idea is that during the early phases of the subgradient algorithm we do not need a highly accurate projection, and $\mathcal{P}_X^\varepsilon(x)$ can be much faster to compute if ε is larger. In the later phases, one then adaptively tightens the requirement on the accuracy.

A large part of this chapter focuses on the investigation of convergence properties of general variants of the projected subgradient method which rely on such adaptive approximate projections. We study conditions on the step sizes and on the accuracy requirements ε_k (in each iteration k) in order to achieve convergence of the sequence of iterates to an optimal point, or at least convergence of the function values to the optimum. (Often, in the latter case, the corresponding sequence of points can also be guaranteed to converge to an optimal solution x^*, although this is not necessarily the case; see [8] for a discussion.) To that end, we investigate two main variants of the algorithm. In the first one, the sequence (α_k) of step sizes forms a divergent but square-summable series ($\sum \alpha_k = \infty$, $\sum \alpha_k^2 < \infty$) and is given a priori. The second variant uses dynamic step sizes which depend on the difference of the current function value to a *constant target value* that estimates the optimal value. We will also discuss extending the latter variant to a *variable target value* method.

A crucial difference of the resulting algorithms to the standard method is the fact that iterates can be infeasible, i.e., are not necessarily contained in X. As a consequence, the objective function values of the iterates might be smaller than the optimum, which requires a nonstandard convergence analysis; see the proofs in Sections 4.2 and 4.3. Moreover, note that our assumption that X is strictly contained in the interior of the domain of f excludes the case $X = \operatorname{dom} f$, to which our algorithm cannot be applied. Furthermore, we shall (implicitly) assume that every iterate lies in $\operatorname{dom} f$, since otherwise no first-order information is available. This is automatically fulfilled if $\operatorname{dom} f$ is the whole space ($\mathbb{R}^n$), or it can be ensured by requiring that the accuracies ε_k are small enough; cf. also Part 4 of Remark 4.6.

The rest of this chapter is organized as follows: In the remainder of this section, we discuss related approaches from the literature regarding subgradient methods and approximate projections before we recall a few basics and lay the foundations for our ISA framework. In the subsequent sections, we state the aforementioned main versions of ISA and provide the respective convergence results, and then discuss further variants and extensions as well as several examples of suitable adaptive approximate projection operators. Finally, bridging the general theory of this chapter to Compressed Sensing, we detail the application of ISA to the Basis Pursuit

problem (and denoising versions), i.e., the ISAL1 algorithm we encountered earlier in the comparison of ℓ_1-solvers (see Chapter 3).

4.1.1 Related Work

The objective function values of the iterates in subgradient algorithms typically do not decrease monotonically. With the right choice of step sizes, the (projected) subgradient method nevertheless guarantees convergence of the objective function values to the minimum, see, e.g., [228, 211, 17, 213].

A typical result of this sort holds for step size sequences (α_k) which are non-summable ($\sum_{k=0}^{\infty} \alpha_k = \infty$), but square-summable ($\sum_{k=0}^{\infty} \alpha_k^2 < \infty$). Note that the second property implies $\alpha_k \to 0$ as $k \to \infty$.

Another widely used step size rule uses an estimate φ of the optimal value f^*, a subgradient h^k of the objective function f at the current iterate x^k, and relaxation parameters $\lambda_k > 0$:

$$\alpha_k = \lambda_k \frac{f(x^k) - \varphi}{\|h^k\|_2^2}. \tag{4.3}$$

The parameters λ_k are either constant or required to obey certain conditions needed for convergence proofs. The dynamic rule (4.3) is a straightforward generalization of the so-called Polyak-type step size rule, which uses $\varphi = f^*$, to the more practical case when f^* is unknown. The convergence results given in [4] extend the work of Polyak [211, 212] to $\varphi \geq f^*$ and $\varphi < f^*$ by imposing certain conditions on the sequence (λ_k). With the ISA framework, we will generalize these results further, using an adaptive approximate projection operator instead of the (exact) Euclidean projection in (4.2).

Many extensions of the basic subgradient scheme exist, employing variable target value rules (see, e.g., [70, 150, 170, 193, 226, 120, 17]), approximate subgradients [25, 2, 166, 75, 229], or incremental projection schemes [201, 193, 154], to name just a few.

In particular, inexact projections have been used previously, probably most prominently for convex feasibility problems in the framework of successive projection methods. Indeed, the optimization problem (4.1) can, at least theoretically, be cast as the convex feasibility problem to determine $x^* \in X \cap \{\, x \,:\, f(x) \leq f^* \,\}$. Using so-called subgradient projections [16] onto the second set leads to a subgradient step

$$x^{k+1} := x^k - \frac{f(x^k) - f^*}{\|h^k\|^2} h^k,$$

which corresponds to using a Polyak-type step size without relaxation parameter,

employing the exact optimal value. As illustrated in [16], this approach leads to a very flexible framework for convex feasibility problems as well as (nonsmooth) convex optimization problems; see also [201].

Moreover, [263] considers additive vanishing nonsummable error terms (for both the projection and the subgradient step) and establishes the existence of a (decaying) bound on the error terms such that the algorithm will reach a small neighborhood of the optimal set. However, these bounds are not given explicitly. In contrast, the results we shall provide regarding ISA contain explicit conditions for the error terms that guarantee convergence to the optimum.

Another example for the use of inexact projections is the level set subgradient algorithm in [153], although there all iterates are strictly feasible; a related article is [11], where the classical projection is replaced by a variant based on a non-Euclidean distance-like function.

4.1.2 Types of Adaptive Approximate Projections

In our framework, we require that we can adapt the accuracy of the approximation of the projected point *in an absolute sense*, i.e., that for any given accuracy parameter $\varepsilon \geq 0$, our adaptive approximate projection $\mathcal{P}_X^\varepsilon : \mathbb{R}^n \to \mathbb{R}^n$ fulfills

$$\|\mathcal{P}_X^\varepsilon(x) - \mathcal{P}_X(x)\|_2 \leq \varepsilon \qquad \text{for all } x \in \mathbb{R}^n. \tag{4.4}$$

In particular, for $\varepsilon = 0$, we have $\mathcal{P}_X^0 = \mathcal{P}_X$. Note that $\mathcal{P}_X^\varepsilon(x)$ does not necessarily produce a point that is *closer* to $\mathcal{P}_X(x)$ (or even to X) than x itself. In fact, this is only guaranteed for $\varepsilon < d_X(x)$.

It is worth emphasizing that there are at least three conceptually different approaches to approximate projections in the present context. The first concept—prominent, e.g., in the field of convex feasibility problems—uses the idea of approximating the *direction towards the feasible set*, i.e., the iterates approximately move towards the constraint set. In the second, related, approach, one *projects exactly onto supersets* of the constraint set which are easier to handle, e.g., half-spaces. With both ideas one can use powerful notions like Fejér-monotonicity (cf. [190]) or the concept of firmly nonexpansive mappings, see, e.g., [16] and the more recent [168]; see also the "feasibility operator" framework proposed in [201]. To employ either approach, one exploits analytical knowledge about the feasible set, e.g., that it can be written as a (sub-)level set of a known and easy-to-handle convex function.

In the third approach, one aims at *approximating the projected point* without further restricting the direction. This concept applies, for instance, in situations

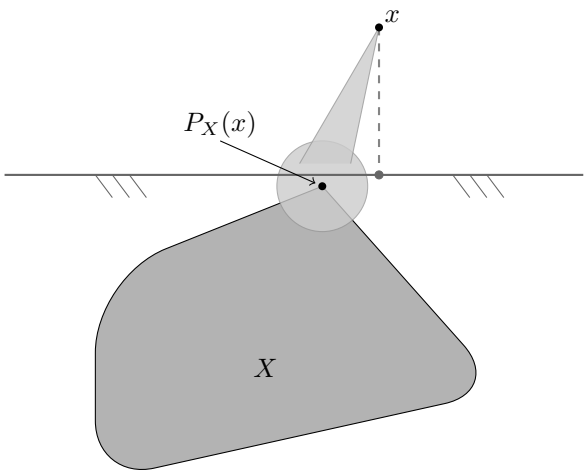


Figure 4.1. Schematic illustration of the three concepts of approximate projection: The approximation of the projection direction (or "moving towards the feasible set") moves from x along a direction within the shaded cone. The exact projection onto a half-space (or other superset) containing X moves along the dashed line. The approximation of the projected point moves from x into a neighborhood of $P_X(x)$, the shaded circle.

in which a computational error is made in the projection step (e.g., as in [263]) or when it is impossible or undesirable to handle the constraints analytically, but a numerical algorithm is available which calculates the projection point up to a given accuracy. Our adaptive approximate projection (4.4) falls under this third category. The latter concept is also similar in spirit to the adaptive evaluation of operators as used, e.g., in adaptive wavelet methods (cf. the **APPLY**-routine in [63]). Recently, in [223, 250], several related approaches of the third type have been studied, based on inexact evaluation of proximity operators (which generalize the Euclidean projection). In particular, the definition of the "type 1 approximation" considered in [223] implies (4.4); however, applied to a projection problem, it also yields feasibility, which (4.4) alone does not guarantee.

Note that, besides the different philosophies and fields of application, none of the approaches directly dominates the other: On the one hand, one may move directly towards the feasible set while missing the projection point, and on the other hand, one may also move closer to the projected point along a direction which is not towards the feasible set; see Figure 4.1 for an illustration. However, one can sometimes, for a given rule which approximates the projection direction, find appropriate half-spaces which contain the feasible set and realize the projection onto

these half-spaces exactly.

In Section 4.5 we will give several examples of adaptive approximate projections in the sense of (4.4). With regard to the above discussion, note that Section 4.5.4 contains a concrete example in which the Fejér-type feasibility operator of [201] is not applicable but the exact projection point can be approximated reasonably well in the sense of our adaptive approximate projection (4.4).

In the ISA framework, we only consider the third approach to approximate projections (more precisely, (4.4)) and thus do not use any assumption like nonexpansiveness or Fejér-monotonicity for the iteration mapping in our convergence analyses in the forthcoming sections.

To conclude this section, let us gather some basic inequalities regarding our approximate projection which will be essential in establishing the results to follow.

Lemma 4.1. *For an adaptive approximate projection $\mathcal{P}_X^\varepsilon$ obeying (4.4), where $X \subset \mathbb{R}^n$ is a closed convex set, it holds for any $y \in \mathbb{R}^n$ and an arbitrary x with $d_X(x) \leq \delta$ that*

$$\|\mathcal{P}_X^\varepsilon(y) - x\|_2 \;\leq\; \|y - x\|_2 + \varepsilon + \delta. \tag{4.5}$$

In particular, for any $y \in \mathbb{R}^n$ and $x \in X$,

$$\|\mathcal{P}_X^\varepsilon(y) - x\|_2 \;\leq\; \|y - x\|_2 + \varepsilon. \tag{4.6}$$

Moreover, for any $x \in X$, $\varepsilon \geq 0$, and for $y = z - \alpha\, h$ with some $z \in \mathbb{R}^n$, $\alpha \geq 0$ and $h \in \partial f(z)$ for a convex function f, we have

$$\|\mathcal{P}_X^\varepsilon(y) - x\|_2^2 \;\leq\; \|z - x\|_2^2 - 2\,\alpha(f(z) - f(x)) + (\alpha\,\|h\|_2 + \varepsilon)^2 + 2\,\|z - x\|_2\,\varepsilon. \tag{4.7}$$

Proof. Recall that the exact Euclidean projection is nonexpansive; therefore,

$$\|\mathcal{P}_X(y) - x\|_2 \;\leq\; \|y - x\|_2 \qquad \forall x \in X. \tag{4.8}$$

Hence, by (4.4) and (4.8), for the adaptive approximate projection $\mathcal{P}_X^\varepsilon$ we obtain (4.6) for all $x \in X$ and any $y \in \mathbb{R}^n$:

$$\begin{aligned} \|\mathcal{P}_X^\varepsilon(y) - x\|_2 \;&=\; \|\mathcal{P}_X^\varepsilon(y) - \mathcal{P}_X(y) + \mathcal{P}_X(y) - x\|_2 \\ &\leq\; \|\mathcal{P}_X^\varepsilon(y) - \mathcal{P}_X(y)\|_2 + \|\mathcal{P}_X(y) - x\|_2 \;\leq\; \varepsilon + \|y - x\|_2. \end{aligned}$$

Moreover, it follows from (4.8) and (4.4) that, for any x with $d_X(x) \leq \delta$,

$$\begin{aligned} \|\mathcal{P}_X^\varepsilon(y) - x\|_2 \;&\leq\; \|\mathcal{P}_X^\varepsilon(y) - \mathcal{P}_X(x)\|_2 + \|\mathcal{P}_X(x) - x\|_2 \;\leq\; \|\mathcal{P}_X^\varepsilon(y) - \mathcal{P}_X(x)\|_2 + \delta \\ &\leq\; \|\mathcal{P}_X(y) - \mathcal{P}_X(x)\|_2 + \|\mathcal{P}_X^\varepsilon(y) - \mathcal{P}_X(y)\|_2 + \delta \;\leq\; \|y - x\|_2 + \varepsilon + \delta. \end{aligned}$$

Algorithm 4.1 Predetermined Step Size ISA

Input: a starting point x^0, sequences (α_k), (ε_k)
Output: an (approximate) solution to (4.1)

1: initialize $k := 0$
2: **repeat**
3: choose a subgradient $h^k \in \partial f(x^k)$ of f at x^k
4: compute the next iterate $x^{k+1} := \mathcal{P}_X^{\varepsilon_k}\left(x^k - \alpha_k h^k\right)$
5: increment $k := k + 1$
6: **until** a stopping criterion is satisfied

Thus, (4.5) holds. Finally, let $z \in \mathbb{R}^n$, $\alpha \geq 0$, $h \in \partial f(z)$, $\varepsilon \geq 0$ and let $y = z - \alpha\, h$. We obtain (4.7) for any $x \in X$:

$$\begin{aligned}
&\|\mathcal{P}_X^{\varepsilon}(y) - x\|_2^2 \\
&\leq \left(\|y - x\|_2 + \varepsilon_k\right)^2 \;=\; \|y - x\|_2^2 + 2\,\|y - x\|_2\,\varepsilon + \varepsilon^2 \\
&= \|z - x\|_2^2 - 2\,\alpha\, h^\top (z - x) + \alpha^2\,\|h\|_2^2 + 2\,\|y - x\|_2\,\varepsilon + \varepsilon^2 \\
&\leq \|z - x\|_2^2 - 2\,\alpha(f(z) - f(x)) + \alpha^2\,\|h\|_2^2 + 2\|z - x\|_2\,\varepsilon + 2\,\alpha\,\|h\|_2\,\varepsilon + \varepsilon^2 \\
&= \|z - x\|_2^2 - 2\,\alpha(f(z) - f(x)) + \left(\alpha\,\|h\|_2 + \varepsilon\right)^2 + 2\,\|z - x\|_2\,\varepsilon,
\end{aligned}$$

where the second inequality follows from the subgradient definition (1.1) and the triangle inequality. □

Note that the above inequalities (4.6), (4.7) and (4.8) hold, in particular, for every optimal point $x^* \in X^*$.

4.2 ISA with Predetermined Step Sizes

In the following, we shall discuss the first variant of ISA, in which we assume that the sequences of step sizes (α_k) and projection accuracies (ε_k) are predetermined (i.e., given a priori); we obtain Algorithm 4.1.

Throughout this section, let (x^k) denote the sequence of points with corresponding objective function values (f_k) and subgradients (h^k), $h^k \in \partial f(x^k)$, as generated by Algorithm 4.1.

Note that while $h^k = 0$ might occur in the course of the iterations, it does not necessarily imply that x^k is optimal, because x^k may be infeasible. In such a case, the adaptive projection (obeying (4.4)) will eventually change x^k to a different point as soon as ε_k becomes small enough.

The following is our main convergence result for this variant of ISA, using fairly standard step size conditions. (The stopping criterion alluded to in the algorithm statement will be ignored for the convergence analysis. In practical implementations, one would stop, e.g., if no significant progress in the objective was achieved within a certain number of iterations.)

Theorem 4.2 (Convergence for predetermined step size sequences)**.** *Let the projection accuracy sequence* (ε_k) *be such that*

$$\varepsilon_k \geq 0, \quad \sum_{k=0}^{\infty} \varepsilon_k < \infty, \tag{4.9}$$

let the positive step size sequence (α_k) *be such that*

$$\sum_{k=0}^{\infty} \alpha_k = \infty, \quad \sum_{k=0}^{\infty} \alpha_k^2 < \infty, \tag{4.10}$$

and let the following relation hold:

$$\alpha_k \geq \sum_{j=k}^{\infty} \varepsilon_j \qquad \forall\, k = 0, 1, 2, \ldots \tag{4.11}$$

Suppose $\|h^k\|_2 \leq H < \infty$ *for all* k*. Then the sequence of ISA iterates* (x^k) *converges to an optimal point* $x^* \in X^*$ *of problem* (4.1)*.*

Remark 4.3. Relations (4.9), (4.10), and (4.11) can be ensured, e.g., by the sequences $\varepsilon_k = 1/k^2$ and $\alpha_k = 1/(k-1)$ for $k > 1$; then, in particular,

$$\sum_{j=k}^{\infty} \varepsilon_k \;=\; \sum_{j=k}^{\infty} \frac{1}{j^2} \;\leq\; \int_{k-1}^{\infty} \frac{1}{y^2}\, dy \;=\; \frac{1}{k-1} = \alpha_k.$$

The proof of convergence of the ISA iterates x^k is somewhat more involved than for the classical subgradient method (as given, e.g., in [228]). This is due to the additional error terms from adaptive approximate projections as well as the fact that $f(x^k) \geq f^*$ is not guaranteed since the iterates may be infeasible.

Proof of Theorem 4.2. Defining

$$\beta_k := \left(\alpha_k \|h^k\|_2 + \varepsilon_k\right)^2 + 2\, \|x^k - x^*\|_2\, \varepsilon_k,$$

we can rewrite (4.7) with $x = x^* \in X^*$ as

$$\|x^{k+1} - x^*\|_2^2 \;\le\; \|x^k - x^*\|_2^2 - 2\,\alpha_k\,(f_k - f^*) + \beta_k. \tag{4.12}$$

Thus, we obtain (by applying (4.12) for $k = 0, \dots, m$)

$$\|x^{m+1} - x^*\|_2^2 \;\le\; \|x^0 - x^*\|_2^2 - 2\sum_{k=0}^{m}(f_k - f^*)\alpha_k + \sum_{k=0}^{m}\beta_k.$$

Our first goal is to show that $\sum_{k=0}^{\infty}\beta_k$ is a convergent series. Using $\|h^k\|_2 \le H$ and denoting $A \coloneqq \sum_{k=0}^{\infty}\alpha_k^2$, we get

$$\sum_{k=0}^{m}\beta_k \;\le\; AH^2 + \sum_{k=0}^{m}\varepsilon_k^2 + 2H\sum_{k=0}^{m}\alpha_k\varepsilon_k + 2\sum_{k=0}^{m}\|x^k - x^*\|_2\,\varepsilon_k.$$

Let $D \coloneqq \|x^0 - x^*\|_2$ and consider the last term (without the factor 2):

$$\begin{aligned}
&\sum_{k=0}^{m}\|x^k - x^*\|_2\,\varepsilon_k \\
&= D\,\varepsilon_0 + \sum_{k=1}^{m}\left\|\mathcal{P}_X^{\varepsilon_{k-1}}\left(x^{k-1} - \alpha_{k-1}h^{k-1}\right) - x^*\right\|_2\,\varepsilon_k \\
&\le D\,\varepsilon_0 + \sum_{k=1}^{m}\left\|\mathcal{P}_X^{\varepsilon_{k-1}}\left(x^{k-1} - \alpha_{k-1}h^{k-1}\right) - \mathcal{P}_X\left(x^{k-1} - \alpha_{k-1}h^{k-1}\right)\right\|_2\,\varepsilon_k \\
&\qquad + \sum_{k=1}^{m}\left\|\mathcal{P}_X\left(x^{k-1} - \alpha_{k-1}h^{k-1}\right) - x^*\right\|_2\,\varepsilon_k \\
&\le D\,\varepsilon_0 + \sum_{k=1}^{m}\varepsilon_{k-1}\varepsilon_k + \sum_{k=1}^{m}\left\|x^{k-1} - \alpha_{k-1}h^{k-1} - x^*\right\|_2\,\varepsilon_k \\
&\le D\,\varepsilon_0 + \sum_{k=0}^{m-1}\varepsilon_k\varepsilon_{k+1} + \sum_{k=0}^{m-1}\|x^k - x^*\|_2\,\varepsilon_{k+1} + \sum_{k=0}^{m-1}\|h^k\|_2\,\alpha_k\,\varepsilon_{k+1} \\
&\le D\,(\varepsilon_0 + \varepsilon_1) + \sum_{k=0}^{m-1}\varepsilon_k\varepsilon_{k+1} + \sum_{k=1}^{m-1}\|x^k - x^*\|_2\,\varepsilon_{k+1} + H\sum_{k=0}^{m-1}\alpha_k\,\varepsilon_{k+1}.
\end{aligned} \tag{4.13}$$

(Above, we employed (4.6) from Lemma 4.1 with respect to $x^* \in X^* \subseteq X$.) Repeating this procedure to eliminate all terms $\|x^k - x^*\|_2$ for $k > 0$, we obtain

$$(4.13) \;\le\; \dots \le D\sum_{k=0}^{m}\varepsilon_k + \sum_{j=1}^{m}\Big(\sum_{k=0}^{m-j}\varepsilon_k\varepsilon_{k+j} + H\sum_{k=0}^{m-j}\alpha_k\varepsilon_{k+j}\Big)$$

$$= D \sum_{k=0}^{m} \varepsilon_k + \sum_{j=1}^{m} \sum_{k=0}^{m-j} (\varepsilon_k + H\alpha_k)\, \varepsilon_{k+j}.$$

Denote $E \coloneqq \sum_{k=0}^{\infty} \varepsilon_k$. From the above chain of inequalities, (4.9) and (4.11), we finally get:

$$\begin{aligned}
&\|x^{m+1} - x^*\|_2^2 + 2 \sum_{k=0}^{m} (f_k - f^*)\, \alpha_k \;\leq\; D^2 + \sum_{k=0}^{m} \beta_k \\
&\leq D^2 + AH^2 + \sum_{k=0}^{m} \varepsilon_k^2 + 2\, H \sum_{k=0}^{m} \alpha_k \varepsilon_k + 2\, D \sum_{k=0}^{m} \varepsilon_k + 2 \sum_{j=1}^{m} \sum_{k=0}^{m-j} (\varepsilon_k + H\alpha_k)\, \varepsilon_{k+j} \\
&\leq D^2 + AH^2 + 2\, D \sum_{k=0}^{m} \varepsilon_k + 2 \sum_{j=0}^{m} \sum_{k=0}^{m-j} \varepsilon_k \varepsilon_{k+j} + 2\, H \sum_{j=0}^{m} \sum_{k=0}^{m-j} \alpha_k \varepsilon_{k+j} \\
&= D^2 + AH^2 + 2\, D \sum_{k=0}^{m} \varepsilon_k + 2 \sum_{j=0}^{m} \left(\varepsilon_j \sum_{k=j}^{m} \varepsilon_k \right) + 2\, H \sum_{j=0}^{m} \left(\alpha_j \sum_{k=j}^{m} \varepsilon_k \right) \\
&\leq D^2 + AH^2 + 2\, D \sum_{k=0}^{m} \varepsilon_k + 2 \sum_{j=0}^{m} E\, \varepsilon_j + 2H \sum_{j=0}^{m} \alpha_j\, \alpha_j \\
&\leq D^2 + AH^2 + 2\, (D + E) \sum_{k=0}^{m} \varepsilon_k + 2\, H \sum_{k=0}^{m} \alpha_k^2 \\
&\leq (D + E)^2 + E^2 + (2 + H)\, A\, H \;=:\; R \;<\; \infty. \qquad (4.14)
\end{aligned}$$

Thus, $\sum_{k=0}^{\infty} \beta_k < \infty$ holds indeed. Note, however, that since the iterates x^k may be infeasible, possibly $f_k < f^*$, whence the second term on the left-most side of (4.14) might be negative. Therefore, we next distinguish two cases:

(i) If $f_k \geq f^*$ for all but finitely many k, we can assume without loss of generality that $f_k \geq f^*$ for all k (by considering only the "later" iterates). Now, because $f_k \geq f^*$ for all k,

$$\sum_{k=0}^{m} (f_k - f^*)\, \alpha_k \;\geq\; \sum_{k=0}^{m} \Big(\underbrace{\min_{j=0,\ldots,m} f_j}_{=:f_m^*} - f^* \Big)\, \alpha_k \;=\; (f_m^* - f^*) \sum_{k=0}^{m} \alpha_k.$$

Together with (4.14) this yields

$$0 \;\leq\; 2\, (f_m^* - f^*) \sum_{k=0}^{m} \alpha_k \;\leq\; R \qquad \Leftrightarrow \qquad 0 \;\leq\; f_m^* - f^* \;\leq\; \frac{R}{2 \sum_{k=0}^{m} \alpha_k}.$$

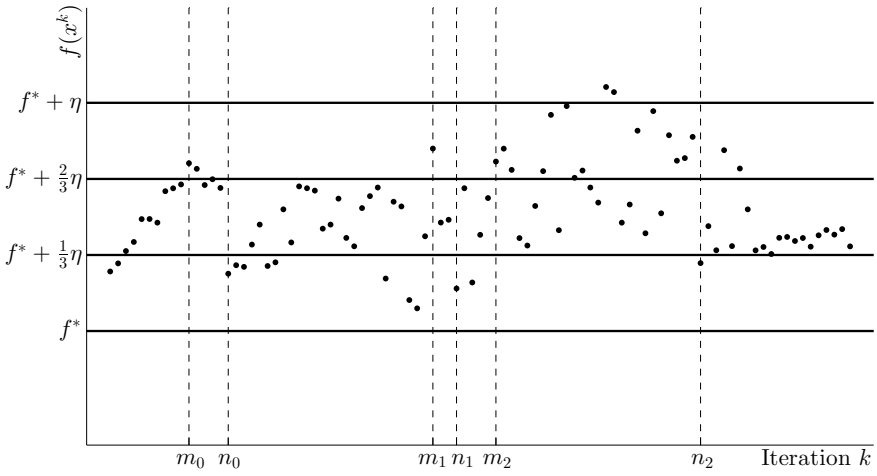


Figure 4.2. The sequences (m_ℓ) and (n_ℓ).

Thus, because $\sum_{k=0}^{m} \alpha_k$ diverges, we have $f_m^* \to f^*$ for $m \to \infty$ (and, in particular, $\liminf_{k\to\infty} f_k = f^*$).

To show that f^* is in fact the only possible accumulation point (and hence the limit) of (f_k), assume that (f_k) has another accumulation point strictly larger than f^*, say $f^* + \eta$ for some $\eta > 0$. Then, both cases $f_k < f^* + \frac{1}{3}\eta$ and $f_k > f^* + \frac{2}{3}\eta$ must occur infinitely often. We can therefore define two index subsequences (m_ℓ) and (n_ℓ) by setting $n_{(-1)} := -1$ and, for $\ell \geq 0$,

$$\begin{aligned} m_\ell &:= \min\{\, k \,:\, k > n_{\ell-1},\ f_k > f^* + \tfrac{2}{3}\eta \,\}, \\ n_\ell &:= \min\{\, k \,:\, k > m_\ell,\ f_k < f^* + \tfrac{1}{3}\eta \,\}. \end{aligned}$$

Figure 4.2 illustrates this choice of indices. Now observe that for any ℓ, writing $y^{m_\ell} := \mathcal{P}_X(x^{m_\ell - 1} - \alpha_{m_\ell - 1} h^{m_\ell - 1}) \in X$ and proceeding similarly to (4.13), we have (using, in particular, the subgradient inequality (1.1), (4.4) and Lemma 4.1):

$$\begin{aligned} \tfrac{1}{3}\eta \;&<\; f_{m_\ell} - f_{n_\ell} \\ &\leq\; (h^{m_\ell})^\top (x^{n_\ell} - x^{m_\ell}) \leq H \cdot \|x^{n_\ell} - x^{m_\ell}\|_2 \\ &\leq\; H\,(\|x^{n_\ell} - y^{m_\ell}\|_2 + \varepsilon_{m_\ell - 1}) \\ &\leq\; H\,\big(\|x^{n_\ell - 1} - y^{m_\ell}\|_2 + H\alpha_{n_\ell - 1} + \varepsilon_{n_\ell - 1} + \varepsilon_{m_\ell - 1}\big) \end{aligned}$$

$$\leq \ \ldots \ \leq \ H\,\|x^{m_\ell} - y^{m_\ell}\|_2 + H^2 \sum_{j=m_\ell}^{n_\ell-1} \alpha_j + H \sum_{j=m_\ell}^{n_\ell-1} \varepsilon_j + H\,\varepsilon_{m_\ell-1}$$

$$\leq H^2 \sum_{j=m_\ell}^{n_\ell-1} \alpha_j + H \sum_{j=m_\ell}^{n_\ell-1} \varepsilon_j + 2\,H\,\varepsilon_{m_\ell-1}. \tag{4.15}$$

For a given m, let $\ell_m := \max\{\,\ell \,:\, n_\ell - 1 \leq m\,\}$ be the number of blocks of indices between two consecutive indices m_ℓ and $n_\ell - 1$ until m. We obtain:

$$\begin{aligned} \tfrac{1}{3}\sum_{\ell=0}^{\ell_m} \eta \ &\leq \ H^2 \sum_{\ell=0}^{\ell_m} \sum_{j=m_\ell}^{n_\ell-1} \alpha_j + H \sum_{\ell=0}^{\ell_m} \sum_{j=m_\ell}^{n_\ell-1} \varepsilon_j + 2\,H \sum_{\ell=0}^{\ell_m} \varepsilon_{m_\ell-1} \\ &\leq \ H^2 \sum_{\ell=0}^{\ell_m} \sum_{j=m_\ell}^{n_\ell-1} \alpha_j + 3HE. \end{aligned} \tag{4.16}$$

For $m \to \infty$, the left hand side tends to infinity, and since $HE < \infty$, this implies

$$\sum_{\ell=0}^{\ell_m} \sum_{j=m_\ell}^{n_\ell-1} \alpha_j \ \to \ \infty.$$

Then, since $\alpha_k > 0$ and $f_k \geq f^*$ for all k, (4.14) yields

$$\begin{aligned} \infty \ &> \ R \ \geq \ \|x^{m+1} - x^*\|_2^2 + 2\sum_{k=0}^{m}(f_k - f^*)\alpha_k \ \geq \ 2\sum_{k=0}^{m}(f_k - f^*)\alpha_k \\ &\geq \ 2\sum_{\ell=0}^{\ell_m} \sum_{j=m_\ell}^{n_\ell-1} \underbrace{(f_j - f^*)}_{>\frac{1}{3}\eta}\alpha_j \ > \ \tfrac{2}{3}\eta \sum_{\ell=0}^{\ell_m} \sum_{j=m_\ell}^{n_\ell-1} \alpha_j. \end{aligned}$$

But for $m \to \infty$, this yields a contradiction since the sum on the right hand side diverges. Hence, there does not exist an accumulation point strictly larger than f^*, so we can conclude $f_k \to f^*$ as $k \to \infty$, i.e., the whole sequence (f_k) converges to f^*.

We now consider convergence of the sequence (x^k). From (4.14) we conclude that both terms on the left hand side are bounded independently of m. In particular, this means (x^k) is a bounded sequence. Hence, by the Bolzano-Weierstraß Theorem, it has a convergent subsequence (x^{k_i}) with $x^{k_i} \to \overline{x}$ (as $i \to \infty$) for some $\overline{x}$. To show that the full sequence (x^k) converges to $\overline{x}$,

take any K and any $k_i < K$ and observe from (4.12) that

$$\|x^K - \overline{x}\|_2^2 \;\leq\; \|x^{k_i} - \overline{x}\|_2^2 + \sum_{j=k_i}^{K-1} \beta_j.$$

Since $\sum_{k=0}^{\infty} \beta_k$ is a convergent series (see (4.14)), the right hand side becomes arbitrarily small for k_i and K large enough. This implies $x^k \to \overline{x}$, and since $\varepsilon_k \to 0$, $f_k \to f^*$, and X^* is closed, $\overline{x} \in X^*$ must hold.

(ii) Now consider the case in which $f_k < f^*$ occurs infinitely often. We write (f_k^-) for the subsequence of (f_k) with $f_k < f^*$ and (f_k^+) for the subsequence with $f_k \geq f^*$. Clearly $f_k^- \to f^*$. Indeed, the corresponding iterates are asymptotically feasible (since the projection accuracy ε_k tends to zero), and hence f^* is the only possible accumulation point of (f_k^-).

Denoting $M_m^- = \{\, k \leq m \;:\; f_k < f^* \,\}$ and $M_m^+ = \{\, k \leq m \;:\; f_k \geq f^* \,\}$, we conclude from (4.14) that

$$\|x^{m+1} - x^*\|_2^2 + 2 \sum_{k \in M_m^+} (f_k - f^*)\, \alpha_k \;\leq\; R + 2 \sum_{k \in M_m^-} (f^* - f_k)\, \alpha_k. \tag{4.17}$$

Note that each summand is nonnegative. To see that the right hand side is bounded independently of m, let $y^{k-1} = x^{k-1} - \alpha_{k-1}\, h^{k-1}$, and observe that here (i.e., with $k \in M_m^-$), due to $f_k < f^* \leq f(\mathcal{P}_X(y^{k-1}))$, we have

$$\begin{aligned} f^* - f_k \;&\leq\; f\big(\mathcal{P}_X(y^{k-1})\big) - f\big(\mathcal{P}_X^{\varepsilon_{k-1}}(y^{k-1})\big) \\ &\leq\; (h^{k-1})^\top \big(\mathcal{P}_X(y^{k-1}) - \mathcal{P}_X^{\varepsilon_{k-1}}(y^{k-1})\big) \\ &\leq\; \|h^{k-1}\|_2 \, \big\|\mathcal{P}_X(y^{k-1}) - \mathcal{P}_X^{\varepsilon_{k-1}}(y^{k-1})\big\|_2 \;\leq\; H \varepsilon_{k-1}, \end{aligned}$$

using the subgradient and Cauchy-Schwarz inequalities as well as property (4.4) of $\mathcal{P}_X^{\varepsilon}$ and the boundedness of the subgradient norms. From (4.17), using (4.10) and (4.11), we thus obtain

$$\begin{aligned} &\|x^{m+1} - x^*\|_2^2 + 2 \sum_{k \in M_m^+} (f_k - f^*) \alpha_k \\ &\leq R + 2H \sum_{k \in M_m^-} \alpha_k\, \varepsilon_{k-1} \;\leq\; R + 2H \sum_{k \in M_m^-} \alpha_k\, \alpha_{k-1} \\ &\leq R + 2H \sum_{k=0}^{\infty} \alpha_k\, \alpha_{k-1} \;\leq\; R + 4AH \;<\; \infty. \end{aligned} \tag{4.18}$$

Similarly to case (i), we conclude that both the sequence (x^k) and the series

$\sum_{k\in M_m^+}(f_k - f^*)\,\alpha_k$ are bounded.
It remains to show that $f_k^+ \to f^*$. Assume to the contrary that (f_k^+) has an accumulation point $f^* + \eta$ for $\eta > 0$. Similarly to before, we construct index subsequences (m_ℓ) and (p_ℓ) as follows: Set $p_{(-1)} := -1$ and define, for $\ell \geq 0$,

$$\begin{aligned} m_\ell &:= \min\{\, k \in M_\infty^+ \,:\, k > p_{\ell-1},\ f_k > f^* + \tfrac{2}{3}\eta \,\}, \\ p_\ell &:= \min\{\, k \in M_\infty^- \,:\, k > m_\ell \,\}. \end{aligned}$$

Then $m_\ell, \ldots, p_\ell - 1 \in M_\infty^+$ for all ℓ, and we have (cf. (4.15))

$$\tfrac{2}{3}\eta \;<\; f_{m_\ell} - f_{p_\ell} \;\leq\; H^2 \sum_{j=m_\ell}^{p_\ell-1} \alpha_j + H \sum_{j=m_\ell}^{p_\ell-1} \varepsilon_j + 2\,H\,\varepsilon_{m_\ell-1}.$$

Therefore (cf. (4.16)), with $\ell_m := \max\{\, \ell \,:\, p_\ell - 1 \leq m \,\}$ for a given m,

$$\tfrac{2}{3}\sum_{\ell=0}^{\ell_m} \eta \;\leq\; H^2 \sum_{\ell=0}^{\ell_m} \sum_{j=m_\ell}^{p_\ell-1} \alpha_j + 3\,HE.$$

Now the left hand side becomes arbitrarily large as $m \to \infty$, so that also $\sum_{\ell=0}^{\ell_m} \sum_{j=m_\ell}^{p_\ell-1} \alpha_j \to \infty$, since $HE < \infty$. Note that because $\alpha_k > 0$ and

$$\sum_{\ell=0}^{\ell_m} \sum_{j=m_\ell}^{p_\ell-1} \alpha_j \;\leq\; \sum_{k\in M_m^+} \alpha_k,$$

this latter series must diverge as well. As a consequence, f^* is itself an (other) accumulation point of (f_k^+): From (4.18) we have

$$\begin{aligned} \infty \;&>\; R + 4AH \;\geq\; 2 \sum_{k\in M_m^+} (f_k - f^*)\alpha_k \\ &\geq \sum_{k\in M_m^+} (\underbrace{\min\{\, f_j \,:\, j \in M_m^+,\ j \leq m \,\}}_{=:\hat f_m^*} - f^*)\,\alpha_k \;=\; (\hat f_m^* - f^*) \sum_{k\in M_m^+} \alpha_k, \end{aligned}$$

and thus

$$0 \;\leq\; \hat f_m^* - f^* \;\leq\; \frac{R + 4\,A\,H}{\sum_{k\in M_m^+} \alpha_k} \;\to\; 0 \qquad \text{as } m \to \infty,$$

since $\sum_{k\in M_m^+} \alpha_k$ diverges. But then, knowing $(\hat f_k^*)$ converges to f^*, we can

use (m_ℓ) and another index subsequence (n_ℓ), given by

$$n_\ell := \min\{\, k \in M_\infty^+ \,:\, k > m_\ell,\ f_k < f^* + \tfrac{1}{3}\eta \,\},$$

to proceed analogously to case (i) to arrive at a contradiction. Thus, we conclude that no $\eta > 0$ exists such that $f^* + \eta$ is an accumulation point of (f_k^+).

On the other hand, since (x^k) is bounded and f is continuous on a neighborhood of X (recall that for all k, x^k is contained in an ε_k-neighborhood of X), (f_k^+) is bounded. Thus, it must have at least one accumulation point. Since $f_k \geq f^*$ for all $k \in M_\infty^+$, the only possibility left is f^* itself. Hence, f^* is the unique accumulation point (i.e., the limit) of the sequence (f_k^+). As this is also true for (f_k^-), the whole sequence (f_k) converges to f^*.

Finally, convergence of the bounded sequence (x^k) to some $\overline{x} \in X^*$ can now be obtained just like in case (i), completing the proof. □

4.3 ISA with Dynamic Step Sizes

In order to apply the dynamic step size rule (4.3), i.e.,

$$\alpha_k = \lambda_k \frac{f(x^k) - \varphi}{\|h^k\|_2^2},$$

we need several modifications of the basic method, yielding Algorithm 4.2. This algorithm works with a constant estimate φ of the optimal objective function value f^* and essentially tries to reach a feasible point x^k with $f(x^k) \leq \varphi$. (Note that if $\varphi = f^*$, we would have obtained an optimal point in this case.) Throughout this section, (x^k), (f_k), and (h^k) denote the sequences of iterates, corresponding objective function values, and subgradients as generated by Algorithm 4.2, respectively. An extension to variable estimates of f^* is discussed in Section 4.4.3 below.

Remark 4.4. A few comments on Algorithm 4.2 are in order:

1. Since $0 < \gamma < 1$, $\gamma^\ell \to 0$ (strictly monotonically) for $\ell \to \infty$. Thus, Steps 3–9 constitute a *projection accuracy refinement phase*, i.e., an inner loop in which the current k is temporarily fixed, and x^k is *recomputed* with a stricter accuracy setting for the adaptive projection. This phase either leads to a point showing $\varphi \geq f^*$ (by termination or convergence in the inner loop over ℓ) or eventually resets x^k to a point with $f_k > \varphi$ and $h^k \neq 0$ so that the regular (outer)

Algorithm 4.2 Dynamic Step Size ISA

Input: estimate φ of f^*, starting point x^0, sequences (λ_k) and (ε_k), refinement parameter $\gamma \in (0,1)$

Output: an (approximate) solution to (4.1)

initialize $k := 0$, $\ell = -1$, $x^{-1} := x^0$, $h^{-1} := 0$, $\alpha_{-1} := 0$, $\varepsilon_{-1} := \varepsilon_0$
repeat
    choose a subgradient $h^k \in \partial f(x^k)$
    if $f_k \leq \varphi$ **or** $h^k = 0$ **then**
        if $x^k \in X$ **then**
            stop (with x^k feasible, showing $\varphi \geq f^*$; optimal if $h^k = 0$)
        increment $\ell := \ell + 1$
        reset $x^k := \mathcal{P}_X^{\varepsilon}(x^{k-1} - \alpha_{k-1} h^{k-1})$ for $\varepsilon = \gamma^\ell \varepsilon_{k-1}$
        go to Step 3
    compute step size $\alpha_k := \lambda_k (f_k - \varphi)/\|h^k\|_2^2$
    compute the next iterate $x^{k+1} := \mathcal{P}_X^{\varepsilon_k}(x^k - \alpha_k h^k)$
    reset $\ell := 0$ and increment $k := k + 1$
until a stopping criterion is satisfied

iteration is resumed (with k no longer fixed).

2. Note that, if x^0 is such that $f_0 \leq \varphi$ or $h^k = 0$, the algorithm begins with such a refinement phase, projecting x^0 more and more accurately until neither case holds any longer (if possible); the initializations with counter -1 are needed for this eventuality. (Clearly, ε_{-1} needs not be initialized to ε_0, but any nonnegative constant would suffice.) Moreover, we could of course postpone the (repeated) determination of a subgradient (Step 3) in a refinement phase until $f_k > \varphi$ is achieved, i.e., $h^k = 0$ would necessarily be the only reason for another accuracy refinement. This may be important in practice, where finding a subgradient is sometimes expensive itself, and the case $h^k = 0$ presumably occurs very rarely anyway. For the sake of brevity we did not treat this explicitly in Algorithm 4.2.

3. There are various ways in which the accuracy refinement phase could be realized. Instead of (γ^ℓ) with constant $\gamma \in (0,1)$, any (strictly) monotonically decreasing sequence (γ_ℓ) could be used. Since we will need $\varepsilon_k \to 0$ to achieve feasibility (in the limit) anyway, which implies that for all k there always exists some $L > 0$ such that $\varepsilon_{k+L} < \varepsilon_k$, we could also use $\min\{\varepsilon_{k-1}, \varepsilon_{k-1+\ell}\}$ as the recalibrated accuracy. Moreover, we do not need to fix k, i.e., repeatedly *replace* x^k by finer approximate projections, but could produce a finite series of identical iterates (each reset to the last one before the inner loop started) until the refinement phase is over. Similarly, we could use $\alpha_k = \max\{0, \lambda_k (f_k - \varphi)/\|h^k\|_2^2\}$ (and 0 if $h^k = 0$); letting $\varepsilon_k \to 0$ then

naturally implements the refinement, while in iterations with $\alpha_k = 0$, the produced point may move up to ε_k *away* from the optimal set. Assuming (ε_k) is summable, this does not impede convergence. For all these variants, analogues of the following convergence results hold true as well; however, the proofs require some extensions to account for the technical differences to the variant we present, which admitted the overall shortest proofs. In practice, we would generally expect these variants to behave similarly.

Moreover, if the adaptive approximate projections are realized using a convergent algorithm, a natural way to refine the accuracy is to simply continue the iterative process until the desired properties hold. Furthermore, note that in principle, the "problematic" cases could also be treated by reverting to exact projections ($\gamma \equiv 0$); however, in our present context this should be avoided since computing the exact projection is considered too expensive.

We obtain the following convergence results, depending on whether φ over- or underestimates f^*. The main proofs are deferred to the next subsection.

Theorem 4.5 (Convergence for dynamic step sizes with overestimation)**.** *Let the optimal point set X^* be bounded, $\varphi \geq f^*$, $0 < \lambda_k \leq \beta < 2$ for all k, and $\sum_{k=0}^{\infty} \lambda_k = \infty$. Let (ν_k) be a nonnegative sequence with $\sum_{k=0}^{\infty} \nu_k < \infty$, and let*

$$\begin{aligned}\overline{\varepsilon}_k := &- \left(\frac{\lambda_k (f_k - \varphi)}{\|h^k\|_2} + d_{X^*}(x^k) \right) \\ &+ \sqrt{\left(\frac{\lambda_k (f_k - \varphi)}{\|h^k\|_2} + d_{X^*}(x^k) \right)^2 + \frac{\lambda_k (2 - \lambda_k)(f_k - \varphi)^2}{\|h^k\|_2^2}}. \end{aligned} \tag{4.19}$$

If the subgradients h^k satisfy $0 < \underline{H} \leq \|h^k\|_2 \leq \overline{H} < \infty$ and (ε_k) satisfies $0 \leq \varepsilon_k \leq \min\{\overline{\varepsilon}_k, \nu_k\}$ for all k, then the following holds.

(i) *For any given $\delta > 0$ there exists some index K such that $f_K \leq \varphi + \delta$.*

(ii) *If additionally $f_k > \varphi$ for all k and if $\lambda_k \to 0$ for $k \to \infty$, then the sequence of objective function values (f_k) of the ISA iterates (x^k) converges to φ.*

(iii) *If in addition to the prerequisites from* (ii)*, $\sum_{k=0}^{\infty} \lambda_k^2 < \infty$ and $\lambda_k \geq \sum_{j=k}^{\infty} \varepsilon_k$ holds for all k, then the sequence of ISA iterates (x^k) converges to a point $\overline{x} \in \{\, x \in X : f(x) = \varphi \,\}$.*

Remark 4.6.

1. The sequence (ν_k) is a technicality needed in the proof to ensure $\varepsilon_k \to 0$. Note from (4.19) that $\overline{\varepsilon}_k > 0$ as long as ISA keeps iterating (in the main loop over k), since $f_k > \varphi$ is then guaranteed by the adaptive accuracy refinements and, by assumption, it holds that $0 < \lambda_k < 2$.

2. More precisely, part (i) of Theorem 4.5 essentially means that after a finite number of iterations, we reach a point x^k with $f^* - c \leq f_k \leq \varphi + \delta$, for any $c > 0$. If $\varphi < f_k \leq \varphi + \delta$, this point may still be infeasible, but the closer f_k gets to φ, the smaller $\bar{\varepsilon}_k$ becomes, i.e., the algorithm automatically increases the projection accuracy. On the other hand, termination in Step 6 implies that $f_k \geq f^*$ (since x^k is then feasible), and if some inner loop is infinite, then the refined projection points converge to a feasible point. Hence, for every $c > 0$, there is some integer $0 \leq L < \infty$ such that after the L-th accuracy refinement and replacement of x^k, $f_k \geq f^* - c$ holds.
3. Part (ii) shows what happens when all function values f_k stay above the overestimate φ of f^*—which particularly holds true *after* possible refinements, if all the accuracy refinement phases are finite (and no termination occurs)—and we impose $\lambda_k \to 0$ for $k \to \infty$: We eventually obtain f_k arbitrarily close to φ, with vanishing feasibility violation as $k \to \infty$. (Indeed, as part (iii) shows, we can also obtain convergence of the iterate sequence with some more technical effort.) Then, as well as in case of termination in Step 6 or convergence in a refinement phase ($\ell \to \infty$), it may be desirable to restart the algorithm using a smaller φ; see Section 4.4.3.
4. The conditions $\|h^k\|_2 \geq \underline{H} > 0$, for all k, in Theorem 4.5 imply that all subgradients used by the algorithm are nonzero. These conditions are often automatically guaranteed, for example, if X is compact and no unconstrained optimum of f lies in X. In this case, $\|h\|_2 \geq \underline{H} > 0$ for all $h \in \partial f(x)$ and $x \in X$; moreover, the same holds for a small enough open neighborhood of X. Also, the norms of the subgradients are bounded from above. Thus, if we start close enough to X and restrict ε_k to be small enough, the conditions of Theorem 4.5 are fulfilled. Another example in which the conditions are naturally satisfied is discussed in Section 4.6.

Theorem 4.7 (Convergence for dynamic step sizes with underestimation)**.** *Let the set of optimal points X^* be bounded, $\varphi < f^*$, $0 < \lambda_k \leq \beta < 2$ for all k, and $\sum_{k=0}^{\infty} \lambda_k = \infty$. Let (ν_k) be a nonnegative sequence with $\sum_{k=0}^{\infty} \nu_k < \infty$, let*

$$L_k := \frac{\lambda_k(2-\beta)(f_k - \varphi)}{\|h^k\|_2^2}\big(f^* - f_k + \frac{\beta}{2-\beta}(f^* - \varphi)\big), \tag{4.20}$$

and let

$$\tilde{\varepsilon}_k := -\left(\frac{\lambda_k(f_k - \varphi)}{\|h^k\|_2} + d_{X^*}(x^k)\right) + \sqrt{\left(\frac{\lambda_k(f_k - \varphi)}{\|h^k\|_2} + d_{X^*}(x^k)\right)^2 - L_k}. \tag{4.21}$$

If the subgradients h^k *satisfy* $0 < \underline{H} \leq \|h^k\|_2 \leq \overline{H} < \infty$ *and* (ε_k) *satisfies* $0 \leq \varepsilon_k \leq \min\{|\tilde{\varepsilon}_k|, \nu_k\}$ *for all* k*, then the following holds.*

(i) *For any given* $\delta > 0$*, there exists some* K *such that*

$$f_K \leq f^* + \frac{\beta}{2-\beta}(f^* - \varphi) + \delta.$$

(ii) *If additionally* $\lambda_k \to 0$*, then the sequence of objective function values* (f_k) *of the ISA iterates* (x^k) *converges to the optimal value* f^**.*

(iii) *If it additionally holds that* $\sum_{k=0}^{\infty} \lambda_k^2 < \infty$ *and* $\lambda_k \geq \sum_{j=k}^{\infty} \varepsilon_k$ *for all* k*, then the sequence of ISA iterates* (x^k) *converges to a point* $x^* \in X^*$*.*

Remark 4.8.

1. If $f_k \leq \varphi < f^*$, Steps 3–8 ensure that after a finite number of projection refinements x^k satisfies $\varphi < f_k$. Thus, the algorithm will never terminate in Step 6 and every refinement phase is finite.
2. Moreover, infeasible points x^k with $\varphi < f_k < f^*$ are possible. Hence, the inequality in Theorem 4.7 (i) may be satisfied too soon to provide conclusive information regarding solution quality. Interestingly, part (ii) shows that by letting the parameters (λ_k) tend to zero, one can nevertheless establish convergence to the optimal value f^* (and $d_X(x^k) \leq d_{X^*}(x^k) \to 0$, i.e., asymptotic feasibility).
3. Theoretically, small values of β yield smaller errors, while in practice this restricts the method to very small steps (since $\lambda_k \leq \beta$), resulting in slow convergence. This illustrates a typical kind of trade-off between solution accuracy and speed.
4. The use of $|\tilde{\varepsilon}_k|$ in Theorem 4.7 avoids conflicting bounds on ε_k in case $L_k > 0$. Because $0 \leq \varepsilon_k \leq \nu_k$ holds notwithstanding, $0 \leq \varepsilon_k \to 0$ is maintained.
5. The same statements on lower and upper bounds on $\|h^k\|_2$ as in Remark 4.6 apply in the context of Theorem 4.7.

Finally, we illustrate that with exact Polyak step sizes and additional assumptions on the adaptive approximate projections one can also get convergence from previously known results:

Theorem 4.9 (Convergence for exact Polyak step sizes with firmly nonexpansive approximate projections)**.** *Let the set of optimal points* X^* *be bounded. Furthermore, assume that* $\epsilon_k \to 0$ *and that the adaptive approximate projections* $P_X^{\varepsilon_k}$ *are also firmly nonexpansive (i.e., the mappings* $2P_X^{\varepsilon_k} - Id$ *are nonexpansive, cf. [16, Fact 1.3]). If the subgradients* h^k *satisfy* $0 < \underline{H} \leq \|h^k\| \leq \overline{H} < \infty$*, and the relaxation parameters*

obey $\xi \leq \lambda_k \leq 2 - \xi$ *for some* $\xi > 0$*, then the sequence of ISA iterates*

$$x^{k+1} := P_X^{\varepsilon_k}\left(x^k - \lambda_k \frac{f_k - f^*}{\|h^k\|^2} h^k\right),$$

converges to a solution of (4.1).

Proof. Since we assume that the $P_X^{\epsilon_k}$ are firmly nonexpansive, the algorithm now fits into the framework of subgradient algorithms in [16, Section 7]. Since $\epsilon_k \to 0$, it follows that the $P_X^{\epsilon_k}$ converge actively pointwise ([16, Definition 3.15]). Then, [16, Theorem 7.7] implies that the algorithm is focusing [16, Definition 3.7] and general results from [16] (e.g., Corollary 3.12 or 3.22) prove the claim. □

It should be noted that the additional assumption of firmly nonexpansiveness may be easy to fulfill in special cases, but this is not always the case. For instance, Section 4.5.4 below contains an example in which firmly nonexpansive projections are not readily available, while the one in Section 4.5.1 admits such projections.

4.3.1 Convergence Proofs

Let us start with a brief discussion of our approach to prove Theorems 4.5 and 4.7, in particular also with respect to typical procedures for feasible methods. In the rest of this section, α_k will always denote step sizes of the form (4.3).

Recall that in subgradient methods, the objective function values need not decrease monotonically. Hence, the key quantity in convergence proofs usually is the distance to the optimal set X^*. For ISA with dynamic step sizes (Algorithm 4.2), we have the following result concerning these distances.

Lemma 4.10. *Let* $x^* \in X^*$*. For the sequence of ISA iterates* (x^k)*, computed with step sizes* $\alpha_k = \lambda_k(f_k - \varphi)/\|h^k\|_2^2$*, it holds that*

$$\begin{aligned}\|x^{k+1} - x^*\|_2^2 \;\leq\; &\|x^k - x^*\|_2^2 + \varepsilon_k^2 + 2\left(\frac{\lambda_k(f_k - \varphi)}{\|h^k\|_2} + \|x^k - x^*\|_2\right)\varepsilon_k \\ &+ \frac{\lambda_k(f_k - \varphi)}{\|h^k\|_2^2}\big(\lambda_k(f_k - \varphi) - 2(f_k - f^*)\big). \end{aligned} \tag{4.22}$$

In particular, also

$$d_{X^*}(x^{k+1})^2 \leq d_{X^*}(x^k)^2 - 2\,\alpha_k(f_k - f^*) + (\alpha_k\|h^k\|_2 + \varepsilon_k)^2 + 2\,d_{X^*}(x^k)\,\varepsilon_k. \tag{4.23}$$

Proof. Plug (4.3) into (4.7) for $x = x^*$ and rearrange terms to obtain (4.22). If the optimization problem (4.1) has a unique optimum x^*, then obviously $\|x^k - x^*\|_2 = d_{X^*}(x^k)$ for all k, so (4.23) is identical to (4.22). Otherwise, note that since X^* is the intersection of the closed set X with the (sub-)level set $\{\, x : f(x) \leq f^* \,\}$ of the convex function f, X^* is closed (cf., for example, [135, Prop. 1.2.2, 1.2.6]) and the projection onto X^* is well-defined. Then, considering $x^* = \mathcal{P}_{X^*}(x^k)$, (4.7) becomes

$$\left\|x^{k+1} - \mathcal{P}_{X^*}(x^k)\right\|_2^2 \leq d_{X^*}(x^k)^2 - 2\alpha_k(f_k - f^*) + (\alpha_k\|h^k\|_2 + \varepsilon_k)^2 + 2\, d_{X^*}(x^k)\, \varepsilon_k.$$

Furthermore, because clearly $f(\mathcal{P}_{X^*}(x)) = f(\mathcal{P}_{X^*}(y)) = f^*$ for all $x, y \in \mathbb{R}^n$, and by definition of the Euclidean projection,

$$d_{X^*}(x^{k+1})^2 \;=\; \left\|x^{k+1} - \mathcal{P}_{X^*}(x^{k+1})\right\|_2^2 \;\leq\; \left\|x^{k+1} - \mathcal{P}_{X^*}(x^k)\right\|_2^2.$$

Combining the last two inequalities yields (4.23).

Moreover, note that these results continue to hold true if x^{k+1} is replaced in a projection refinement phase (starting in the next iteration $k+1$), since then only accuracy parameters smaller than ε_k are used. □

Typical convergence results are often derived by showing that the sequence $(\|x^k - x^*\|_2)$ is monotonically decreasing (for arbitrary $x^* \in X^*$) under certain assumptions on the step sizes, subgradients, etc. For instance, this is done in [4], where (4.22) with $\varepsilon_k = 0$ for all k is the central inequality, cf. [4, Prop. 2]. In our case, i.e., working with adaptive approximate projections as specified by (4.4), we can follow this principle to derive conditions on the projection accuracies (ε_k) which still allow for a (monotonic) decrease of the distances from the optimal set: If the last summand in (4.22) is negative, the resulting gap between the distances from X^* of subsequent iterates can be exploited to relax the projection accuracy, i.e., to choose $\varepsilon_k > 0$ without destroying monotonicity.

Naturally, to achieve feasibility (at least in the limit), we will need to have (ε_k) diminishing ($\varepsilon_k \to 0$ as $k \to \infty$). It will become clear that this, combined with summability ($\sum_{k=0}^{\infty} \varepsilon_k < \infty$) and with monotonicity conditions as described above, is already enough to extend the analysis to cover iterations with $f_k < f^*$, which may occur since we project inaccurately.

For different choices of the estimate φ of f^*, we will now derive the proofs of Theorems 4.5 and 4.7 via a series of intermediate results. Corresponding results for exact projections ($\varepsilon_k = 0$) can be found in [4]. In fact, on $\mathbb{R}^n$, our analysis for adaptive approximate projections improves on some of these earlier results (e.g., [4, Prop. 10] states convergence of some *sub*sequence of the function values to the optimum for the case $\varphi < f^*$, whereas our Theorem 4.7 gives convergence of the

whole sequence (f_k), for approximate and also for exact projections).

For the remainder of this section we can assume that ISA (Algorithm 4.2) does not terminate in Step 6 and that all inner projection accuracy refinement loops are finite. Otherwise, there is a refinement phase starting at some iteration $\underline{k}$ such that, as $\ell \to \infty$, $x^{\underline{k}}$ is repeatedly reset to

$$y_{\underline{k}}^{\ell} := \mathcal{P}_X^{\gamma^{\ell}\varepsilon_{\underline{k}-1}}(x^{\underline{k}-1} - \alpha_{\underline{k}-1}h^{\underline{k}-1}) \quad \to \quad \mathcal{P}_X^{0}(x^{\underline{k}-1} - \alpha_{\underline{k}-1}h^{\underline{k}-1}) \in X,$$

with $f(y_{\underline{k}}^{\ell}) \to \underline{\varphi} \leq \varphi$; cf. Remarks 4.6 and 4.8.

4.3.1.1 Using Overestimates of the Optimal Value

In this part we will focus on the case $\varphi \geq f^*$. As might be expected, this relation allows for eliminating the unknown f^* from (4.23).

Lemma 4.11. *Let $\varphi \geq f^*$ and $\lambda_k \geq 0$. If $f_k \geq \varphi$ for some $k \in \mathbb{N}$, then*

$$\begin{aligned} d_{X^*}(x^{k+1})^2 \; &\leq d_{X^*}(x^k)^2 + \varepsilon_k^2 + 2\left(\frac{\lambda_k(f_k-\varphi)}{\|h^k\|_2} + d_{X^*}(x^k)\right)\varepsilon_k \\ &\quad + \frac{\lambda_k(\lambda_k-2)(f_k-\varphi)^2}{\|h^k\|_2^2}. \end{aligned} \tag{4.24}$$

Proof. This is obtained immediately from Lemma 4.10, using $f_k \geq \varphi \geq f^*$ and $\lambda_k \geq 0$. □

Note that ISA guarantees $f_k > \varphi$ by sufficiently accurate projection (otherwise the method stops or the inner refinement loop over ℓ, with fixed k, is infinite, indicating φ was too large, see Steps 3–9 of Algorithm 4.2), and that the last summand in (4.24) is always negative for $0 < \lambda_k < 2$. Hence, adaptive approximate projections ($\varepsilon_k > 0$) can always be employed without destroying the monotonic decrease of $(d_{X^*}(x^k))$, as long as the ε_k are chosen small enough.

The following result provides a theoretical bound on how large the projection accuracies ε_k may become.

Lemma 4.12. *Let $0 < \lambda_k < 2$ for all k. For $\varphi \geq f^*$, the sequence $(d_{X^*}(x^k))$ is monotonically decreasing and converges to some $\zeta \geq 0$, if $0 \leq \varepsilon_k \leq \bar{\varepsilon}_k$ for all k, where $\bar{\varepsilon}_k$ is defined in (4.19) of Theorem 4.5.*

Proof. Considering (4.24), it suffices to show that for $\varepsilon_k \leq \bar{\varepsilon}_k$, we have

$$\varepsilon_k^2 + 2\left(\frac{\lambda_k(f_k-\varphi)}{\|h^k\|_2} + d_{X^*}(x^k)\right)\varepsilon_k + \frac{\lambda_k(\lambda_k-2)(f_k-\varphi)^2}{\|h^k\|_2^2} \;\leq\; 0. \tag{4.25}$$

The bound $\overline{\varepsilon}_k$ from (4.19) is precisely the (unique) positive root of the quadratic function in ε_k given by the left hand side of (4.25). Thus, we have a monotonically decreasing (i.e., nonincreasing) sequence $(d_{X^*}(x^k))$, and since its members are bounded below by zero, it converges to some nonnegative value, say ζ. □

Consequently, if X^* is bounded, we obtain boundedness of the iterate sequence (x^k).

Corollary 4.13. *Let X^* be bounded. If the sequence $(d_{X^*}(x^k))$ is monotonically decreasing, then the sequence (x^k) is bounded.*

Proof. By monotonicity of $(d_{X^*}(x^k))$, making use of the triangle inequality,

$$\begin{aligned}\|x^k\|_2 &= \left\|x^k - \mathcal{P}_{X^*}(x^k) + \mathcal{P}_{X^*}(x^k)\right\|_2 \\ &\leq d_{X^*}(x^k) + \left\|\mathcal{P}_{X^*}(x^k)\right\|_2 \leq d_{X^*}(x^0) + \sup_{x\in X^*}\|x\|_2 < \infty,\end{aligned}$$

since X^* is bounded by assumption. □

We now have all the tools at hand for proving Theorem 4.5.

Proof of Theorem 4.5. First, we show part (i). Let the main assumptions of Theorem 4.5 hold and suppose—contrary to the desired result (i)—that $f_k > \varphi + \delta$ for all k (possibly after finitely many refinements of the projection accuracy used to compute x^k). By Lemma 4.11,

$$\frac{\lambda_k(2-\lambda_k)(f_k-\varphi)^2}{\|h^k\|_2^2} \leq d_{X^*}(x^k)^2 - d_{X^*}(x^{k+1})^2 + \varepsilon_k^2 + 2\left(\frac{\lambda_k(f_k-\varphi)}{\|h^k\|_2} + d_{X^*}(x^k)\right)\varepsilon_k.$$

Since $0 < \underline{H} \leq \|h^k\|_2 \leq \overline{H} < \infty$, $0 < \lambda_k \leq \beta < 2$, and $f_k - \varphi > \delta$ for all k by assumption, we have

$$\frac{\lambda_k(2-\lambda_k)(f_k-\varphi)^2}{\|h^k\|_2^2} \geq \frac{\lambda_k(2-\beta)\delta^2}{\overline{H}^2}.$$

By Lemma 4.12, $d_{X^*}(x^k) \leq d_{X^*}(x^0)$. Also, by Corollary 4.13 there exists $F < \infty$ such that $f_k \leq F$ for all k. Hence, $\lambda_k(f_k - \varphi) \leq \beta(F-\varphi)$, and since $1/\|h^k\|_2 \leq 1/\underline{H}$, we obtain

$$\frac{(2-\beta)\delta^2}{\overline{H}^2}\lambda_k \leq d_{X^*}(x^k)^2 - d_{X^*}(x^{k+1})^2 + \varepsilon_k^2 + 2\left(\frac{\beta(F-\varphi)}{\underline{H}} + d_{X^*}(x^0)\right)\varepsilon_k. \quad (4.26)$$

Summation of the inequalities (4.26) for $k = 0, 1, \ldots, m$ yields

$$\frac{(2-\beta)\delta^2}{\overline{H}^2}\sum_{k=0}^{m}\lambda_k \;\le\; d_{X^*}(x^0)^2 - d_{X^*}(x^{m+1})^2 + \sum_{k=0}^{m}\varepsilon_k^2 + 2\left(\frac{\beta(F-\varphi)}{\underline{H}} + d_{X^*}(x^0)\right)\sum_{k=0}^{m}\varepsilon_k.$$

Now, by assumption, the left hand side tends to infinity as $m \to \infty$, while the right hand side remains finite (note that nonnegativity and summability of (ν_k) imply the summability of (ν_k^2), properties that carry over to (ε_k)). Thus, we have reached a contradiction, which proves part (i) of Theorem 4.5, i.e., that $f_K \le \varphi + \delta$ holds in some iteration K.

We now turn to part (ii): Besides the main assumptions of Theorem 4.5, let $\lambda_k \to 0$ and suppose $f_k > \varphi$ for all k (again, possibly after refinements). Then, since we know from part (i) that the function values fall below every $\varphi + \delta$, we can construct a monotonically decreasing subsequence (f_{K_j}) such that $f_{K_j} \to \varphi$. (To see this, note that if $f_k < \varphi + \delta$ is reached with $f_k < \varphi$, the ensuing refinement phase not necessarily ends with x^k replaced by a point with $\varphi < f_k < \varphi + \delta$. However, then there always exists a $K > k$ such that $\varphi < f_K < \varphi + \delta$, since $\lambda_k \to 0$, $\varepsilon_k \to 0$, and by continuity of f.)

To show that φ is the unique accumulation point of (f_k), assume to the contrary that there is another subsequence of (f_k) which converges to $\varphi + \eta$, with some $\eta > 0$. We can now employ the same technique as in the proof of Theorem 4.2 to reach a contradiction:

The two cases $f_k < \varphi + \frac{1}{3}\eta$ and $f_k > \varphi + \frac{2}{3}\eta$ must both occur infinitely often, since φ and $\varphi + \eta$ are accumulation points. Set $n_{(-1)} := -1$ and define, for $\ell \ge 0$,

$$m_\ell := \min\{\, k \,:\, k > n_{\ell-1},\, f_k > \varphi + \tfrac{2}{3}\eta \,\},$$
$$n_\ell := \min\{\, k \,:\, k > m_\ell,\, f_k < \varphi + \tfrac{1}{3}\eta \,\}.$$

Then, with $\infty > F \ge f_k$ for all k (such an F exists since (x^k) is bounded and, therefore, so is (f_k)) and the subgradient norm bounds, we obtain

$$\tfrac{1}{3}\eta \;<\; f_{m_\ell} - f_{n_\ell} \;\le\; \overline{H}\|x^{m_\ell} - x^{n_\ell}\|_2 \;\le\; \frac{\overline{H}(F-\varphi)}{\underline{H}}\sum_{j=m_\ell}^{n_\ell-1}\lambda_j + \overline{H}\sum_{j=m_\ell}^{n_\ell-1}\varepsilon_j$$

and from this, denoting $\ell_m := \max\{\,\ell \,:\, n_\ell - 1 \le m\,\}$ for a given m,

$$\tfrac{1}{3}\sum_{\ell=0}^{\ell_m}\eta \;\le\; \frac{\overline{H}(F-\varphi)}{\underline{H}}\sum_{\ell=0}^{\ell_m}\sum_{j=m_\ell}^{n_\ell-1}\lambda_j + \overline{H}\sum_{\ell=0}^{\ell_m}\sum_{j=m_\ell}^{n_\ell-1}\varepsilon_j.$$

Since for $m\to\infty$, the left hand side tends to infinity, the same must hold for the right hand side. But since $\sum_{\ell=0}^{\ell_m}\sum_{j=m_\ell}^{n_\ell-1}\varepsilon_j \le \sum_{k=0}^{m}\varepsilon_k \le \sum_{k=0}^{m}\nu_k < \infty$, this implies

$$\sum_{\ell=0}^{\ell_m}\sum_{j=m_\ell}^{n_\ell-1}\lambda_j \;\to\; \infty \qquad \text{for } m\to\infty. \tag{4.27}$$

Moreover, using the same estimates as in part (i) above, (4.24) yields

$$\underbrace{\tfrac{2-\beta}{\overline{H}}}_{=:C_1<\infty}(f_k-\varphi)^2\lambda_k \;\le\; d_{X^*}(x^k)^2 - d_{X^*}(x^{k+1})^2 + \varepsilon_k^2 + \underbrace{2\left(\tfrac{\beta(F-\varphi)}{\underline{H}} + d_{X^*}(x^0)\right)}_{=:C_2<\infty}\varepsilon_k$$

and hence, by summation for $k=0,\dots,m$ for a given m, it holds that

$$C_1\sum_{k=0}^{m}(f_k-\varphi)^2\lambda_k \;\le\; d_{X^*}(x^0)^2 - d_{X^*}(x^{m+1})^2 + \sum_{k=0}^{m}\varepsilon_k^2 + C_2\sum_{k=0}^{m}\varepsilon_k. \tag{4.28}$$

Observe that all summands of the left hand side term are positive, and thus

$$C_1\sum_{k=0}^{m}(f_k-\varphi)^2\lambda_k \;\ge\; C_1\sum_{\ell=0}^{\ell_m}\sum_{j=m_\ell}^{n_\ell-1}\underbrace{(f_j-\varphi)}_{>\frac{1}{3}\eta}{}^2\lambda_j \;>\; \frac{C_1\eta^2}{9}\sum_{\ell=0}^{\ell_m}\sum_{j=m_\ell}^{n_\ell-1}\lambda_j.$$

Therefore, as $m\to\infty$, the left hand side of (4.28) tends to infinity (by (4.27) and the above inequality) while the right hand side expression remains finite (recall $0\le\varepsilon_k\le\nu_k$ with (ν_k) summable and hence also square-summable). Thus, we have reached a contradiction, and it follows that φ is the only accumulation point (i.e., the limit) of the whole sequence (f_k), which proves part (ii).

Regarding part (iii) of the theorem, assume additionally that $\sum_{k=0}^{\infty}\lambda_k^2<\infty$ and that $\lambda_k \ge \sum_{j=k}^{\infty}\varepsilon_k$ for all k. We already saw that (x^k) is bounded (whence also, with some $F<\infty$, $f_k\le F$ for all k), and since moreover, $\varepsilon_k\to 0$ (and therefore $d_X(x^k)\to 0$), there exists a subsequence (x^{k_i}) of (x^k) that converges to some $\overline{x}\in X$. In fact, since $f_k\to\varphi$, it must hold that $f(\overline{x})=\varphi$. By (4.7) from Lemma 4.1, and due to the various properties of (λ_k), (f_k), (x^k), (ε_k) and the subgradient norms $(\|h^k\|_2)$,

it holds that, for any k,

$$\begin{aligned}\|x^{k+1}-\overline{x}\|_2^2 &\leq \|x^k-\overline{x}\|_2^2 - 2\alpha_k(f_k-\varphi) + (\alpha_k\|h^k\|_2+\varepsilon_k)^2 + 2\,\|x^k-\overline{x}\|_2\,\varepsilon_k\\ &= \|x^k-\overline{x}\|_2^2 + 2\frac{(f_k-\varphi)}{\|h^k\|_2}\lambda_k\varepsilon_k + \varepsilon_k^2 + 2\|x^k-\overline{x}\|_2\,\varepsilon_k\\ &\quad + \frac{(f_k-\varphi)^2}{\|h^k\|_2^2}\lambda_k(\lambda_k-2)\\ &\leq \|x^k-\overline{x}\|_2^2 + \underbrace{2\frac{(F-\varphi)}{\underline{H}}\lambda_k\varepsilon_k + \varepsilon_k^2 + 2\|x^k-\overline{x}\|_2\,\varepsilon_k}_{=:\overline{\beta}_k}\end{aligned}$$

Summation of these inequalities for $k=0,\dots,m$, for some $m\in\mathbb{N}$, yields

$$\|x^{m+1}-\overline{x}\|_2^2 \;\leq\; \|x^0-\overline{x}\|_2^2 + \sum_{k=0}^{m}\overline{\beta}_k.$$

Completely analogous to the derivation of (4.13) and (4.14) in the proof of Theorem 4.2, letting $\overline{D}\coloneqq\|x^0-\overline{x}\|_2$, we get

$$\begin{aligned}&\sum_{k=0}^{m}\|x^k-\overline{x}\|_2\varepsilon_k\\ &= \overline{D}\varepsilon_0 + \sum_{k=1}^{m}\left\|\mathcal{P}_X^{\varepsilon_{k-1}}\left(x^{k-1}-\lambda_{k-1}\frac{(f_{k-1}-\varphi)}{\|h^{k-1}\|_2^2}h^{k-1}\right)-\overline{x}\right\|_2\varepsilon_k\\ &\leq \overline{D}(\varepsilon_0+\varepsilon_1) + \sum_{k=0}^{m-1}\varepsilon_k\varepsilon_{k+1} + \sum_{k=1}^{m-1}\|x^k-\overline{x}\|_2\,\varepsilon_{k+1} + \frac{\overline{H}(F-\varphi)}{\underline{H}^2}\sum_{k=0}^{m-1}\lambda_k\varepsilon_{k+1}\\ &\leq \;\dots\; \leq\; \overline{D}\sum_{k=0}^{m}\varepsilon_k + \sum_{j=1}^{m}\left(\sum_{k=0}^{m-j}\varepsilon_k\varepsilon_{k+j} + \frac{\overline{H}(F-\varphi)}{\underline{H}^2}\sum_{k=0}^{m-j}\lambda_k\varepsilon_{k+j}\right)\\ &= \overline{D}\sum_{k=0}^{m}\varepsilon_k + \sum_{j=1}^{m}\sum_{k=0}^{m-j}\left(\varepsilon_k + \frac{\overline{H}(F-\varphi)}{\underline{H}^2}\lambda_k\right)\varepsilon_{k+j},\end{aligned}$$

and therefore, writing $E\coloneqq\sum_{k=0}^{\infty}\varepsilon_k$,

$$\begin{aligned}\sum_{k=0}^{m}\overline{\beta}_k \;&\leq 2\frac{(F-\varphi)}{\underline{H}}\sum_{k=0}^{m}\lambda_k\varepsilon_k + \sum_{k=0}^{m}\varepsilon_k^2 + 2\overline{D}\sum_{k=0}^{m}\varepsilon_k\\ &\quad + 2\sum_{j=1}^{m}\sum_{k=0}^{m-j}\left(\varepsilon_k + \frac{\overline{H}(F-\varphi)}{\underline{H}^2}\lambda_k\right)\varepsilon_{k+j}\end{aligned}$$

$$\leq 2\overline{D}\sum_{k=0}^{m}\varepsilon_k + 2\sum_{j=0}^{m}\left(\varepsilon_j\sum_{k=j}^{m}\varepsilon_k\right) + 2\frac{\overline{H}(F-\varphi)}{\underline{H}^2}\sum_{j=0}^{m}\left(\lambda_k\sum_{k=j}^{m}\varepsilon_k\right)$$
$$\leq 2(\overline{D}+E)\sum_{k=0}^{m}\varepsilon_k + 2\frac{\overline{H}(F-\varphi)}{\underline{H}^2}\sum_{k=0}^{m}\lambda_k^2.$$

Thus, $\sum_{k=0}^{\infty}\overline{\beta}_k < \infty$. From this, and since $x^{k_i} \to \overline{x}$, we see—just like in the proof of Theorem 4.2—that for any sufficiently large k_i and $K > k_i$, the right hand side of $\|x^K - \overline{x}\|_2^2 \leq \|x^{k_i} - \overline{x}\|_2^2 + \sum_{j=k_i}^{K-1}\overline{\beta}_j$ becomes arbitrarily small, which implies that $x^k \to \overline{x}$.

This shows part (iii) and thus completes the proof of Theorem 4.5. □

Note also that for $\overline{x}$ from Theorem 4.5, it holds that $d_{X^*}(\overline{x}) = \zeta \geq 0$, where ζ is the same as in Lemma 4.12.

4.3.1.2 Using Lower Bounds on the Optimal Value

In the following, we focus on the case $\varphi < f^*$, i.e., using a constant lower bound in the step size definition (4.3). Such a lower bound is often more readily available than (useful) upper bounds; for instance, it can be computed via the dual problem, or sometimes derived directly from properties of the objective function such as, e.g., nonnegativity of the function values.

Following arguments similar to those in the previous part, we can prove convergence of ISA (under certain assumptions), provided that the projection accuracies (ε_k) obey conditions analogous to those for the case $\varphi \geq f^*$. Moreover, recall that for $\varphi < f^*$, every refinement phase is finite (cf. Remark 4.8), so that $f_k > \varphi$ is guaranteed for all k; in particular, Step 6 of Algorithm 4.2 is never executed, since $X \cap \{x : f(x) < \varphi\} = \emptyset$.

Let us start with analogues of Lemmas 4.11 and 4.12.

Lemma 4.14. *Let $\varphi < f^*$ and $0 < \lambda_k \leq \beta < 2$. If $f_k \geq \varphi$ for some $k \in \mathbb{N}$, then*

$$d_{X^*}(x^{k+1})^2 \;\leq\; d_{X^*}(x^k)^2 + \varepsilon_k^2 + 2\left(\frac{\lambda_k(f_k-\varphi)}{\|h^k\|_2} + d_{X^*}(x^k)\right)\varepsilon_k + L_k, \qquad (4.29)$$

where $L_k = \frac{\lambda_k(2-\beta)(f_k-\varphi)}{\|h^k\|_2^2}(f^* - f_k + \frac{\beta}{2-\beta}(f^*-\varphi))$ as in (4.20) of Theorem 4.7.

Proof. For $\varphi < f^*$, $0 < \lambda_k \leq \beta < 2$, and $f_k \geq \varphi$, it holds that

$$\lambda_k(f_k-\varphi) - 2(f_k - f^*) \;\leq\; \beta(f_k-\varphi) - 2(f_k-f^*) \;=\; \beta(f^*-\varphi) + (2-\beta)(f^*-f_k).$$

The claim now follows immediately from Lemma 4.10 and (4.20). □

Lemma 4.15. *Let $\varphi < f^*$, let $0 < \lambda_k \leq \beta < 2$ and $f_k \geq f^* + \frac{\beta}{2-\beta}(f^* - \varphi)$ for all k, and let L_k be given by (4.20). Then $(d_{X^*}(x^k))$ is monotonically decreasing and converges to some $\xi \geq 0$, if $0 \leq \varepsilon_k \leq \tilde{\varepsilon}_k$ for all k, where $\tilde{\varepsilon}_k$ is defined in (4.21).*

Proof. The condition $f_k \geq f^* + \frac{\beta}{2-\beta}(f^* - \varphi)$ implies $L_k \leq 0$ and hence ensures that adaptive approximate projection can be used while still allowing for a decrease in the distances of the subsequent iterates from X^*. The rest of the proof is completely analogous to that of Lemma 4.12, considering (4.29) and (4.20) to derive the upper bound $\tilde{\varepsilon}_k$ given by (4.21) on the projection accuracy. □

We can now state the proof of our convergence results for the case $\varphi < f^*$.

Proof of Theorem 4.7. Let the main assumptions of Theorem 4.7 hold. We start with proving part (i): Let some $\delta > 0$ be given and suppose—contrary to the desired result (i)—that $f_k > f^* + \frac{\beta}{2-\beta}(f^* - \varphi) + \delta$ for all k (possibly after refinements). By Lemma 4.14,

$$d_{X^*}(x^{k+1})^2 \;\leq\; d_{X^*}(x^k)^2 + \varepsilon_k^2 + 2\left(\frac{\lambda_k(f_k - \varphi)}{\|h^k\|_2} + d_{X^*}(x^k)\right)\varepsilon_k + L_k.$$

Since $0 < \underline{H} \leq \|h^k\|_2 \leq \overline{H}$, $0 < \lambda_k \leq \beta < 2$, and $\varphi < f_k$, and due to our assumption

$$f^* - f_k + \frac{\beta}{2-\beta}(f^* - \varphi) \;<\; -\delta \qquad \text{for all } k,$$

it follows that

$$L_k \;<\; -\frac{\lambda_k(2-\beta)(f_k - \varphi)\delta}{\overline{H}^2} \;<\; 0.$$

By Lemma 4.15, $d_{X^*}(x^k) \leq d_{X^*}(x^0)$, and Corollary 4.13 again ensures existence of some $F < \infty$ such that $f_k \leq F$ for all k. Because also $\lambda_k(f_k - \varphi) \leq \beta(F - \varphi)$ and $1/\|h^k\|_2 \leq 1/\underline{H}$, we hence obtain

$$\begin{aligned}\frac{\lambda_k(2-\beta)(f_k - \varphi)\delta}{\overline{H}^2} \;<\; -L_k \;&\leq d_{X^*}(x^k)^2 - d_{X^*}(x^{k+1})^2 \\ &\quad + \varepsilon_k^2 + 2\left(\frac{\beta(F-\varphi)}{\underline{H}} + d_{X^*}(x^0)\right)\varepsilon_k. \qquad (4.30)\end{aligned}$$

Summation of these inequalities for $k = 0, 1, \ldots, m$ shows that

$$\frac{(2-\beta)\delta}{\overline{H}^2}\sum_{k=0}^{m}(f_k-\varphi)\lambda_k \;<\; d_{X^*}(x^0)^2 - d_{X^*}(x^{m+1})^2 + \sum_{k=0}^{m}\varepsilon_k^2 + 2\left(\frac{\beta(F-\varphi)}{\underline{H}} + d_{X^*}(x^0)\right)\sum_{k=0}^{m}\varepsilon_k. \tag{4.31}$$

Moreover, our assumption on f_k yields

$$f_k - \varphi \;>\; f^* + \frac{\beta}{2-\beta}f^* - \frac{\beta}{2-\beta}\varphi + \delta - \varphi \;=\; \frac{2}{2-\beta}(f^*-\varphi) + \delta.$$

It follows from (4.31) that

$$\frac{\big(2(f^*-\varphi)+(2-\beta)\delta\big)\delta}{\overline{H}^2}\sum_{k=0}^{m}\lambda_k < d_{X^*}(x^0)^2 - d_{X^*}(x^{m+1})^2 + \sum_{k=0}^{m}\varepsilon_k^2 + 2\left(\frac{\beta(F-\varphi)}{\underline{H}} + d_{X^*}(x^0)\right)\sum_{k=0}^{m}\varepsilon_k.$$

Now, by assumption, the left hand side tends to infinity as $m \to \infty$, whereas by Lemma 4.15 and the choice of $0 \le \varepsilon_k \le \min\{|\tilde{\varepsilon}_k|, \nu_k\}$ with a nonnegative summable (and hence also square-summable) sequence (ν_k), the right hand side remains finite. Thus, we have reached a contradiction, and part (i) is proven, i.e., there does exist some K such that $f_K \le f^* + \frac{\beta}{2-\beta}(f^*-\varphi) + \delta$ (possibly after refinements of the projection accuracy and recomputation of x^K).

Let us now turn to part (ii): Again, let the main assumptions of Theorem 4.7 hold and let $\lambda_k \to 0$. Recall that for $\varphi < f^*$, we have $f_k > \varphi$ for all k by construction of ISA (refinement loops). We distinguish three cases:

If $f_k < f^*$ holds for all $k \ge k_0$ for some k_0, then $f_k \to f^*$ is obtained immediately, just like in the proof of Theorem 4.2, by asymptotic feasibility.

On the other hand, if $f_k \ge f^*$ for all k larger than some k_0, then repeated application of part (i) yields a subsequence of (f_k) which converges to f^*: For any $\delta > 0$ we can find an index K such that $f^* \le f_K \le f^* + \frac{\beta}{2-\beta}(f^*-\varphi) + \delta$; thus, obviously, we get arbitrarily close to f^* if we choose β and δ small enough. However, we have the restriction $\lambda_k \le \beta$. But since $\lambda_k \to 0$, we may "restart" our argumentation if λ_k is small enough and replace β (and δ) with a smaller one. With the convergent subsequence constructed along these lines, we can then use the same technique as in the proof of Theorem 4.5 (ii) to show that (f_k) has no other accumulation point but f^*, whence $f_k \to f^*$ follows.

Finally, when both cases $f_k < f^*$ and $f_k \geq f^*$ occur infinitely often, we can proceed similarly to the proof of Theorem 4.2: The subsequence of function values below f^* converges to f^*, since $\varepsilon_k \to 0$. For the function values greater or equal to f^*, we assume that there is an accumulation point $f^* + \eta$ larger than f^*, deduce that an appropriate sub-sum of the λ_k's diverges and then sum up equation (4.30) for the respective indices (belonging to $\{\, k : f_k \geq f^* \,\}$) to arrive at a contradiction. Note that the iterate sequence (x^k) is bounded, due to Corollary 4.13 (for iterations k with $f_k \geq f^*$) and because the iterates with $\varphi < f_k < f^*$ stay within a bounded neighborhood of the bounded set X^*, since (ε_k) tends to zero and is summable. Hence, as f is continuous on a neighborhood of X (which contains all x^k from some k on), (f_k) is bounded as well and therefore must have at least one accumulation point. The only possibility left now is f^*, so we conclude $f_k \to f^*$.

Now consider part (iii): With $f_k \to f^*$ and $\varepsilon_k \to 0$, we obviously have that $d_{X^*}(x^k) \to 0$. Thus, the bounded sequence (x^k) has a subsequence (x^{k_i}) that converges to some $x^* \in X^*$. From (4.7) in Lemma 4.1 and the various properties of the parameter, point, or objective function value sequences involved, we obtain

$$\begin{aligned}
\|x^{k+1} - x^*\|_2^2 &\leq \|x^k - x^*\|_2^2 - 2\frac{(f_k - \varphi)(f_k - f^*)}{\|h^k\|_2^2}\lambda_k + \varepsilon_k^2 \\
&\quad + 2\frac{(f_k - \varphi)}{\|h^k\|_2}\lambda_k\varepsilon_k + \frac{(f_k - \varphi)^2}{\|h^k\|_2^2}\lambda_k^2 + 2\|x^k - x^*\|_2\,\varepsilon_k \\
&\leq \|x^k - x^*\|_2^2 + \varepsilon_k^2 + 2\|x^k - x^*\|_2\,\varepsilon_k \\
&\quad + 2\frac{(F - \varphi)}{\underline{H}}\lambda_k\varepsilon_k + \frac{(F - \varphi)^2}{\underline{H}^2}\lambda_k^2 + 2\frac{(f^* - \varphi)^2}{\underline{H}^2}\lambda_k,
\end{aligned}$$

since $-(f_k - \varphi)(f_k - f^*) < 0$ if and only if $f_k > f^*$, and $-(f_k - \varphi)(f_k - f^*) \leq (f^* - \varphi)^2$ otherwise. Now we can proceed completely analogously to the proof of Theorem 4.5(iii) to obtain convergence of the whole sequence (x^k) to x^*; for brevity, we omit the details.

This completes the proof of Theorem 4.7. □

4.4 Discussion: Extensions of the ISA Framework

In this section, we will discuss some extensions that are perhaps particularly relevant from a practical point of view. We start by considering approximate subgradients, which can be incorporated into both ISA variants with predetermined (Algorithm 4.1) or dynamic Polyak-type step sizes (Algorithm 4.2), respectively.

For the latter variant, we then illustrate how to obtain bounds on the projection accuracies that are independent of the (generally unknown) distance from the optimal set, and thus computable. We also sketch how to extend the method from using a constant estimate φ of the optimal objective function value f^* to employing variable target values, thus overcoming the need to know a priori whether $\varphi \geq f^*$ or $\varphi < f^*$.

4.4.1 Integration of ϵ-Subgradients

Sometimes, finding a subgradient can itself be nontrivial and thus require some computational effort. Moreover, numerical issues may lead to acquiring erroneous subgradients instead of exact ones. Thus, relaxing the subdifferential may simplify acquiring suitable subgradient directions.

In view of such possible difficulties arising in applications of subgradient methods, it is noteworthy that the ISA convergence analyses (Theorems 4.2, 4.5 and 4.7) can be extended to work with (so-called) *ϵ-subgradients* [25], i.e., when replacing $\partial f(x)$ by

$$\partial_\rho f(x) \coloneqq \{\, h \in \mathbb{R}^n \,:\, f(y) - f(x) \geq h^\top (y - x) - \rho \quad \forall\, y \in \mathbb{R}^n \,\}.$$

(To avoid confusion with the projection accuracy parameter ε, we use ρ instead of ϵ in spite of the moniker "ϵ-subgradients".) For instance, we immediately obtain the following result.

Corollary 4.16. *Let ISA with predetermined step sizes (Algorithm 4.1) choose $h^k \in \partial_{\rho_k} f(x^k)$ with $\rho_k \geq 0$ for all k. Under the assumptions of Theorem 4.2, if (ρ_k) is chosen summable ($\sum_{k=0}^{\infty} \rho_k < \infty$) and such that*

(i) *$\rho_k \leq \mu\, \alpha_k$ for some $\mu > 0$, or*

(ii) *$\rho_k \leq \mu\, \varepsilon_k$ for some $\mu > 0$,*

then the sequence of ISA iterates (x^k) converges to an optimal point.

Proof. The proof is analogous to that of Theorem 4.2; we will therefore only sketch the necessary modifications: Choosing $h^k \in \partial_{\rho_k} f(x^k)$ (instead of $h^k \in \partial f(x^k)$) adds the term $+2\alpha_k \rho_k$ to the right hand side of (4.7). If $\rho_k \leq \mu\, \alpha_k$ for some constant $\mu > 0$, the square-summability of (α_k) suffices: By upper bounding $2\alpha_k \rho_k$, the constant term $+2\mu A$ is added to the definition of R in (4.14). Similarly, $\rho_k \leq \mu\, \varepsilon_k$ does not impair convergence under the assumptions of Theorem 4.2, because then

the additional summand in (4.14) is

$$2\sum_{k=0}^{m}\alpha_k\rho_k \;\le\; 2\mu\sum_{k=0}^{m}\alpha_k\varepsilon_k \;\le\; 2\mu\sum_{k=0}^{m}\Big(\alpha_k\sum_{\ell=k}^{\infty}\varepsilon_k\Big) \;\le\; 2\mu\sum_{k=0}^{m}\alpha_k^2 \;\le\; 2\mu A.$$

The rest of the proof is almost identical, using R modified as explained above, and with some other minor changes where ρ_k-terms need to be considered (e.g., the term $+\rho_{m_\ell}$ is introduced in (4.15), yielding an additional sum in (4.16), which remains finite when passing to the limit because (ρ_k) is summable). □

Similar extensions are possible when using dynamic step sizes of the form (4.3). The upper bounds (4.19) and (4.21) for the projection accuracies (ε_k) will depend on (ρ_k) as well, which of course must be taken into account when extending the proofs of Theorems 4.5 and 4.7 accordingly. Then, summability of (ρ_k) (implying $\rho_k \to 0$) is enough to maintain convergence. In particular, one could again require $\rho_k \le \mu\,\varepsilon_k$ for some $\mu > 0$. We will not go into further detail here, since the extensions are straightforward.

4.4.2 Computable Bounds for the Distance to the Optimal Point Set

The results in Theorems 4.5 and 4.7 hinge on bounds $\bar{\varepsilon}_k$ and $\tilde{\varepsilon}_k$ on the projection accuracy parameters ε_k, respectively. These bounds depend on unknown information and therefore seem of little practical use such as, for instance, an automated accuracy control in an implementation of the dynamic step size ISA. While the quantity f^* can sometimes be replaced by estimates directly, it may generally be hard to obtain useful estimates for the distance of the current iterate to the optimal set. However, such estimates are available for certain classes of objective functions. We will sketch several examples in the following.

For instance, when f is a *strongly convex function*, i.e., there exists some constant $C > 0$ such that for all x, y and $\mu \in [0,1]$

$$f(\mu x + (1-\mu)y) \;\le\; \mu f(x) + (1-\mu)f(y) - C\,\mu(1-\mu)\|x-y\|_2^2,$$

one can use the following upper bound on the distance to the optimal set [150]:

$$d_{X^*}(x) \;\le\; \min\Big\{\sqrt{\tfrac{f(x)-f^*}{C}},\, \tfrac{1}{2C}\min_{h\in\partial f(x)}\|h\|_2\Big\}.$$

For functions f such that $f(x) \ge C\,\|x\|_2 - D$, with constants $C, D > 0$, one

can make use of $d_{X^*}(x) \leq \|x\|_2 + \frac{1}{C}(f^* + D)$, obtained by simply employing the triangle inequality. Another related example class is induced by coercive self-adjoint operators F, i.e., $f(x) \coloneqq \langle Fx, x\rangle \geq C\|x\|_2^2$ with some constant $C > 0$ and a scalar product $\langle\cdot,\cdot\rangle$. The (usually) unknown f^* appearing above may again be treated using estimates.

Yet another important class is comprised of functions which have a set of *weak sharp minima* [105] over X, i.e., for which there exists a constant $\mu > 0$ such that

$$f(x) - f^* \;\geq\; \mu\, d_{X^*}(x) \qquad \forall\, x \in X. \tag{4.32}$$

Using $d_{X^*}(x) \leq d_X(x) + d_{X^*}(\mathcal{P}_X(x))$ for $x \in \mathbb{R}^n$, we can then estimate the distance of x to X^* via the weak sharp minima property of f. For instance, knowing some $\varphi \leq f^*$ (e.g., a dual lower bound), (4.32) yields

$$d_{X^*}(x) \;\leq\; \frac{f(x) - \varphi}{\mu} \qquad \forall x \in X.$$

Thus, when the bounds on the distance to the optimal set derived from using the above inequality become too conservative (i.e., too large, resulting in very small $\tilde{\varepsilon}_k$-bounds, cf. (4.21)), one could try to improve the above bound by improving the lower estimate φ of f^*.

An important subclass of functions with weak sharp minima is composed of the *polyhedral functions*, i.e., f has the form

$$f(x) = \max\{\, a_i^\top x + b_i \;:\; 1 \leq i \leq N \,\},$$

where $a_i \neq 0$ for all i; the scalar μ is then given by

$$\mu = \min\{\, \|a_i^\top\|_2 \;:\; 1 \leq i \leq N \,\}.$$

In practice, one might have access to (problem-specific) estimates of $d_{X^*}(x)$; in [70], it is claimed that "for most problems" prior experience or heuristical considerations can be used to that end. For instance, if X is compact, the diameter of X leads to the (conservative) estimate $d_{X^*}(x) \leq \operatorname{diam}(X) + d_X(x)$.

4.4.3 Variable Target Values

From a practical viewpoint, it may be desirable to have an algorithm, using dynamic step sizes, that does not require the user to *know* a priori whether an estimate φ is larger or smaller than f^*, respectively. Moreover, relying on a constant estimate

may lead to overly small or large steps, which slows down the convergence process (and, with respect to ISA (Algorithm 4.2), can also lead to many projection accuracy refinement phases). Thus, a typical approach is to replace the constant estimate φ by *variable target values* φ_k. These target values are then updated in the course of the algorithm to increasingly better estimates of f^*, so that the dynamic step size (4.3) more and more resembles the "ideal" Polyak step size (which would use $\varphi = f^*$). In principle, such extensions are also possible for the ISA framework. We briefly describe the most important aspects in the following, complemented by a simple example algorithm of this type.

First, recall that Theorems 4.5 and 4.7 provide bounds on the projection accuracies (ε_k) needed for convergence; clearly, if it is unknown whether $\varphi_k \geq f^*$ or $\varphi_k < f^*$, one must therefore choose $0 \leq \varepsilon_k \leq \min\{\bar{\varepsilon}_k, |\tilde{\varepsilon}_k|, \nu_k\}$, with $\bar{\varepsilon}_k$ and $\tilde{\varepsilon}_k$ given by (4.19) and (4.21), respectively, and with (ν_k) as in the theorems.

Crucial for any variable target value method is the ability to eventually somehow recognize whether $\varphi_k \geq f^*$ or $\varphi_k < f^*$. If all iterates were feasible, this would amount to detecting $X \cap \{\, x \,:\, f(x) \leq \varphi_k \,\} \neq \emptyset$ (i.e., $f(x) \leq \varphi_k$, since $x \in X$), which implies $\varphi_k \geq f^*$, or recognizing $X \cap \{\, x \,:\, f(x) \leq \varphi_k \,\} = \emptyset$, to infer that $\varphi_k < f^*$; see, e.g., [70]. However, in the case of (possibly) infeasible iterates, $f_k \leq \varphi_k$ does not necessarily imply that φ_k is too large. On the other hand, viewing the ISA iterates x^k as points of the "relaxed" feasible set

$$\mathcal{B}_X^{\varepsilon_{k-1}} := \{\, x \,:\, x = y + z,\ y \in X,\ \|z\|_2 \leq \varepsilon_{k-1} \,\},$$

then $\mathcal{B}_X^{\varepsilon_k} \cap \{x \,:\, f(x) \leq \varphi_k\} = \emptyset$ also implies that $\varphi_k < f^*$, since $X \subseteq \mathcal{B}_X^{\varepsilon_k}$.

In view of Theorem 4.7, keeping φ_k constant once we recognized that $\varphi_k < f^*$ ensures convergence of (f_k) to f^* (in practice, it may nevertheless be desirable to further improve the estimate φ_k in order to avoid overly large steps in the vicinity of the optimum). The associated case $\mathcal{B}_X^{\varepsilon_k} \cap \{\, x \,:\, f(x) \leq \varphi_k \,\} = \emptyset$ can be detected in practice, see [70, Section III.C] for details in the case of a feasible method; these results are extensible to the ISA framework with appropriate modifications.

The other case, $\varphi_k \geq f^*$, could be detected, e.g., with the help of an estimate of the Lipschitz constant of f (recall that every convex function is locally Lipschitz-continuous, and useful estimates should usually be available) and the distances to X implied by the projection accuracies. More precisely, suppose f is Lipschitz-continuous with constant $L > 0$, i.e.,

$$|f(x) - f(y)| \ \leq\ L\,\|x - y\|_2 \qquad \forall\, x, y \in \mathbb{R}^n.$$

By (4.4) and since replacing x^k (Step 8 in a possible refinement phase in Algorithm 4.2) is always done using accuracies $\gamma^\ell \varepsilon_{k-1} < \varepsilon_{k-1}$, we know that

$\|x^k - \mathcal{P}_X(x^k)\|_2 \leq \varepsilon_{k-1}$ for all $k \geq 0$. Thus, for $x^k = \mathcal{P}_X^{\gamma^\ell \varepsilon_{k-1}}(x^{k-1} - \alpha_{k-1} h^{k-1})$ with $\ell \geq 0$, we have

$$|f(x^k) - f(\mathcal{P}_X(x^k))| \;\leq\; L\,\|x^k - \mathcal{P}_X(x^k)\|_2 \;=\; L\,d_X(x^k) \;\leq\; L\,\gamma^\ell \varepsilon_{k-1} \;\leq\; L\,\varepsilon_{k-1}.$$

Then, $x^k \in \mathcal{B}_X^{\varepsilon_{k-1}}$ and $f(x^k) \leq \varphi_{k-1} - L\varepsilon_{k-1}$ imply that $f(\mathcal{P}_X(x^k)) \leq \varphi_{k-1}$, showing that $\varphi_{k-1} \geq f^*$. Furthermore, convergence of (f_k) to φ_k as in Theorem 4.5, termination in Step 6 (with $h^k \neq 0$), or convergence to a function value smaller than or equal to φ_k in case of an infinite refinement phase ($\ell \to \infty$) would also indicate that $\varphi_k \geq f^*$, and thus should be taken into account accordingly. (Moreover, note that for Lipschitz-continuous f, $\mathcal{B}_X^{\varepsilon_k} \cap \{\, x : f(x) \leq \varphi_k + L\varepsilon_k \,\} = \emptyset$ implies $\varphi_k < f^*$.)

In the literature, various schemes have been considered as update rules for variable targets (φ_k), see, e.g., [17, 150, 120, 226, 193, 70, 154, 170]. In principle, many such rules could be straightforwardly used in, or adapted to, a variable target value ISA.

4.4.3.1 VTV-ISA for Lipschitz-Continuous Objectives

In the remainder of this section, we concretize the previous discussion by presenting a specific example of a variable target value ISA (VTV-ISA). The simple target update rule we use exploits Lipschitz-continuity (though this is not essential, see Remark 4.19 below) of the objective function and the convergence results for constant targets φ from Theorems 4.5 and 4.7. We summarize the scheme in Algorithm 4.3.

Remark 4.17. We point out a few aspects regarding Algorithm 4.3.

1. The target value φ_k in iteration k is defined by reducing the estimate $\overline{\varphi}_k$ of f^* by a constant $\tau > 0$. Variants of this choice are quite common in variable target methods (see, e.g., [193, 70, 120]); often, τ is itself occasionally adjusted depending on algorithmic progress. In the usual feasible setting, $\overline{\varphi}_k \geq f^*$ is easily maintained and the wish to achieve—if possible—a better objective value (than $\overline{\varphi}_k$) is intuitively realized by setting the target to $\overline{\varphi}_k - \tau$. Here, we follow the same idea, although $\overline{\varphi}_k \leq f^*$ does not necessarily always hold (cf. item 4 below); for simplicity, we keep τ constant throughout the algorithm.
2. If $f_k \leq \varphi_k - L\varepsilon_{k-1}$, we would know that $\varphi_k \geq f^*$; see the earlier discussion. However, the target value update (Step 3) employed in the VTV-ISA Algorithm 4.3 (see also Step 12) prevents this case. Indeed, were this to hold, we would have

   $$f_k \;\leq\; \varphi_k - L\varepsilon_{k-1} \;=\; \overline{\varphi}_k - L\varepsilon_{k-1} - \tau \;\leq\; f_k + L\varepsilon_{k-1} - L\varepsilon_{k-1} - \tau \;=\; f_k - \tau \;<\; f_k,$$

   a contradiction. Consequently, if $f_k \leq \varphi_k$ occurs (Step 5), then at the same

Algorithm 4.3 VARIABLE TARGET VALUE ISA for Lipschitz-continuous objectives

Input: starting point x^0, sequences (λ_k), (ε_k), parameters $\gamma \in (0,1)$, $\gamma_\varepsilon > 0$, $\tau > 0$, some $d_0 \geq d_X(x^0)$ and Lipschitz-constant $L > 0$ of f

Output: an (approximate) solution to (4.1)

1: initialize $k := 0$, $\ell = -1$, $x^{-1} := x^0$, $h^{-1} := 0$, $\alpha_{-1} := 0$, $\varepsilon_{-1} := d_0$, $\overline{\varphi}_{-1} := \infty$
2: **repeat**
3: update target value: $\overline{\varphi}_k := \min\{\overline{\varphi}_{k-1}, f_k + L\varepsilon_{k-1}\}$, $\varphi_k := \overline{\varphi}_k - \tau$
4: choose a subgradient $h^k \in \partial f(x^k)$ of f at x^k
5: **if** $f_k \leq \varphi_k$ **or** $h^k = 0$ **then**
6: **if** $x^k \in X$ **and** $h^k = 0$ **then**
7: stop (with x^k optimal, showing $\varphi = f^*$)
8: **if** $x^k \in X$ **and** $h^k \neq 0$**, or if** $\gamma^\ell \varepsilon_{k-1} < \gamma_\varepsilon$ **then**
9: set $\overline{\varphi}_k := \varphi_k$, $\varphi_k := \overline{\varphi}_k - \tau$, and go to Step 5
10: increment $\ell := \ell + 1$
11: reset $x^k := \mathcal{P}_X^\varepsilon(x^{k-1} - \alpha_{k-1}h^{k-1})$ for $\varepsilon = \gamma^\ell \varepsilon_{k-1}$
12: update $\overline{\varphi}_k := \min\{\overline{\varphi}_k, f_k + L\gamma^\ell \varepsilon_{k-1}\}$, $\varphi_k := \overline{\varphi}_k - \tau$
13: go to Step 4
14: compute step size $\alpha_k := \lambda_k(f_k - \varphi_k)/\|h^k\|_2^2$
15: compute the next iterate $x^{k+1} := \mathcal{P}_X^{\varepsilon_k}(x^k - \alpha_k h^k)$
16: reset $\ell := 0$ and increment $k := k + 1$
17: **until** a stopping criterion is satisfied

time also $f_k > \varphi_k - L\varepsilon_{k-1}$.

3. If $x^k \in X$ and $h^k \neq 0$ in Step 8, then $\varphi_k \geq f^*$ holds. In particular, Step 9 maintains $\overline{\varphi}_k \geq f^*$ and eventually leads to $f_k > \varphi_k$ so that no accuracy refinement is required.
4. On the other hand, if $\gamma^\ell \varepsilon_{k-1} < \gamma_\varepsilon$ in Step 8, then the ensuing target update in Step 9 may lead to either $\overline{\varphi}_k \geq f^*$ or $\overline{\varphi}_k < f^*$. This criterion is intended to recognize when a refinement loop will be infinite, which would translate into $\varphi_k \geq f^*$. Indeed, this idea worked correctly if $\overline{\varphi}_k \geq f^*$ after Step 9. The other case $\overline{\varphi}_k < f^*$, however, implies that also $\varphi_k < f^*$, and while every refinement loop would then, in principle, be finite, the criterion $\gamma^\ell \varepsilon_{k-1} < \gamma_\varepsilon$ may lead to premature termination and thus a further reduction of $\overline{\varphi}_k$ and φ_k. (Nevertheless, if $\varepsilon_k \to 0$, the case $f_k \leq \varphi_k$ will eventually never occur again. Then, $\overline{\varphi}_k$ and φ_k remain constant for all following iterations and Theorem 4.7 can be invoked. This reasoning is made precise in the proof of the convergence theorem below.)
5. The additional target update in Step 12 can lead to shorter refinement loops (which is certainly desirable), since now not only f_k but also φ_k is (possibly) updated, increasing the chances that $f_k > \varphi_k$.

Theorem 4.18 (Convergence of VTV-ISA (Algorithm 4.3))**.** *Let the optimal point set X^* be bounded, $0 < \lambda_k \leq \beta < 2$ for all k, $\sum_{k=0}^{\infty} \lambda_k = \infty$ and $\lambda_k \to 0$, and let (ν_k) be a nonnegative sequence with $\sum_{k=0}^{\infty} \nu_k < \infty$. Let $\overline{\varepsilon}_k$ be defined as in (4.19), and let $\tilde{\varepsilon}_k$ be defined as in (4.21), each with φ replaced by φ_k, respectively.*

If the subgradients h^k satisfy $0 < \underline{H} \leq \|h^k\|_2 \leq \overline{H} < \infty$ and (ε_k) satisfies $0 \leq \varepsilon_k \leq \min\{\overline{\varepsilon}_k, |\tilde{\varepsilon}_k|, \nu_k\}$ for all k, then the sequence (f_k) of objective function values of the VTV-ISA iterates (x^k) converges to the optimal value f^*.

Proof. Let the assumptions of Theorem 4.18 hold. First, we will show that there exists some $K < \infty$ such that $\varphi_K < f^*$. Suppose this is not true, i.e., $\varphi_k \geq f^*$ for all k. Now, we can distinguish two cases:

(i) There is some $\tilde{k}$ such that $\overline{\varphi}_k = \overline{\varphi}_{\tilde{k}}$ for all $k \geq \tilde{k}$, or equivalently, $\varphi_k = \varphi_{\tilde{k}} = \overline{\varphi}_{\tilde{k}} - \tau$ stays constant for all following iterations $k \geq \tilde{k}$. Then (w.l.o.g.), $f_k > \varphi_{\tilde{k}}$ for all $k \geq \tilde{k}$, i.e., no refinement loops are executed anymore. (Since $\varepsilon_k \to 0$, for k large enough and thus ε_k sufficiently small, Step 9 would otherwise be executed at the very beginning of a refinement loop, immediately contradicting the constancy of the target value.) Thus, we can apply Theorem 4.5 and conclude $f_k \to \varphi_{\tilde{k}}$. But this means that there exists some $j > \tilde{k}$ such that $f_j + L\varepsilon_{j-1} < \varphi_{j-1} + \tau = \overline{\varphi}_{j-1}$, so that the target update in Step 3 yields $\overline{\varphi}_j < \overline{\varphi}_{j-1}$ (and thus $\varphi_j < \varphi_{j-1}$), contradicting the fact that the target value stays constant.

(ii) There is some $\varphi \geq f^*$ such that $\varphi_k \to \varphi$. By construction, the sequences $(\overline{\varphi}_k)$ and (φ_k) are monotonically nonincreasing; in particular, $\varphi_{k+1} \leq \varphi_k$ for all k. Moreover, the refinement loop ensures that $f_k > \varphi_k$ for all k. Now, it is easy to see that the proof of Theorem 4.5 can be straightforwardly modified to show that we also have $f_k \to \varphi$. However, as $\varepsilon_k \to 0$, this would imply that $\overline{\varphi}_k \to \varphi$ as well, whence by construction $\varphi_k \to \varphi - \tau < \varphi$, a contradiction.

Hence, we have established that for some $K < \infty$, it holds that $\varphi_k < f^*$ for all $k \geq K$. Moreover, since $\varepsilon_k \to 0$ and by continuity of f, there exists some $\tilde{K} \geq K$ such that $f_k = f(\mathcal{P}_X^{\varepsilon_{k-1}}(x^{k-1} - \alpha_{k-1}h^{k-1})) > \varphi_k$ for all $k \geq \tilde{K}$, i.e., from iteration $\tilde{K}$ onwards, the target remains at the constant value $\varphi_{\tilde{K}} < f^*$. (Before, even though $\varphi_k < f^*$, it might have been further reduced by premature refinement loop termination in Step 9.) In particular, no more refinement loops can occur. Now, we can invoke Theorem 4.7 and conclude that $f_k \to f^*$, as claimed. □

Remark 4.19. In fact, one could modify Algorithm 4.3 as follows: Setting $\gamma_\varepsilon = \infty$ essentially disables any projection accuracy refinements, instead simply reducing the target value each time that $f_k \leq \varphi_k$ occurs. That way, $f_k > \varphi_k$ is always achieved after finitely many target updates (Step 9), and eventually, $\varphi_k < f^*$ holds and remains constant from some $K < \infty$ onwards. Then, convergence of $f_k \to f^*$ can

be achieved under the same assumptions as in Theorem 4.18 and along the same lines as in its proof. Moreover, note that the Lipschitz-continuity assumption is not essential either: Instead of trying to keep an upper bound $\overline{\varphi}_k$ of f^* (although after premature refinement loop termination, this property might no longer hold anyway), we could just use any constant initial $\overline{\varphi}_0 \in \mathbb{R}$, $\varphi_0 = \overline{\varphi}_0 - \tau$, and remove the Lipschitz-dependent terms ("setting" $L = 0$).

Nevertheless, contrary to the purpose of working with variable target values in the first place, these simplified schemes can lead to a behavior much more similar to that using just some constant target $\varphi < f^*$ to begin with: If (early) iterates are far away from X, they may have $f_k \ll f^*$, which would lead to $\varphi_k \ll f^*$ very early on.

4.5 Examples of Adaptive Approximate Projection Operators

In this section, we give several detailed examples for constructing adaptive approximate projections that can achieve any given projection accuracy. In view of the main topic of this thesis, finding sparse solutions of underdetermined linear equation systems, we first consider the feasible set of problems like (P_0) and (P_1). Then, we discuss approximate projection onto (full-dimensional) ellipsoids, before turning to the constraint set of the denoising problems (P_0^δ) and (P_1^δ) (which can be viewed as a degenerate ellipsoid) and variants incorporating other norms. Finally, we will see a probabilistic construction of a projection operator that addresses convex expected-value constraints.

4.5.1 Linear Equality Constraints

Consider the problem of projection onto the feasible set $X \subseteq \mathbb{R}^n$ given by

$$X \coloneqq \{\, x \,:\, Ax = b \,\},$$

where $A \in \mathbb{R}^{m \times n}$ with $\operatorname{rank}(A) = m < n$. For a given point z, the projection onto X can be explicitly formulated as

$$\mathcal{P}_X(z) = z - A^\top (AA^\top)^{-1}(Az - b). \tag{4.33}$$

We can split this computation into two steps, namely

$$\begin{aligned} v^* &:= \text{the solution to } AA^\top v = Az - b, \\ \mathcal{P}_X(z) &= z - A^\top v^*. \end{aligned} \tag{4.34}$$

Since A has full rank, $AA^\top$ is symmetric and positive definite. Hence, we can apply (for instance) the well-known method of conjugate gradients (CG) [133] to solve the system in (4.34); cf. Section 1.3 and, for details, the references therein. In exact arithmetic, CG terminates with the exact solution after at most m iterations. Thus, letting

$$v^{(k)} := \text{the approx. solution to } AA^\top v = Az - b \text{ after } k \text{ CG iterations}, \tag{4.35}$$

we have, in particular, $v^* = v^{(m)}$.

Computing the projection onto X exactly can be very costly, practically prohibiting methods relying on fast projections for such linearly constrained problems. This motivates the idea to use the CG procedure to obtain approximate projections. (Another advantage is that for CG, one does not need to explicitly build $AA^\top$, which can be dense even if A itself is very sparse.) In particular, the question we investigate is: How does a *truncated* CG procedure (which does not run the full m iterations theoretically needed for exact solution of the projection problem) fit into our framework (4.4)? The answer is given by the following result.

Proposition 4.20. *Let $X := \{ x : Ax = b \}$ for $A \in \mathbb{R}^{m\times n}$ with $\operatorname{rank}(A) = m < n$ and $b \in \mathbb{R}^m$. Then the mapping $\mathcal{P}_X^\varepsilon$ defined by*

$$z \mapsto \mathcal{P}_X^\varepsilon(z) := z - A^\top v^{(k_\varepsilon)}, \tag{4.36}$$

where $v^{(k_\varepsilon)}$ is the approximate solution to $AA^\top v = Az - b$ obtained after k_ε CG iterations (with an arbitrary starting point $v^{(0)} \in \mathbb{R}^m$) and

$$k_\epsilon := \arg\min \left\{ k : \|AA^\top v^{(k)} - (Az - b)\|_2 \le \sigma_{\min}(A)\, \varepsilon \right\},$$

is an adaptive approximate projection operator for X in the sense of (4.4).

Proof. For $v^{(k)}$ as defined in (4.35), denote the corresponding residual by

$$r^{(k)} := AA^\top v^{(k)} - (Az - b).$$

The proposition states that, for our purpose, we can stop the CG iteration (prematurely) as soon as the residual norm $\|r^{(k)}\|_2$ becomes "small enough". Specifically,

we can bound the distance of the point $z - A^\top v^{(k)}$ to the exact projection of z, using (4.33) and the Cauchy-Schwarz inequality:

$$\begin{aligned}\|(z - A^\top v^{(k)}) - \mathcal{P}_X(z)\|_2 &= \|z - A^\top v^{(k)} - (z - A^\top (AA^\top)^{-1}(Az - b))\|_2 \\ \leq \|A^\top (AA^\top)^{-1}\|_2 \, \|AA^\top v^{(k)} - (Az - b)\|_2 &= \|A^\dagger\|_2 \, \|r^{(k)}\|_2 = \frac{\|r^{(k)}\|_2}{\sigma_{\min}(A)}.\end{aligned}$$

Consequently, stopping the CG method as soon as, for a given projection accuracy ε, it holds that

$$\|r^{(k)}\|_2 \leq \sigma_{\min}(A)\, \varepsilon \tag{4.37}$$

ensures that the procedure given by (4.36) yields an adaptive approximate projection operator of type (4.4). □

Remark 4.21. It is easy to see that the above proof is independent of the algorithm used to compute $v^{(k_\varepsilon)}$; in particular, the requirement (4.37) on the respective residual always ensures conformity to (4.4) of the constructed adaptive projection operator. Therefore, in principle, one could use any suitable iterative and convergent algorithm in the above setting, although the CG method is perhaps the most natural choice (and has the practically desirable property of not needing $AA^\top$ explicitly).

4.5.1.1 Connection with Related Notions of Approximate Projection

In the context of the subgradient method proposed in [201], approximate projections onto a closed convex set C are called *feasibility operators*, denoted by $\mathcal{F}_C$, and need to obey the following criterion, which is closely related to Féjer-monotonicity (but slightly more restrictive):

$$\forall\, \delta > 0 \quad \exists\, \varepsilon_\delta > 0 \quad \forall\, z,\ d_C(z) \geq \delta : \quad \|\mathcal{F}_C(z) - x\|^2 \leq \|z - x\|^2 - \varepsilon_\delta \quad \forall\, x \in C. \tag{4.38}$$

Moreover, it is required that $\mathcal{F}_C(x) = x$ for all points $x \in C$. Thus, (4.38) ensures that a feasibility operator produces a point that is closer to *every* (feasible) point in C than the (infeasible) unprojected point $z \notin C$. The technical, or conceptual, differences between the inexact projections based on (4.4) and (4.38), respectively, were discussed earlier in Section 4.1.2.

It turns out that a truncated CG procedure with *any fixed number* of iterations yields a feasibility operator in the sense of (4.38).

Proposition 4.22. *Let $K \in [m]$ be arbitrary and let $v^{(K)}$ denote the approximate solution to $AA^\top v = Az - b$ obtained after K CG iterations using the starting point $v^{(0)} = 0$. Then, $\mathcal{F}_X^K(z) := z - A^\top v^{(K)}$ is a feasibility operator satisfying (4.38).*

Proof. Note that by construction, for any integer $K \geq 0$, the point $\mathcal{F}_X^K(z)$ lies in the subspace $z+\mathcal{R}(A^\top)$. Moreover, $\mathcal{R}(A^\top)$ corresponds to the normal cone of X (at any feasible point and therefore, in particular, at $\mathcal{P}_X(z)$ for any $z \in \mathbb{R}^n$; cf. page 59). Hence, $\mathcal{F}_X^K(z)$ is obtained by taking a step along a normal direction, thus reducing the distance to *every* point in X if and only if that to $\mathcal{P}_X(z)$ is reduced. Therefore, (4.38) can be verified directly by showing that for $K \geq 1$, we always obtain a point that is strictly closer to $\mathcal{P}_X(z)$ than z, and that the difference of these distances is bounded by some ε_δ, i.e., in dependence of the lower distance bound δ of z to the feasible set. We proceed to do just that.

With starting point $v^{(0)} = 0$, we have for all $k = 1, \dots, K$ that

$$v^{(k)} \in \operatorname{span}\{Az-b, (AA^\top)(Az-b), \dots, (AA^\top)^{k-1}(Az-b)\} =: \mathcal{K}_k,$$

i.e., $v^{(k)}$ lies in the k-th Krylov subspace $\mathcal{K}_k$ associated with the system we wish to solve. Moreover, the CG method is applied to a certain type of normal equation here, and $v^{(k)}$ minimizes the energy norm (w.r.t. $AA^\top$) of the error over $\mathcal{K}_k$, cf. [227], [222, Section 8.3.2]. More precisely, with the exact solution $v^* = (AA^\top)^{-1}(Az-b)$ (as in (4.34)), it holds that

$$\begin{aligned} v^{(k)} &= \arg\min_{v\in\mathcal{K}_k} (v^*-v)^\top AA^\top (v^*-v) = \arg\min_{v\in\mathcal{K}_k} \|A^\top(v^*-v)\|_2^2 \\ &= \arg\min_{v\in\mathcal{K}_k} \|A^\top\big((AA^\top)^{-1}(Az-b)-v\big)\|_2^2 = \arg\min_{v\in\mathcal{K}_k} \|A^\top v - A^\dagger(Az-b)\|_2^2, \end{aligned}$$

and in fact, writing $\psi(v) \coloneqq \|A^\top v - A^\dagger(Az-b)\|_2^2$, we have (cf. [133, Section 6])

$$\psi(v^{(k)}) < \psi(v^{(\ell)}) \quad \text{for all } k > \ell \ (k \leq m). \tag{4.39}$$

By (4.33), the distance of z to X is $d_X(z) = \|A^\dagger(Az-b)\|_2$. Furthermore,

$$\|\mathcal{F}_X^K(z) - \mathcal{P}_X(z)\|_2^2 = \|z - A^\top v^{(K)} - z + A^\dagger(Az-b)\|_2^2 = \psi(v^{(K)}),$$

so that in particular, with $v^{(0)} = 0$, we obtain

$$\psi(v^{(0)}) = \|A^\top v^{(0)} - A^\dagger(Az-b)\|_2^2 = \|A^\dagger(Az-b)\|_2^2 = d_X(z)^2.$$

Therefore, due to (4.39),

$$\|\mathcal{F}_X^K(z) - \mathcal{P}_X(z)\|_2^2 < d_X(z)^2 \quad \text{for all } K \in [m],$$

and $\mathcal{F}_X^K(z)$ is in fact closer to *every* feasible point $x \in X$ than z (recall the discussion at the beginning of this proof).

Moreover, since $\psi(v^{(k)})$ is strictly monotonically decreasing in k, we can bound the reduction ε_δ in distance to X from below by that achieved in the first of $K \geq 1$ CG iterations, namely,

$$\varepsilon_\delta \geq \frac{\|Az-b\|_2^4}{\|A^\top(Az-b)\|_2^2}.$$

To see this, note that with starting point $v^{(0)} = 0$, the first CG iteration produces $v^{(1)} = (\|Az-b\|_2^2/\|A^\top(Az-b)\|_2^2)(Az-b)$, and we therefore obtain

$$\begin{aligned}
\psi(v^{(1)}) &= \|\mathcal{F}_X^1(z) - \mathcal{P}_X(z)\|_2^2 = \|z - A^\top v^{(1)} - (z - A^\dagger(Az-b))\|_2^2 \\
&= \left\| z - \frac{\|Az-b\|_2^2}{\|A^\top(Az-b)\|_2^2} A^\top(Az-b) - z + A^\top(AA^\top)^{-1}(Az-b) \right\|_2^2 \\
&= \left\| \frac{\|Az-b\|_2^2}{\|A^\top(Az-b)\|_2^2} A^\top(Az-b) \right\|_2^2 + d_X(z)^2 - 2\frac{\|Az-b\|_2^4}{\|A^\top(Az-b)\|_2^2} \\
&= d_X(z)^2 - \frac{\|Az-b\|_2^4}{\|A^\top(Az-b)\|_2^2}.
\end{aligned}$$

This completes the proof. □

Remark 4.23. It should be noted that using the construction suggested in [201, Section 3.2], we can obtain the feasibility operator

$$\mathcal{F}_X(z) = \begin{cases} z - \nu \frac{\|Az-b\|_2^2}{\|A^\top(Az-b)\|_2^2} A^\top(Az-b), & z \notin X \\ z, & z \in X \end{cases}$$

by rewriting the constraint $Ax = b$ as $\frac{1}{2}\|Ax-b\|_2^2 \leq 0$. For every $\nu \in (0,2)$, this construction obeys (4.38). It can easily be seen that $\mathcal{F}_X(z)$ differs from $\mathcal{F}_X^1(z)$ only in the relaxation factor ν, and that the value of ν for which $\mathcal{F}_X(z) = \mathcal{F}_X^1(z)$, i.e., $\nu = 1$, is in fact the one that minimizes the distance from $\mathcal{F}_X(z)$ to $\mathcal{P}_X(z)$. (Recall also that the first step of the CG method corresponds to a specific gradient descent step [227], whence taking $z - A^\top v^{(1)}$ is the same as using $\nu = 1$ above (for $z \notin X$).)

Moreover, note that the feasibility operator construction from [201, Section 3.2] corresponds precisely to what is called "subgradient projections" in the context of convex feasibility problems (see, e.g., [16] and the discussion of different approaches to approximate projections in Section 4.1.2). Such projections can often offer a very simple way to construct feasibility operators; however, we shall see an example where this approach is not applicable later, in Section 4.5.4.

Remark 4.24. Using a truncated CG-procedure, we are in fact in the general framework of [16]. Therefore, if $\varphi = f^*$, Theorem 4.9 is also applicable when employing

the adaptive approximate projection from Proposition 4.22 (even with $K = 1$).

4.5.2 Ellipsoids

Ellipsoidal (quadratic) constraints form another important class of constraints in optimization with many practical applications. Consider an ellipsoid

$$E \coloneqq \{\, y \,:\, y^\top S y \le \alpha \,\} \subset \mathbb{R}^n,$$

where $S \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix and $\alpha > 0$. In the following, we shall assume w.l.o.g. that $\alpha = 1$ (other values can be dealt with by replacing S with $\frac{1}{\alpha}S$).

For a given point $w \in \mathbb{R}^n$, the projection problem onto E is

$$\mathcal{P}_E(w) \coloneqq \arg\min_y \|w - y\|_2 \quad \text{s.t.} \quad y \in E. \tag{4.40}$$

A Lagrangean approach to (4.40), see, e.g., [151], leads to

$$\mathcal{P}_E(w) = (\mu^* S + I)^{-1} w, \tag{4.41}$$

where $\mu^* > 0$ is the unknown Lagrange multiplier whose value must be chosen such that the projected point lies on the boundary of the ellipsoid, i.e.,

$$\mathcal{P}_E(w)^\top S\, \mathcal{P}_E(w) = 1.$$

For an adaptive approximate projection, we can employ sequences (μ_k) and (μ^k) such that

$$\mu_k \le \mu^* \le \mu^k \quad \text{for all } k, \tag{4.42}$$

and

$$\mu_k \nearrow \mu^* \text{ and } \mu^k \searrow \mu^* \text{ monotonically as } k \to \infty. \tag{4.43}$$

In [151], three algorithms were suggested for computing μ^*. The first two (quadratically convergent) algorithms apply a Newton scheme to compute sequences of lower and upper bounds, respectively, which have the desired properties. The third algorithm is a type of nonlinear Newton scheme which empirically outperforms both other algorithms (see the experiments in the cited work); there, another sequence approaching μ^* from below is computed, which could be used for our purposes as well.

Define $\hat{\mu}_k \coloneqq (\mu^k + \mu_k)/2$; then, clearly, $\hat{\mu}_k \to \mu^*$ as $k \to \infty$, and

$$|\hat{\mu}_k - \mu^*| \leq \mu^k - \hat{\mu}_k = \hat{\mu}_k - \mu_k = \frac{1}{2}(\mu^k - \mu_k). \tag{4.44}$$

Moreover, we shall denote the exact projection (4.41) by y^* and the sequential approximations to y^* based on $(\hat{\mu}_k)$ (and thus (μ_k) and (μ^k)) by

$$y^k \coloneqq (\hat{\mu}_k S + I)^{-1} w.$$

In the following lemma, we gather some auxiliary results and observations.

Lemma 4.25. *Let S, μ^*, and $\hat{\mu}_k$ be defined as above. The following statements hold:*

(i) $(\hat{\mu}_k S + I)^{-1} - (\mu^* S + I)^{-1}$ *is symmetric.*

(ii) *For any $\mu > 0$, all eigenvalues of $(\mu S + I)^{-1}$ are contained in $(0, 1)$.*

(iii) *For any invertible matrices A and B, it holds that $A^{-1} + B^{-1} = A^{-1}(A + B)B^{-1}$.*

Proof. The first statement is trivial: $\mu S + I$ is clearly symmetric for any $\mu \in \mathbb{R}$, hence so is its inverse if it exists (which it does, e.g., if $\mu \geq 0$, and in particular for $\mu^*, \hat{\mu}_k > 0$), and the sum of two symmetric matrices is again symmetric. Regarding (ii), observe that the eigenvalues of $\mu S + I$ are precisely $\mu\lambda_i + 1$, $i = 1, \ldots, n$, where λ_i is the i-th eigenvalue of S. Thus, the eigenvalues of the inverse are $0 < 1/(1 + \mu\lambda_i) < 1$ for all i, since $\lambda_i > 0$ (by the positive definiteness of S) and $\mu > 0$. The third statement is easily checked; a proof can be found in [225, p.151]. □

The following result specifies an adaptive approximate projection onto ellipsoids that fits our framework.

Proposition 4.26. *Let $E \coloneqq \{\, y \,:\, y^\top S y \leq 1 \,\}$ for $S \in \mathbb{R}^{n \times n}$ symmetric positive definite. Then the mapping $\mathcal{P}_E^\varepsilon$ defined by*

$$w \mapsto \mathcal{P}_E^\varepsilon(w) \coloneqq y^{k_\varepsilon} = (\hat{\mu}_{k_\varepsilon} S + I)^{-1} w,$$

where

$$k_\varepsilon \coloneqq \arg\min \left\{ k \,:\, \mu^k - \mu_k \leq \left(\frac{2}{\lambda_{\max}(S)\, \|w\|_2} \right) \varepsilon \right\}$$

and, for every $k \geq 0$, $\hat{\mu}_k \coloneqq (\mu^k + \mu_k)/2$ with sequences (μ_k) and (μ^k) obeying (4.42) and (4.43) (e.g., generated by the respective algorithms from [151]), is an adaptive approximate projection operator for E in the sense of (4.4).

Proof. Let S, w and ε be given. Then,

$$\begin{aligned}
\|y^k - y^*\|_2 &\leq \|(\hat{\mu}_k S + I)^{-1} w - (\mu^* S + I)^{-1} w\|_2 \\
&= \|\big((\hat{\mu}_k S + I)^{-1} - (\mu^* S + I)^{-1}\big) w\|_2 \\
&= \|\big((\hat{\mu}_k S + I)^{-1}((\hat{\mu}_k - \mu^*)S + I - I)(-\mu^* S - I)^{-1}\big) w\|_2 \\
&= |\hat{\mu}_k - \mu^*| \cdot \|(\hat{\mu}_k S + I)^{-1} S (-\mu^* S - I)^{-1} w\|_2 \\
&\leq |\hat{\mu}_k - \mu^*| \cdot \|(\hat{\mu}_k S + I)^{-1}\|_2 \, \|S\|_2 \, \|-(\mu^* S + I)^{-1}\|_2 \, \|w\|_2 \\
&\leq \frac{1}{2}(\mu^k - \mu_k)\lambda_{\max}(S)\|w\|_2.
\end{aligned}$$

(The second equality is due to Lemma 4.25(iii), the second inequality is an application of the Cauchy-Schwarz inequality, and the last inequality follows from Lemma 4.25(ii), (4.44), and since for symmetric matrices, the spectral norm equals the largest eigenvalue.)

Clearly, we therefore have

$$\mu^{k_\varepsilon} - \mu_{k_\varepsilon} \leq \left(\frac{2}{\lambda_{\max}(S)\|w\|_2}\right)\varepsilon \qquad \Rightarrow \qquad \|\mathcal{P}_E^\varepsilon(w) - \mathcal{P}_E(w)\|_2 = \|y^{k_\varepsilon} - y^*\|_2 \leq \varepsilon,$$

which concludes the proof. □

Other specialized algorithms for projection onto ellipsoids exist; see, e.g., [74] for ellipsoids of the (more general) form $\{\, y \,:\, y^\top Q y + q^\top y \leq \alpha \,\}$ with Q symmetric positive definite. Moreover, one could of course employ generic solvers for quadratically constrained convex quadratic programs (see, e.g., [38]), since the projection problem (4.40) is obviously of this form.

4.5.3 Denoising Constraints

Recall the Basis Pursuit Denoising problem

$$\min \|x\|_1 \quad \text{s.t.} \quad \|Ax - b\|_2 \leq \delta, \tag{P_1^δ}$$

where $A \in \mathbb{R}^{m \times n}$ with $\operatorname{rank}(A) = m \leq n$, $b \in \mathbb{R}^m$ and $\delta > 0$. Note that the feasible set

$$X_\delta := \{\, x \,:\, \|Ax - b\|_2 \leq \delta \,\} = \{\, x \,:\, x^\top (A^\top A) x - 2(b^\top A)x \leq \delta^2 - \|b\|_2^2 \,\} \subset \mathbb{R}^n$$

is an ellipsoid. However, whenever $m < n$, this ellipsoid is *degenerate* since $A^\top A$ then is rank-deficient. The algorithms for projection onto ellipsoids mentioned in the

previous subsection are only shown to converge in the nondegenerate case. Therefore, we will take a different approach in the following.

Note that the projection problem for a given point $z \in \mathbb{R}^n$ onto the constraint set X_δ of (P_1^δ) can be written as

$$\min_{x \in \mathbb{R}^n} \; f_p(x) + g_p(Ax), \tag{4.45}$$

where $f_p(x) \coloneqq \frac{1}{2}\|x - z\|_2^2$ and $g_p(x) \coloneqq \iota_{\{\, v \,:\, \|v-b\|_2 \leq \delta \,\}}(x)$. We obtain the following result.

Proposition 4.27. *The dual problem of* (4.45) *is*

$$\max_{y \in \mathbb{R}^m} \; (Az - b)^\top y - \delta\|y\|_2 - \tfrac{1}{2}\|A^\top y\|_2^2. \tag{4.46}$$

Moreover, if y^* is an optimal solution of (4.46)*, then $x^* \coloneqq z - A^\top y^*$ is an optimal solution of* (4.45)*.*

Proof. By Fenchel-Rockafellar duality (Lemma 1.3), the dual of (4.45) is

$$\max_{y \in \mathbb{R}^m} \; -f_p^*(-A^\top y) - g_p^*(y),$$

where f_p^* and g_p^* are the conjugate functions of f_p and g_p, respectively. In the proof of Lemma 1.5 (see p. 15), we already saw that $g_p^*(y) = b^\top y + \delta\|y\|_2$. Moreover, it is easily verified (using the first-order optimality condition $\nabla f_p^*(v) = 0$) that $f_p^*(v) = z^\top v + \frac{1}{2}\|v\|_2^2$. Hence, the dual problem of (4.45) reads

$$\begin{aligned} & \max \; -(-z^\top A^\top y + \tfrac{1}{2}\|-A^\top y\|_2^2) - (b^\top y + \delta\|y\|_2) \\ = \; & \max \; (Az - b)^\top y - \delta\|y\|_2 - \tfrac{1}{2}\|A^\top y\|_2^2, \end{aligned}$$

as claimed. Moreover, strong duality holds, since the primal and dual problems are both always feasible and attain finite optima.

To show the second statement, let y^* be an optimal dual solution and assume w.l.o.g. that $z \notin X_\delta \coloneqq \{\, x \;:\; \|Ax - b\|_2 \leq \delta \,\}$ (otherwise, trivially, $y^* = 0$ and $x^* = z$). The saddle-point property from Lemma 1.3 yields

$$\begin{aligned} x^* \; &= \; \arg\min_{x \in \mathbb{R}^n} \; (Ax)^\top y^* + f_p(x) - g_p^*(y^*) \\ &= \; \arg\min_{x \in \mathbb{R}^n} \; x^\top A^\top y^* + \tfrac{1}{2}\|x - z\|_2^2 - b^\top y^* - \delta\|y^*\|_2, \end{aligned}$$

for which the first-order optimality condition (setting the objective gradient to zero) immediately gives the claimed identity $x^* = z - A^\top y^*$. This completes the proof. □

Remark 4.28. The result from Proposition 4.27 is not new; see, for instance, [111, Section 15.2] or the more general duality results [69, Propositions 3.3 and 3.4].

The strong duality statements of Proposition 4.27 allow us to solve the projection problem via its dual, which in turn can be cast as an unconstrained smooth convex minimization problem:

$$(4.46) \quad = \quad -\min_{y\in\mathbb{R}^m} -(Az-b)^\top y + \delta\|y\|_2 + \tfrac{1}{2}\|A^\top y\|_2^2. \tag{4.47}$$

Various iterative algorithms can be applied to solve this type of problem. For our purposes, i.e., to construct an adaptive approximate projection in the sense of (4.4), we are particularly interested in (nonasymptotic) *convergence rates* for such methods. More precisely, we wish to estimate the rate by which a sequence of primal approximations (x^k) of x^*, obtained as in Proposition 4.27 from some algorithm (for (4.47)), converges to x^*. To that end, we start with the following observations.

Lemma 4.29. *Let $F_d(y)$ denote the objective function of (4.47) and let (x^*, y^*) be a primal-dual optimal pair for (4.45) and (4.46) (or equivalently, (4.47)). Let $\hat{y} \in \mathbb{R}^m$ be arbitrary, and associate with it a point $\hat{x} := z - A^\top \hat{y} \in \mathbb{R}^n$. Finally, let $C \geq 0$ be an arbitrary real constant. Then the following holds:*

(i) *If $\|\hat{y} - y^*\|_2 \leq C$ then $\|\hat{x} - x^*\|_2 \leq C\|A\|_2$.*

(ii) *If $F_d(\hat{y}) - F_d(y^*) \leq C$ then $\|\hat{x} - x^*\|_2 \leq \sqrt{2C}$.*

Proof. The first statement is easy to see: Suppose $\|\hat{y} - y^*\|_2 \leq C$. Then, by construction and by Proposition 4.27,

$$\begin{aligned}\|\hat{x} - x^*\|_2 &= \|z - A^\top\hat{y} - (z - A^\top y^*)\|_2 \;=\; \|A^\top(\hat{y} - y^*)\|_2\\ &\leq \|\hat{y} - y^*\|_2\|A^\top\|_2 \;\leq\; C\|A\|_2.\end{aligned}$$

For the second statement, consider the first-order optimality condition for (4.46):

$$-(Az-b) + \delta\frac{y^*}{\|y^*\|_2} + AA^\top y^* = 0 \quad \Leftrightarrow \quad Az - b = \delta\frac{y^*}{\|y^*\|_2} + AA^\top y^*.$$

Thus, we obtain:

$$\begin{aligned}&F_d(\hat{y}) - F_d(y^*)\\ &= (Az-b)^\top(y^* - \hat{y}) - \delta(\|y^*\|_2 - \|\hat{y}\|_2) - \tfrac{1}{2}(\|A^\top y^*\|_2^2 - \|A^\top\hat{y}\|_2^2)\\ &= \delta\|y^*\|_2 - \delta\frac{(y^*)^\top(\hat{y})}{\|y^*\|_2} + \|A^\top y^*\|_2^2 - (y^*)^\top AA^\top(\hat{y}) - \delta\|y^*\|_2 + \delta\|\hat{y}\|_2\\ &\quad - \tfrac{1}{2}(\|A^\top y^*\|_2^2 - \|A^\top\hat{y}\|_2^2)\end{aligned}$$

$$
\begin{aligned}
&= \delta\|\hat{y}\|_2 - \delta\frac{(y^*)^\top(\hat{y})}{\|y^*\|_2} + \tfrac{1}{2}\|A^\top(y^* - \hat{y})\|_2^2 \\
&= \delta\|\hat{y}\|_2 - \delta\frac{(y^*)^\top(\hat{y})}{\|y^*\|_2} + \tfrac{1}{2}\|z - x^* - (z - \hat{x})\|_2^2 \\
&\geq \delta\|\hat{y}\|_2 - \delta\|\hat{y}\|_2 + \tfrac{1}{2}\|x^* - \hat{x}\|_2^2 \;=\; \tfrac{1}{2}\|\hat{x} - x^*\|_2^2.
\end{aligned}
$$

The inequality above follows from the Cauchy-Schwarz inequality, which implies that

$$
(\hat{y})^\top\Big(\frac{y^*}{\|y^*\|_2}\Big) \leq \|\hat{y}\|_2\left\|\frac{y^*}{\|y^*\|_2}\right\|_2 = \|\hat{y}\|_2.
$$

Hence, we see that if $F_d(\hat{y}) - F_d(y^*) \leq C$, then $\|\hat{x} - x^*\|_2 \leq \sqrt{2C}$. □

By the above lemma, we can exploit all (nonasymptotic) convergence rate results that either pertain to the convergence of the dual function values *or* the dual iterates (y^k) to derive rates for the associated sequence of primal iterates (x^k), taking $x^k := z - A^\top y^k$ for all k. Typically, convergence rates for function values do not imply rates for the iterates, and nonasymptotic rates directly involving the iterates are arguably less common. Hence, the flexibility provided by Lemma 4.29 with respect to the subject of convergence rates is clearly advantageous regarding the choice of suitable algorithms.

As mentioned earlier, and as bolstered by the previous lemma, we could employ various methods (with known convergence rates) to solve (4.46) (or equivalently, (4.47)), such as the standard gradient method (see, e.g., [24]) or the proximal point algorithm (see, e.g., [220]). For the sake of exposition, we shall apply the methods ISTA and FISTA from [18] to derive adaptive approximate projections for X_δ in the following.

Consider the following composite view on the problem (4.47):

$$
\min\, F_d(y) \quad = \quad \min\, f_d(y) + g_d(y),
$$

with $f_d(y) := -(Az - b)^\top y + \frac{1}{2}\|A^\top y\|_2^2$ and $g_d(y) := \delta\|y\|_2$. Note that the gradient of f_d, i.e., $\nabla f_d(y) = -(Az - b) + AA^\top y$, is Lipschitz-continuous with (smallest) Lipschitz constant $L_{f_d} := \|A\|_2^2 = \lambda_{\max}(AA^\top) = \sigma_{\max}(A)^2$: For all $y_1, y_2 \in \mathbb{R}^m$, it holds that

$$
\begin{aligned}
\|\nabla f_d(y_1) - \nabla f_d(y_2)\|_2 \;&= \|AA^\top(y_1 - y_2)\|_2 \\
&\leq \|AA^\top\|_2\|y_1 - y_2\|_2 \;=\; \|A\|_2^2\|y_1 - y_2\|_2.
\end{aligned}
$$

The *constant step size ISTA* [18], applied to (4.47), consists of the iterate update

rule (for $k \geq 0$)

$$\begin{aligned} y^{k+1} &:= y^k + \frac{1}{L}\big(A(z - A^\top y^k) - b\big) - \frac{1}{L}\mathcal{P}_{\mathcal{B}_2^\delta}(Ly^k + A(z - A^\top y^k) - b) \\ &= \frac{1}{L}\big[\mathrm{Id} - \mathcal{P}_{\mathcal{B}_2^\delta}\big](Ly^k + A(z - A^\top y^k) - b), \end{aligned} \tag{4.48}$$

where $y^0 \in \mathbb{R}^m$ can be chosen arbitrarily, $L \geq L_{f_d}$, Id denotes the identity operator, and $\mathcal{P}_{\mathcal{B}_2^\delta}$ is the projection onto the ℓ_2-norm ball of radius δ, i.e.,

$$\mathcal{P}_{\mathcal{B}_2^\delta}(q) = \left(\frac{1}{\max\{1, \|q\|_2/\delta\}}\right) q. \tag{4.49}$$

(For the sake of brevity, we omit the derivation of (4.48); it is straightforwardly obtained from applying the respective general formulas from [18] to (4.47), or as a special case of more general splitting methods like, e.g., [69, Algorithm 3.5].) The following convergence rate result derives from [18, Theorem 3.1], and holds for any dual optimal solution y^* (of which there is at least one [69]):

$$F_d(y^k) - F_d(y^*) \;\leq\; \frac{L_{f_d}}{2k}\|y^0 - y^*\|_2^2 \qquad \text{for any } k \geq 1. \tag{4.50}$$

In particular, this implies that the iteration number required to reach an ϵ-optimal solution $\hat{y}$, i.e., such that $F_d(\hat{y}) - F_d(y^*) \leq \epsilon$, is at most $\lceil((L_{f_d}/2)\|y^0 - y^*\|_2^2)/\epsilon\rceil$, see [18]. Combined with Lemma 4.29, we thus obtain the following result.

Proposition 4.30. *Let (y^k) be generated by (4.48) with a fixed $L \geq \|A\|_2^2$ and some $y^0 \in \mathbb{R}^m$. Let (x^k) be the associated primal iterate sequence given by $x^k := z - A^\top y^k$ for all k, and let $\varepsilon > 0$ be given. Then the mapping $\mathcal{P}^\varepsilon_{X_\delta, ISTA}$ defined by*

$$z \mapsto \mathcal{P}^\varepsilon_{X_\delta, ISTA}(z) := x^{k_\varepsilon},$$

where

$$k_\varepsilon := \left\lceil \frac{\|A\|_2^2\,\|y^0 - y^*\|_2^2}{\varepsilon^2} \right\rceil,$$

is an adaptive approximate projection in the sense of (4.4). Alternatively, k_ε can be replaced by the upper bound

$$\hat{k}_\varepsilon := \left\lceil \frac{L}{\varepsilon^2}\left(\|y^0\|_2 + \frac{\|Az - b\|_2 + \delta}{\sigma_{\min}(A)}\right)^2 \right\rceil.$$

Proof. To achieve a prescribed accuracy of ε in the sense of (4.4), i.e., such that $\|x^k - x^*\|_2 \leq \varepsilon$, we know from Lemma 4.29 and (4.50) that we must ensure $F_d(y^k) -$

$F_d(y^*) \leq \varepsilon^2/2$. Plugging this into the iteration number estimate following from [18, Theorem 3.1], we obtain k_ε as defined above. By substituting with suitable estimates, we can only worsen this bound (i.e., increase the implied iteration number which guarantees the desired projection accuracy). In particular, we can employ $\|A\|_2^2 = L_{f_d} \leq L$ as in the algorithm, and $\|y^0 - y^*\|_2 \leq \|y^0\|_2 + \|y^*\|_2$, where y^0 is known from the initialization and since $x^* = z - A^\top y^*$ implies $AA^\top y^* = A(z - x^*)$, and $AA^\top$ is invertible,

$$\begin{aligned}\|y^*\|_2 &= \|(AA^\top)^{-1}(Az - Ax^*)\|_2 = \|(AA^\top)^{-1}\big((Az - b) - (Ax^* - b)\big)\|_2 \\ &\leq \|(AA^\top)^{-1}\|_2\big(\|Az - b\|_2 + \|Ax^* - b\|_2\big) = \frac{\|Az - b\|_2 + \delta}{\sigma_{\min}(A)}.\end{aligned}$$

These estimates precisely yield the claimed upper bound $\hat{k}_\varepsilon$. □

Remark 4.31. Note that, strictly speaking, k_ε is undefined for $\varepsilon = 0$. However, since $\lim_{k_\varepsilon \to \infty} \mathcal{P}^{k_\varepsilon}_{X_\delta, ISTA}(z) = \lim_{\varepsilon \searrow 0} \mathcal{P}^{k_\varepsilon}_{X_\delta, ISTA}(z) = \mathcal{P}^0_{X_\delta}(z)$, practical convergence (to within numerical machine precision) is of course always achieved after finitely many iterations.

In [18], the convergence rate of ISTA was improved to $\mathcal{O}(1/k^2)$ with a slight modification similar to Nesterov's accelerated gradient scheme [198] (see also [199, 200]), resulting in the *constant step size FISTA* algorithm:

$$y^k := \frac{1}{L}\big[\mathrm{Id} - \mathcal{P}_{\mathcal{B}_2^\delta}\big](Lu^k + A(z - A^\top u^k) - b), \tag{4.51}$$

$$t_{k+1} := (1 + \sqrt{4t_k^2 + 1})/2, \tag{4.52}$$

$$u^{k+1} := y^k + \Big(\frac{t_k - 1}{t_{k+1}}\Big)(y^k - y^{k-1}), \tag{4.53}$$

for $k \geq 1$, with some $L \geq L_{f_d} = \|A\|_2^2$, $t_1 := 1$, $y_0 \in \mathbb{R}^m$ arbitrary, and $u_1 := y_0$. Using Lemma 4.29, we obtain the following result using FISTA.

Proposition 4.32. *Let (y^k) be generated by (4.51)–(4.53) with some $L \geq \|A\|_2^2$, an arbitrary $y^0 \in \mathbb{R}^m$, $t_1 := 1$ and $u^1 := y^0$. Let (x^k) be the associated primal iterate sequence, with $x^k := z - A^\top y^k$ for all k, and let $\varepsilon > 0$ be given. Then the mapping $\mathcal{P}^\varepsilon_{X_\delta, FISTA}$ defined by*

$$z \mapsto \mathcal{P}^\varepsilon_{X_\delta, FISTA}(z) := x^{k_\varepsilon},$$

where

$$k_\varepsilon := \left\lceil \frac{2\|A\|_2\,\|y^0 - y^*\|_2}{\varepsilon} - 1 \right\rceil,$$

is an adaptive approximate projection in the sense of (4.4). *Alternatively,* k_ε *can be replaced by the upper bound*

$$\hat{k}_\varepsilon := \left\lceil \frac{2\sqrt{L}}{\varepsilon} \left(\|y^0\|_2 + \frac{\|Az - b\|_2 + \delta}{\sigma_{\min}(A)} \right) - 1 \right\rceil .$$

Proof. For any dual optimal solution y^*, [18, Theorem 4.4] provides the convergence rate estimate

$$F_d(y^k) - F_d(y^*) \;\leq\; \frac{2L_{f_d}}{(k+1)^2} \|y^0 - y^*\|_2^2 \qquad \text{for any } k \geq 1, \tag{4.54}$$

which implies that FISTA needs at most $\left\lceil \sqrt{2L_{f_d}}\, \|y^0 - y^*\|_2 / \sqrt{\epsilon} - 1 \right\rceil$ iterations to reach a point $\hat{y}$ with $F_d(\hat{y}) - F_d(y^*) \leq \epsilon$. The remainder of the proof is completely analogous to that of Proposition 4.30. □

Remark 4.33. Naturally, Remark 4.31 applies to $\mathcal{P}^{\varepsilon}_{X_\delta, FISTA}$ as well. Moreover, the following points are noteworthy.

1. For both ISTA and FISTA, there are backtracking variants in which estimates of L_{f_d} are computed dynamically in each iteration, see [18]. The respective convergence rates (4.50) and (4.54) change only by a constant factor, whence the (primal) rates exploited in Propositions 4.30 and 4.32 to obtain our adaptive approximate projections can be straightforwardly extended to accommodate backtracking as well.
2. The ISTA algorithm from [18] can be seen as a simple special instance of the broad classes of (proximal) forward-backward splitting methods or monotone operator splitting algorithms, see, e.g., [71, 72, 250, 243, 104, 57]. Its convergence to an optimal solution y^* is in fact also guaranteed for variable step sizes γ_k replacing the constant $1/L$, for $\gamma_k \in (0, 2/\|A\|_2^2)$, in more general Hilbert space settings, and when augmented with various possible extensions such as, e.g., allowing summable errors in operator evaluations (see, e.g., the aforementioned references). Moreover, the application (4.48) to projection onto X_δ via the dual problem was briefly sketched in [104] (see Lemma 2 and Proposition 6 there), specializing and applying a method from [72]; it can also be derived directly from [57].

   ISTA can also be seen as a constant-step special case (alternatively, as the limiting case $\epsilon = 1/\|A\|_2^2$) of the method from [69, Proposition 4.2], in which the step sizes (γ_k) are taken from the interval $[\epsilon, 2/\|A\|_2^2 - \epsilon]$ with $\epsilon \in (0, \min\{1, 1/\|A\|_2^2\})$.
3. For the case that A is a *tight frame*, i.e., $AA^\top = c\,I$ for some $c \neq 0$, the

projection onto X_δ has a closed form solution, see, e.g., [104, Lemma 2(i)]:

$$\mathcal{P}_{X_\delta}(z) \;=\; z - \frac{1}{c} A^\top \big[\mathcal{P}_{\mathcal{B}_2^\delta} - \mathrm{Id}\big](Az - b).$$

4. Implementations of ISTA (4.48) and FISTA (4.51)–(4.53) can be found in the MATLAB package "UNLocBoX", available from `http://unlocbox.sourceforge.net/`, which contains several more related projection or proximity operators (see, e.g., [71] for numerous examples of this generalization of the notion of projections).

4.5.3.1 Generalization to Variants of Denoising Constraints

When the denoising constraint $\|Ax - b\|_2 \leq \delta$ is replaced by analogues using other ℓ_p-norms (with $p \geq 1$), the same algorithms discussed above can be employed with the modification of replacing $\mathcal{P}_{\mathcal{B}_2^\delta}$ by the projection onto the respective ℓ_p-balls. This fact was pointed out in [104] (in connection with a different but related splitting method) and can also be seen from [69, Section 4.1], where the general problem under consideration covers projection problems onto feasible sets w.r.t. constraints of the form $Ax - b \in C$ for some closed convex set C, and the proposed algorithmic scheme contains ISTA as a special case (cf. Remark 4.33 above); of course, corresponding FISTA variants work analogously.

In particular, we obtain the following results:

Proposition 4.34. *Let $p \geq 1$ and denote $\mathcal{B}_p^\delta := \{\, x \,:\, \|x\|_p \leq \delta \,\}$.*

(i) The dual problem of projecting a point $z \in \mathbb{R}^n$ onto the set $X_\delta^p := \{\, x \,:\, \|Ax - b\|_p \leq \delta \,\}$ reads

$$\max_{y \in \mathbb{R}^m} \; (Az - b)^\top y - \delta \|y\|_p^* - \tfrac{1}{2}\|A^\top y\|_2^2. \tag{4.55}$$

With $y_{(p)}^*$ denoting the optimal solution of (4.55), it holds that $\mathcal{P}_{X_\delta^p}(z) = z - A^\top y_{(p)}^*$.

(ii) Let $(y_{(p)}^k) := (y^k)$ be generated by ISTA (4.48) (with $L \geq \|A\|_2^2$ and $y^0 \in \mathbb{R}^m$), in which $\mathcal{P}_{\mathcal{B}_2^\delta}$ is replaced by $\mathcal{P}_{\mathcal{B}_p^\delta}$. Moreover, let $(x_{(p)}^k)$ be the associated primal iterate sequence, with $x_{(p)}^k := z - A^\top y_{(p)}^k$ for all k, and let $\varepsilon > 0$ be given. Then the mapping $\mathcal{P}^\varepsilon_{X_\delta^p, ISTA}$ defined by

$$z \mapsto \mathcal{P}^\varepsilon_{X_\delta^p, ISTA}(z) := x_{(p)}^{k_\varepsilon^{ISTA}}, \quad \text{where } k_\varepsilon^{ISTA} := \left\lceil \frac{\|A\|_2^2 \, \|y^0 - y_{(p)}^*\|_2^2}{\varepsilon^2} \right\rceil,$$

is an adaptive approximate projection for X_δ^p in the sense of (4.4). Alternatively, k_ε^{ISTA} can be replaced by the upper bound

$$\hat{k}_\varepsilon^{ISTA} := \left\lceil \frac{L}{\varepsilon^2} \left(\|y^0\|_2 + \frac{\|Az-b\|_2 + \max\left\{1, n^{\frac{1}{2}-\frac{1}{p}}\right\}\delta}{\sigma_{\min}(A)} \right)^2 \right\rceil .$$

(iii) Let $(y_{(p)}^k) := (y^k)$ be generated by FISTA (4.51)–(4.53) (with $L \geq \|A\|_2^2$, $y^0 \in \mathbb{R}^m$, $t_1 := 1$ and $u^1 := y^0$), in which $\mathcal{P}_{\mathcal{B}_2^\delta}$ is replaced by $\mathcal{P}_{\mathcal{B}_p^\delta}$. Moreover, let $(x_{(p)}^k)$ be the associated primal iterate sequence, with $x_{(p)}^k := z - A^\top y_{(p)}^k$ for all k, and let $\varepsilon > 0$ be given. Then the mapping $\mathcal{P}^\varepsilon_{X_\delta^p, FISTA}$ defined by

$$z \mapsto \mathcal{P}^\varepsilon_{X_\delta^p, FISTA}(z) := x_{(p)}^{k_\varepsilon^{FISTA}}, \quad \text{where } k_\varepsilon^{FISTA} := \left\lceil \frac{2\|A\|_2 \, \|y^0 - y_{(p)}^*\|_2}{\varepsilon} - 1 \right\rceil,$$

is an adaptive approximate projection for X_δ^p in the sense of (4.4). Alternatively, k_ε^{FISTA} can be replaced by the upper bound

$$\hat{k}_\varepsilon^{FISTA} := \left\lceil \frac{2\sqrt{L}}{\varepsilon} \left(\|y^0\|_2 + \frac{\|Az-b\|_2 + \max\left\{1, n^{\frac{1}{2}-\frac{1}{p}}\right\}\delta}{\sigma_{\min}(A)} \right) - 1 \right\rceil .$$

Proof. Proceeding as for $\iota_{\{\, v \,:\, \|v-b\|_2 \leq \delta \,\}}(x)$ in the proof of Lemma 1.5, we obtain the conjugate function of $g(x) := \iota_{\{\, v \,:\, \|v-b\|_p \leq \delta \,\}}(x)$ as

$$g^*(y) \;=\; \max_x \{\, y^\top x \;:\; \|v-b\|_p \leq \delta \,\} \;=\; b^\top y + \delta \, \|y\|_p^*. \tag{4.56}$$

The remainder of the proof for part (i) is completely analogous to that of Proposition 4.27.

Parts (ii) and (iii) immediately follow from the preceding discussion; in particular, Propositions 4.30, 4.32 and part (i) above. (The alternative iteration numbers are obtained by extending the estimate for $\|y^*\|_2$ from the proof of Proposition 4.30: For $p \leq 2$, $\|Ax^* - b\|_2 \leq \|Ax^* - b\|_p \leq \delta$, and for $p \geq 2$, $\|Ax^* - b\|_2 \leq n^{(1/2-1/p)}\|Ax^* - b\|_p \leq n^{(1/2-1/p)}\,\delta$, with the usual convention $1/\infty = 0$. Since $n \geq 1$, the factor $\max\{1, n^{(1/2-1/p)}\}$ correctly adapts to both cases.) □

In view of (4.56), recall, in particular, that $\|\cdot\|_1^* = \|\cdot\|_\infty$ and $\|\cdot\|_\infty^* = \|\cdot\|_1$.

For practical applications of (F)ISTA in the above context, note that the projection of a point z onto $\mathcal{B}_p^\delta \subset \mathbb{R}^m$ should be simple to compute, which is indeed the case for $p \in \{1, 2, \infty\}$: $\mathcal{P}_{\mathcal{B}_2^\delta}(z) = z/\max\{\|z\|_2/\delta, 1\}$ (see (4.49) earlier), $\mathcal{P}_{\mathcal{B}_\infty^\delta}(z)$

is given componentwise by $z_i / \max\{|z_i|/\delta, 1\}$, and the projection onto $\mathcal{B}_1^\delta$ is also realizable in linear time $\mathcal{O}(m)$, see [246, 96].

To conclude, the above results (together with those of Section 4.5.1) show that ISA is applicable to various prominent problems in CS Sparse Recovery. Further details of the ISA adaption to ℓ_1-minimization will be given in Section 4.6 below (see, in particular, Theorems 4.36 and 4.38).

4.5.4 Convex Expected Value Constraints

To conclude our set of examples for the adaptive approximate projection employed in the ISA framework, let us consider *expected value constraints* [215, 164] of the following form

$$g(x) \;:=\; \mathbf{E}[f(x;\omega)] \;=\; \int_{\mathbb{R}^q} f(x;\omega)\, p(\omega)\, d\omega \;\leq\; \eta, \tag{4.57}$$

where $\mathbf{E}$ denotes the expected value, $\omega \subseteq \mathbb{R}^q$ is a vector of random variables with density p, x are deterministic variables in $\mathbb{R}^n$, $f : \mathbb{R}^n \times \mathbb{R}^q \to \mathbb{R}$, and $\eta \in \mathbb{R}$. If f is convex in x for every ω, (4.57) is a convex constraint. Expected value constraints appear in stochastic programming as, for instance, the expectational form of chance constraints, see, e.g., [56, 28], or when modeling expected loss or Value-at-Risk via integrated chance constraints, see, e.g., [156, 146, 157].

In general, $g(x)$ cannot easily be computed exactly [37]; however, it can be approximated using Monte Carlo methods if samples of ω can be (cheaply) generated: Taking k independent samples $\omega^1, \ldots, \omega^k$ yields the approximation

$$\hat{g}_k(x) \;:=\; \frac{1}{k} \sum_{i=1}^{k} f(x;\omega^i)$$

of $g(x)$. Moreover, assume that we can compute a subgradient $G(x;\omega) \in \partial_x f(x;\omega)$ for each value of x and ω. Thus, we have

$$h \;:=\; \mathbf{E}[G(x;\omega)] \;=\; \int_{\mathbb{R}^q} G(x;\omega) p(\omega) d\omega \;\in\; \partial g(x).$$

We then use the approximation

$$\hat{h}_k \;:=\; \hat{h}_k(x) \;:=\; \frac{1}{k} \sum_{i=1}^{k} G(x;\omega^i),$$

which is a "noisy unbiased subgradient" of g at x, see [37] for details. Then $\mathbf{E}[\hat{h}_k] = h \in \partial g(x)$. Note that the unbiasedness of $\hat{g}_k(x)$ and $\hat{h}_k$ is in fact independent of k, i.e., even for $k = 1$ we have unbiased estimators $\hat{g}_1(x)$ and $\hat{h}_1$, though of course they should be rather poor approximations of the true values. Large k, on the other hand, likely yield good approximations; in fact, $\lim_{k\to\infty} \hat{g}_k(x) = \mathbf{E}[\hat{g}_k(x)] = g(x)$ and $\lim_{k\to\infty} \hat{h}_k = \mathbf{E}[\hat{h}_k] = h$.

Consider the problem $\min\{ \|z - x\|_2^2/2 \,:\, g(x) \le \eta \}$ of projecting a point z onto the set $C \coloneqq \{ x \,:\, g(x) \le \eta \}$. The (necessary and sufficient) optimality conditions for the projection problem can easily be derived from, e.g, [221, Theorem 3.34], and read:

$$\begin{aligned} -z + x^* + \mu^* h^* &= 0 \quad \text{for some } h^* \in \partial g(x^*), \\ \mu^*(g(x^*) - \eta) &= 0, \\ g(x^*) \le \eta,\ \mu^* &\ge 0. \quad (\mu^* \in \mathbb{R}) \end{aligned}$$

Moreover, for $z \notin C$, it must in fact hold that the (Lagrange) multiplier $\mu^* > 0$ (because otherwise, $x^* = \mathcal{P}_C(z) = z$, which holds if and only if $z \in C$) and consequently, by the second condition above (complementary slackness), $g(x^*) = \eta$. Then, the idea is to replace $g(x^*)$ and h^* by the respective (Monte Carlo) estimates $\hat{g}_k(x)$ and $\hat{h}_k$. An adaptive approximate projection is obtained by solving (for μ)

$$\hat{x} \coloneqq z - \mu\, \hat{h}_k, \qquad \hat{g}_k(\hat{x}) = \eta. \tag{4.58}$$

For an appropriate sampling process, we can adaptively control the resulting projection error (with high probability).

4.5.4.1 Linear Special Case

We shall now demonstrate this approach on a simple example constraint for which the system (4.58) can be solved easily and explicit projection error bounds can be obtained. Consider the special case of a linear function $f : \mathbb{R}^n \to \mathbb{R}$ with random coefficients:

$$f(x;\omega) = \omega^\top x + \tau.$$

Without loss of generality, let us assume $\tau = 0$. This particular type of constraint is closely related to integrated chance constraints, which are used, for instance, to model bounds on expected losses of some kind; see, e.g., [156, 146]. For this choice of f, our Monte Carlo estimates are

$$\hat{h}_k \;=\; \hat{h}_k(x) \;=\; \frac{1}{k}\sum_{i=1}^{k} G(x;\omega^i) \;=\; \frac{1}{k}\sum_{i=1}^{k} \nabla_x f(x;\omega^i) \;=\; \frac{1}{k}\sum_{i=1}^{k} \omega^i$$

and

$$\hat{g}_k(x) \;=\; \frac{1}{k}\sum_{i=1}^{k} f(x;\omega^i) \;=\; \frac{1}{k}\big((\omega^1)^\top x + \cdots + (\omega^k)^\top x\big) \;=\; \Big(\frac{1}{k}\sum_{i=1}^{k}\omega^i\Big)^\top x \;=\; \hat{h}_k^\top x.$$

Moreover, assume that h, $\hat{h}_k \neq 0$; this corresponds to imposing a positive lower bound on the subgradient norm. (Note that $\mathbf{E}[\hat{h}_k] = \mathbf{E}[G(x;\omega)] = h$ is possibly unknown[7].)

Observing that $\hat{h}_k$ is in fact independent of x (so that, in particular, it holds that $\hat{h}_k(z) = \hat{h}_k$), (4.58) yields the optimal Lagrange multiplier $\mu^* = (\hat{h}_k^\top z - \eta)/\|\hat{h}_k\|_2^2$ and thus the solution

$$\mathcal{P}_C^k(z) \;\coloneqq\; y^* \;=\; z - \left(\frac{\hat{h}_k^\top z - \eta}{\|\hat{h}_k\|_2^2}\right)\hat{h}_k$$

to the approximated projection problem[8]. The exact projection is given by

$$\mathcal{P}_C(z) \;=\; \mathcal{P}_C^\infty(z) \;\coloneqq\; z - \frac{h^\top z - \eta}{\|h\|_2^2}h.$$

As the notation suggests, we do in fact have $\lim_{k\to\infty}\mathcal{P}_C^k(z) = \mathcal{P}_C^\infty(z)$ almost-surely, since $\mathrm{Prob}(\lim_{k\to\infty}\hat{h}_k = h) = 1$ by the (strong) law of large numbers.

Now, for sufficiently large k, we can give explicit confidence intervals for the expected value $h = \mathbf{E}[\hat{h}_k]$ of the vector $\hat{h}_k$ (whose distribution and variance may also be unknown) via the central limit theorem: Given some $\alpha \in (0,1)$, using the corrected sample variances $\frac{1}{k-1}\sum_{i=1}^k(\omega_j^i - (\hat{h}_k)_j)^2$ for each component j, and with the $(1-\frac{\alpha}{2})$-quantile $q_{(1-\alpha/2)}$ of the standard normal distribution $\mathcal{N}(0,1)$, we have

$$\mathrm{Prob}\left(h \in \left[\hat{h}_k \pm \frac{q_{(1-\alpha/2)}}{\sqrt{k}\sqrt{k-1}}\begin{pmatrix}\sqrt{\sum_{i=1}^k(\omega_1^i - (\hat{h}_k)_1)^2}\\ \vdots \\ \sqrt{\sum_{i=1}^k(\omega_n^i - (\hat{h}_k)_n)^2}\end{pmatrix}\right]\right) \;=\; 1-\alpha$$

(here, we abbreviate as $[\hat{h}_k \pm (\dots)]$ the interval $[\hat{h}_k - (\dots), \hat{h}_k + (\dots)]$), or equivalently,

[7] Note that the feasibility operator construction suggested in [201], although in principle very flexible, is in fact not applicable in this case.

[8] In fact, this approach produces a feasible point, although it needs not be the true projection of z onto the feasible set: Since $\hat{g}_k(x) = \hat{h}_k^\top x$, it holds that

$$g\Big(\mathcal{P}_C^k(z)\Big) = \mathbf{E}\Big[\hat{g}_k(\mathcal{P}_C^k(z))\Big] = \mathbf{E}\Big[\hat{h}_k^\top\Big(z - \Big(\tfrac{\hat{h}_k^\top z - \eta}{\hat{h}_k^\top \hat{h}_k}\Big)\hat{h}_k\Big)\Big] = \mathbf{E}[\hat{h}_k^\top z - \hat{h}_k^\top z + \eta] = \mathbf{E}[\eta] = \eta.$$

writing

$$v_k \;\coloneqq\; \frac{q_{(1-\alpha/2)}}{\sqrt{k}\,\sqrt{k-1}} \left(\sqrt{\sum_{i=1}^{k} (\omega_1^i - (\hat{h}_k)_1)^2}, \ldots, \sqrt{\sum_{i=1}^{k} (\omega_n^i - (\hat{h}_k)_n)^2} \right)^\top ,$$

$\mathrm{Prob}\big(h = \hat{h}_k + c\,v_k,\ c \in [-1,1]\big) = 1-\alpha$. Note that $\alpha \le 1$ implies $1/2 \le 1-\alpha/2$, and thus $q_{(1-\alpha/2)} \ge 0$; typical values for α are 0.1, 0.05 or 0.01.

Observe that, provided $h = \hat{h}_k + c\,v_k$, some elementary computations yield

$$\begin{aligned}
\|\mathcal{P}_C^k(z) - \mathcal{P}_C^\infty(z)\|_2^2 \;&=\; \left\| z - z + \frac{\hat{h}_k^\top z - \eta}{\|\hat{h}_k\|_2^2}\hat{h}_k - \frac{(\hat{h}_k + c\,v_k)^\top z - \eta}{\|\hat{h}_k + c\,v_k\|_2^2}(\hat{h}_k + c\,v_k) \right\|_2^2 \\
&= \frac{\left\| (\hat{h}_k^\top z - \eta)(\hat{h}_k + c\,v_k) - (\hat{h}_k^\top z - \eta + c\,v_k^\top z)\hat{h}_k \right\|_2^2}{\|\hat{h}_k\|_2^2\,\|\hat{h}_k + c\,v_k\|_2^2} \\
&= \frac{\left\| c\left((\hat{h}_k^\top z - \eta)v_k - (v_k^\top z)\hat{h}_k \right) \right\|_2^2}{\|\hat{h}_k\|_2^2\,\|\hat{h}_k + c\,v_k\|_2^2} \\
&= \underbrace{\left(\frac{\|(\hat{h}_k^\top z - \eta)v_k - (v_k^\top z)\hat{h}_k\|_2^2}{\|\hat{h}_k\|_2^2} \right)}_{=:D_k} \frac{c^2}{\|\hat{h}_k + c\,v_k\|_2^2}.
\end{aligned}$$

Clearly, for a given (fixed) k, D_k is a nonnegative constant. Therefore,

$$\mathrm{Prob}\left(\|\mathcal{P}_C^k(z) - \mathcal{P}_C^\infty(z)\|_2^2 \in \left[\min_{c\in[-1,1]} \varphi(c),\ \max_{c\in[-1,1]} \varphi(c) \right] \right) = 1-\alpha,$$

where

$$\varphi(c) \coloneqq D_k \frac{c^2}{\|\hat{h}_k + c v_k\|_2^2}.$$

We have

$$\varphi'(c) \;\coloneqq\; \frac{\partial}{\partial c}\varphi(c) \;=\; D_k \frac{2(\hat{h}_k^\top v_k)c^2 + 2\,\|\hat{h}_k\|_2^2\,c}{(\|v_k\|_2^2\,c^2 + 2(\hat{h}_k^\top v_k)c + \|\hat{h}_k\|_2^2)^2}$$

and

$$\varphi'(c) \;=\; 0 \qquad \text{for } c \in \left\{ 0, -\frac{\|\hat{h}_k\|_2^2}{\hat{h}_k^\top v_k} \right\}.$$

Moreover,

$$\varphi''(c) \coloneqq \frac{\partial^2}{\partial c^2}\varphi(c) = -2\,D_k\,\frac{2\,\|v_k\|_2^2\,(\hat{h}_k^\top v_k)\,c^3 + 3\,\|\hat{h}_k\|_2^2\,\|v_k\|_2^2\,c^2 - \|\hat{h}_k\|_2^4}{(\|v_k\|_2^2\,c^2 + 2(\hat{h}_k^\top v_k)c + \|\hat{h}_k\|_2^2)^3}$$

and thus in particular, $\varphi''(0) = 2D_k/\|\hat{h}_k\|_2^2$, and

$$\varphi''\left(-\frac{\|\hat{h}_k\|_2^2}{\hat{h}_k^\top v_k}\right) = \frac{-2\,D_k\,(\hat{h}_k^\top v_k)^4}{\|\hat{h}_k\|_2^2\,(\|\hat{h}_k\|_2^2\,\|v_k\|_2^2 - (\hat{h}_k^\top v_k)^2)^2}.$$

Since $\varphi''(0) \geq 0$, the (global) minimum of $\varphi(c)$ is attained at $\underline{c} \coloneqq 0 \in [-1,1]$, yielding $\varphi(\underline{c}) = 0$, the natural lower bound for any norm. On the other hand, the global maximum of $\varphi(c)$ is attained at $\tilde{c} \coloneqq -\|\hat{h}_k\|_2^2/(\hat{h}_k^\top v_k)$, since $\varphi''(\tilde{c}) \leq 0$. Hence, the maximum of $\varphi(c)$ over the interval $[-1,1]$ is attained at

$$\overline{c} \coloneqq \mathcal{P}_{[-1,1]}(\tilde{c}) = \begin{cases} -1, & \tilde{c} < -1, \\ 1, & \tilde{c} > 1, \\ \tilde{c}, & \tilde{c} \in [-1,1]. \end{cases}$$

Note that we had implicitly assumed $\hat{h}_k^\top v_k \neq 0$, whence $\tilde{c}$ is well-defined, and also that $v_k \to 0$ as $k \to \infty$, since $\hat{h}_k \to h$; consequently, for k sufficiently large, $\tilde{c} \notin [-1,1]$ and $\overline{c} = -\mathrm{sign}(\hat{h}_k^\top v_k)$.

Since the zero lower bound is trivial, we can omit it (as well as the squares) and obtain

$$\mathrm{Prob}\left(\ \|\mathcal{P}_C^k(z) - \mathcal{P}_C^\infty(z)\|_2 \leq \varepsilon_k\ \right) = 1-\alpha,$$

where

$$\varepsilon_k \coloneqq \left\| \frac{\hat{h}_k^\top z - \eta}{\|\hat{h}_k\|_2^2}\hat{h}_k - \frac{\hat{h}_k^\top z - \eta + \overline{c}\,v_k^\top z}{\|\hat{h}_k + \overline{c}\,v_k\|_2^2}(\hat{h}_k + \overline{c}\,v_k) \right\|_2$$

with $\overline{c} = -\mathrm{sign}(\hat{h}_k^\top v_k)$. Thus, for any given $\alpha \in (0,1)$, and for k sufficiently large, $\mathcal{P}_C^k(z)$ defines an adaptive approximate projection operator in the sense of (4.4), with probability $1-\alpha$.

Remark 4.35. It is noteworthy that the projection accuracy directly depends on k, and in the linear example above we could iteratively refine the estimate $\hat{h}_k$ easily by incorporating newly drawn independent samples.

4.6 Application in Compressed Sensing: ISAL1

Let us now describe a specialization of the ISA framework to the ℓ_1-minimization problem

$$\min \|x\|_1 \quad \text{s.t.} \quad Ax = b, \tag{P_1}$$

where $A \in \mathbb{R}^{m\times n}$ with $\operatorname{rank}(A) = m < n$ and $b \in \mathbb{R}^m$. Recall that if $b = 0$, the (unique) optimal solution is trivially the all-zero vector.

As seen before, the subdifferential of the ℓ_1-norm at a point x is given by

$$\partial\|x\|_1 = \left\{ h \in [-1,1]^n \,:\, h_i = \frac{x_i}{|x_i|} \text{ for all } i \in \{1,\dots,n\} \text{ with } x_i \neq 0 \right\}.$$

Thus, in particular, $0 \in \partial\|x\|_1$ if and only if $x = 0$. In the algorithm, we may simply use the signs of the iterates as subgradients, i.e.,

$$\partial\|x^k\|_1 \ni \operatorname{sign}(x^k) = \begin{cases} 1, & (x^k)_i > 0, \\ 0, & (x^k)_i = 0, \\ -1, & (x^k)_i < 0. \end{cases} \tag{4.59}$$

A lower bound φ on the optimal objective value f^* is easily available for problem (P_1); e.g., we could use the trivial zero bound, or any dual lower bound ($-b^\top y$ for any y with $\|A^\top y\|_\infty \le 1$), see also Section 4.6.1 below.

Moreover, we can use a truncated CG procedure as the adaptive approximate projection $\mathcal{P}_X^\varepsilon$ (here, $X = \{\, x \,:\, Ax = b \,\}$) for our algorithm; the details were given by Proposition 4.20 in Section 4.5.1.

We summarize the method in Algorithm 4.4, and obtain the following convergence result.

Theorem 4.36 (ISAL1 convergence)**.** *Let $X = \{\, x \,:\, Ax = b \,\}$ and $\mathcal{P}_X^\varepsilon$ as defined in Proposition 4.20, and let f^* and X^* be the optimal value and point set of problem (P_1), respectively. Let $\varphi \le f^*$, $0 < \lambda_k \le \beta < 2$ for all k, and $\sum_{k=0}^\infty \lambda_k = \infty$. For an arbitrary upper bound $\overline{\varphi} \ge f^*$, let*

$$\overline{L}_k := \frac{\lambda_k(2-\beta)(\|x^k\|_1 - \varphi)}{\|h^k\|_2^2}\Big(\overline{\varphi} - \|x^k\|_1 + \frac{\beta}{2-\beta}(\overline{\varphi} - \varphi)\Big),$$

and define, for any k,

$$\overline{d}_{X^*}(x^k) := 2\,\min\left\{\varepsilon_{k-1}, \frac{\|Ax^k - b\|_2}{\sigma_{\min}(A)}\right\} + \frac{\|x^k\|_1 - \varphi}{\sqrt{n}}.$$

Algorithm 4.4 ISAL1

Input: matrix $A \in \mathbb{R}^{m\times n}$, vector $b \in \mathbb{R}^m$, estimate $\varphi \in [0, f^*]$, starting point x^0, sequences (λ_k) and (ε_k), parameter $\gamma \in (0,1)$

Output: an (approximate) solution to $\min\{\, \|x\|_1 : x \in X \,\}$

if $b = 0$ **then**
  return optimal solution $x^* = 0$
initialize $k := 0$, $\ell = -1$, $x^{-1} := x^0$, $h^{-1} := 0$, $\alpha_{-1} := 0$, $\varepsilon_{-1} := \varepsilon_0$
repeat
  choose subgradient $h^k \in \partial\|x^k\|_1$ (e.g., $h^k = \operatorname{sign}(x^k)$)
  if $\|x^k\|_1 \le \varphi$ **or** $x^k = 0$ **then**
    if $x^k \in X$ **then**
      stop (with x^k optimal, $\|x^k\|_1 = \varphi = f^*$)
    increment $\ell := \ell + 1$
    reset $x^k := \mathcal{P}_X^{\varepsilon}(x^{k-1} - \alpha_{k-1}h^{k-1})$ for $\varepsilon = \gamma^\ell \varepsilon_{k-1}$
    go to Step 5
  compute step size $\alpha_k := \lambda_k(\|x^k\|_1 - \varphi)/\|h^k\|_2^2$
  compute next iterate $x^{k+1} := \mathcal{P}_X^{\varepsilon_k}(x^k - \alpha_k h^k)$
  reset $\ell := 0$ and increment $k := k + 1$
until a stopping criterion is satisfied

Furthermore, let

$$\bar{\varepsilon}_k := -\left(\frac{\lambda_k(\|x^k\|_1 - \varphi)}{\|h^k\|_2} + \overline{d}_{X^*}(x^k)\right) + \sqrt{\left(\frac{\lambda_k(\|x^k\|_1 - \varphi)}{\|h^k\|_2} + \overline{d}_{X^*}(x^k)\right)^2 - \overline{L}_k}. \tag{4.60}$$

Finally, let (ν_k) be a nonnegative sequence with $\sum_{k=0}^{\infty} \nu_k < \infty$.

If (ε_k) satisfies $0 \le \varepsilon_k \le \min\{|\bar{\varepsilon}_k|, \nu_k\}$ for all k, then the following holds.

(i) *For any given $\delta > 0$, there exists some index K such that*

$$\|x^K\|_1 \le \overline{\varphi} + \tfrac{\beta}{2-\beta}(\overline{\varphi} - \varphi) + \delta.$$

(ii) *If additionally $\lambda_k \to 0$, then all accumulation points of the sequence of objective function values $(\|x^k\|_1)$ of the ISAL1 iterates (x^k) are contained in the closed interval $[f^*, \overline{\varphi}]$. In particular, if $\overline{\varphi} = f^*$ then the whole sequence $(\|x^k\|_1)$ converges to the optimal value f^*, whereas on the other hand, if additionally $\|x^k\|_1 > \overline{\varphi}$ for all k, then $\|x^k\|_1 \to \overline{\varphi}$ as $k \to \infty$.*

Proof. We start by considering the special case $\overline{\varphi} = f^*$. Recall that Theorem 4.7 (applied to (P_1)) yields statements (i) and (ii) for this case, under the conditions that X^* is bounded, the subgradients h^k satisfy $0 < \underline{H} \le \|h^k\|_2 \le \overline{H} < \infty$, and (ε_k)

obeys $0 \le \varepsilon_k \le \min\{|\tilde{\varepsilon}_k|, \nu_k\}$ for all k, where

$$\tilde{\varepsilon}_k := -\left(\frac{\lambda_k(\|x^k\|_1 - \varphi)}{\|h^k\|_2} + d_{X^*}(x^k)\right) + \sqrt{\left(\frac{\lambda_k(\|x^k\|_1 - \varphi)}{\|h^k\|_2} + d_{X^*}(x^k)\right)^2 - L_k} \tag{4.61}$$

with

$$L_k := \frac{\lambda_k(2-\beta)(\|x^k\|_1 - \varphi)}{\|h^k\|_2^2}\Big(f^* - \|x^k\|_1 + \frac{\beta}{2-\beta}(f^* - \varphi)\Big).$$

The conditions on the subgradients are clearly satisfied in ISAL1: Since $x^k \neq 0$ for all k (possibly after sufficient projection accuracy refinement) and by (4.59), we have $1 \le \|h^k\|_2 \le \sqrt{n}$ for all k. Also, since for problem (P_1), $X^* \subseteq \{\, x \,:\, \|x\|_1 = f^* \,\}$, the boundedness of X^* is easy to see. Thus, it remains to show that the conditions on the projection accuracies are satisfied as well.

To that end, the main point is to replace the (exact) distance of x to X^* by a computable estimate based on the weak sharp minima property of $\|x\|_1$ over X. Indeed, note that the ℓ_1-norm can be written as a polyhedral function:

$$\|x\|_1 \;=\; \operatorname{sign}(x)^\top x \;=\; \max\{\, s^\top x \,:\, s_i \in \{\pm 1\} \text{ for all } 1 \le i \le n \,\}.$$

Then, with $\mu = \min\{\, \|s\|_2 \,:\, s_i \in \{\pm 1\} \text{ for all } i \,\} = \sqrt{n}$, it follows from results in [105] (see also Section 4.4.2) that

$$\|x\|_1 - f^* \;\ge\; \sqrt{n}\, d_{X^*}(x) \qquad \text{for all } x \in X = \{\, x \,:\, Ax = b \,\} \subset \mathbb{R}^n. \tag{4.62}$$

Moreover, note that $d_X(x^k) \le \varepsilon_{k-1}$ for all k, by construction of the adaptive approximate projection (we can assume without loss of generality that $\varepsilon_0 \ge d_X(x^0)$; otherwise, just consider iterations with $k \ge 1$). Thus, with $\varphi \le f^*$, we can employ the triangle inequality, (4.62), the norm relation $\|x\|_1 \le \sqrt{n}\,\|x\|_2$ and the inequality $\|x - \mathcal{P}_X(x)\|_2 \le \|Ax - b\|_2/\sigma_{\min}(A)$ (see the derivation of (4.37)) to obtain:

$$\begin{aligned}
d_{X^*}(x^k) \;&\le\; d_X(x^k) + d_{X^*}(\mathcal{P}_X(x^k)) \;\le\; d_X(x^k) + \frac{\|\mathcal{P}_X(x^k)\|_1 - f^*}{\sqrt{n}} \\
&\le\; \min\left\{\varepsilon_{k-1}, \frac{\|Ax^k - b\|_2}{\sigma_{\min}(A)}\right\} + \frac{\|\mathcal{P}_X(x^k) - x^k\|_1 + \|x^k\|_1 - \varphi}{\sqrt{n}} \\
&\le\; \min\left\{\varepsilon_{k-1}, \frac{\|Ax^k - b\|_2}{\sigma_{\min}(A)}\right\} + \frac{\sqrt{n}}{\sqrt{n}}\, d_X(x^k) + \frac{\|x^k\|_1 - \varphi}{\sqrt{n}} \\
&\le\; 2\,\min\left\{\varepsilon_{k-1}, \frac{\|Ax^k - b\|_2}{\sigma_{\min}(A)}\right\} + \frac{\|x^k\|_1 - \varphi}{\sqrt{n}} \;=\; \overline{d}_{X^*}(x^k).
\end{aligned}$$

Recall that the bound $\tilde{\varepsilon}_k$ given in (4.61) was chosen such that, for $\varepsilon_k \leq \tilde{\varepsilon}_k$,

$$\varepsilon_k^2 + 2\left(\frac{\lambda_k(\|x^k\|_1 - \varphi)}{\|h^k\|_2} + d_{X^*}(x^k)\right)\varepsilon_k + L_k \;\leq\; 0,$$

see Lemma 4.15. Replacing $d_{X^*}(x^k)$ by $\overline{d}_{X^*}(x^k)$ here yields the more restrictive upper bound

$$\tilde{\varepsilon}'_k := -\left(\frac{\lambda_k(\|x^k\|_1 - \varphi)}{\|h^k\|_2} + \overline{d}_{X^*}(x^k)\right) + \sqrt{\left(\frac{\lambda_k(\|x^k\|_1 - \varphi)}{\|h^k\|_2} + \overline{d}_{X^*}(x^k)\right)^2 - L_k}.$$

Thus, we can replace $\tilde{\varepsilon}_k$ by $\tilde{\varepsilon}'_k$ in Theorem 4.7 and maintain validity of the results when applied to (P_1). Moreover, since for $\overline{\varphi} = f^*$, $\overline{L}_k = L_k$ and thus $\bar{\varepsilon}_k = \tilde{\varepsilon}'_k$, the assertions of Theorem 4.7(ii) continue to hold true for ISAL1 in this special case, for which the proof is thus complete.

From now on, consider an arbitrary upper bound $\overline{\varphi} \geq f^*$. We wish to additionally replace the unknown value f^* in L_k. Observe that $\tilde{\varepsilon}'_k \geq 0$ only if $L_k \leq 0$. Thus, using the upper bound

$$L_k \;\leq\; \frac{\lambda_k(2-\beta)(\|x^k\|_1 - \varphi)}{\|h^k\|_2^2}\Big(\overline{\varphi} - \|x^k\|_1 + \frac{\beta}{2-\beta}(\overline{\varphi} - \varphi)\Big) \;=\; \overline{L}_k,$$

we can derive the computable bound $\bar{\varepsilon}_k$ given by (4.60); consequently, $\bar{\varepsilon}_k \geq 0$ if and only if $\overline{L}_k \leq 0$.

Completely analogous to the proof of Theorem 4.7(i), it can be shown that part (i) holds true (i.e., that for any $\delta > 0$ there exists an index K such that $f_K = \|x^K\|_1 \leq \overline{\varphi} + \frac{\beta}{2-\beta}(\overline{\varphi} - \varphi) + \delta$); we thus omit the details.

Regarding part (ii), the remaining special case $\|x^k\|_1 > \overline{\varphi}$ for all k can be handled similarly to the proof of Theorem 4.7(ii): Repeated application of part (i) yields a subsequence $(\|x^{K_j}\|_1)$ which converges to $\overline{\varphi}$ as $j \to \infty$. With the same techniques employed previously, one can then show that no other accumulation point (larger than $\overline{\varphi}$) can exist, whence the whole sequence of objective function values indeed converges to $\overline{\varphi}$.

Finally, assume that $\|x^k\|_1 \leq \overline{\varphi}$ does occur eventually. Let (f_k^-) and (f_k^+) denote the subsequences of objective function values at most f^* or at least $\overline{\varphi}$, respectively. By asymptotic feasibility, (f_k^-) clearly converges to f^*, if this subsequence is infinite. Moreover, the above discussion of the case that $\|x^k\|_1 > \overline{\varphi}$ holds for all k can naturally be applied to (f_k^+), which therefore converges to $\overline{\varphi}$, if this inequality holds infinitely often.

Obviously, all possibly remaining accumulation points of $(\|x^k\|_1)$ must lie within

$[f^*, \overline{\varphi}]$. (Indeed, if $f^* \leq \|x^k\|_1 \leq \overline{\varphi}$ occurs infinitely often, the associated subsequence is bounded, and hence at least one such accumulation point must exist.) □

We give the following remarks on Theorem 4.36.

Remark 4.37.

1. The main difference to the general convergence Theorem 4.7 is that Theorem 4.36 allows for replacing the (predetermined) input sequence (ε_k) by bounds that guarantee convergence and can be (automatically) computed in each iteration. Note also that Theorem 4.7 assumes $\varphi < f^*$. However, it is easy to see that the corresponding proof goes through for $\varphi = f^*$ as well (see also Theorem 4.5).
2. For the case $\overline{\varphi} = f^*$, convergence of the sequence of iterates (x^k) to an optimal point can be established by further assumptions on the relaxation parameters (λ_k), cf. Theorem 4.7(iii). The same holds (without additional assumptions) whenever the optimal solution is unique. This is particularly noteworthy in the applications of ℓ_1-minimization in Compressed Sensing, where solution uniqueness is a typical occurrence.
3. Clearly, more flexibility regarding the projection accuracy (i.e., larger bounds $\overline{\varepsilon}_k$) can be obtained via larger lower bounds φ and smaller upper bounds $\overline{\varphi}$, which improve $\overline{d}_{X^*}(x^k)$ and $\overline{L}_k$, respectively. For instance, one could make use of the Lipschitz-continuity of the ℓ_1-norm (with Lipschitz-constant 1), i.e.,

$$|\,\|x\|_1 - \|y\|_1\,| \;\leq\; \|x - y\|_2 \qquad \text{for all } x, y \in \mathbb{R}^n.$$

   Since the adaptive approximate projection ensures $d_X(x^k) \leq \varepsilon_{k-1}$, it holds that $f^* \leq \|x^k\|_1 + \varepsilon_{k-1}$ for all k; see also the variable target value schemes discussed in Section 4.4.3.
4. Finally, it should be mentioned that convergence of the objective function values to one specific value cannot be guaranteed as before, due to replacing L_k by $\overline{L}_k$: The condition for $\overline{L}_k$ to be nonpositive (so that nonnegative bounds $\overline{\varepsilon}_k$ can be employed) is sufficient to have $L_k \leq 0$, but clearly not necessary. However, one then cannot proceed as in the proof of Theorem 4.7 to obtain convergence to f^* for both subsequences below or above this value, unless $\overline{\varphi} = f^*$ (i.e., $\overline{L}_k = L_k$). For $\overline{\varphi} > f^*$, monotonicity of the distance to X^* is not controllable (using $\overline{\varepsilon}_k$) if $\|x^k\|_1 < \overline{\varphi} + \frac{\beta}{2-\beta}(\overline{\varphi} - \varphi)$, i.e., $\overline{L}_k > 0$. Then, the contradictions regarding accumulation points other than the desired value cannot be reproduced along the same lines as before (see the proofs of Theorems 4.5(ii) and 4.7(ii)), since the monotonic decrease of $(d_{X^*}(x^k))$ is not

known to hold for all k.

While we are not aware of an immediate remedy to this issue, several approaches suggest themselves to achieve convergence to the optimal value eventually. For instance, one could resort to exact projections as soon as $\overline{L}_k > 0$, although (given our original motivation) this is clearly undesirable. Variable target value approaches (see Section 4.4.3) could be combined with employing computable bounds as used above, in particular regarding upper bounds of f^* to improve $\overline{L}_k$. Similarly, a reset mechanism could be included in which ISAL1 is restarted with improved $\overline{L}_k$, perhaps several times over (ideally achieving $\overline{\varphi} \to f^*$).

In fact, ISAL1 can handle BP Denoising problems as well—the following result covers (P_1^δ) and variants with other ℓ_p-norms in the constraints:

Theorem 4.38 (ISAL1 convergence for ℓ_p-norm denoising constraints)**.** *Let $p \geq 1$ and let f^* and X_δ^* be the optimal value and point set of*

$$\min \|x\|_1 \quad \text{s.t.} \quad \|Ax - b\|_p \leq \delta, \tag{$\mathrm{P}_{1,p}^\delta$}$$

respectively, and suppose we apply ISAL1 to this problem, i.e., in Algorithm 4.4 we let $X = X_\delta^p := \{\, x : \|Ax - b\|_p \leq \delta \,\}$ and let $\mathcal{P}_X^\varepsilon$ be one of the adaptive approximate projection operators for X_δ^p given in Section 4.5.3. Then, under the same assumptions, the convergence results of Theorem 4.36 hold with respect to ($\mathrm{P}_{1,p}^\delta$) as well. Moreover, the condition $b = 0$ in Step 1 can be replaced by $\|b\|_p \leq \delta$.

Proof. Clearly, for ($\mathrm{P}_{1,p}^\delta$), if $\|b\|_p \leq \delta$, then $x^* = 0$, which justifies relaxing the condition $b = 0$ in Step 1. Moreover, the same statements regarding boundedness of the subgradients, achieving $x^k \neq 0$ for all k (possibly via projection accuracy refinements), and boundedness of the optimal set X_δ^* hold true for ISAL1 with $\mathcal{P}_X^\varepsilon$ replaced by $\mathcal{P}_{X_\delta^p,(F)ISTA}^\varepsilon$ as given in Propositions 4.30, 4.32 or 4.34, respectively. Thus, we can proceed completely analogously to the proof of Theorem 4.36.

In fact, the only point requiring reinspection is the bound $\overline{d}_{X^*}(x)$ (here, $X^* \equiv X_\delta^*$): Since for any $p \geq 1$ (and $\delta \geq 0$), $X_0 := \{\, x : Ax = b \,\} \subseteq X_\delta^p$, we have $d_{X_\delta^p}(x) \leq d_{X_0}(x)$. Therefore, with $X_\delta^* = \arg\min\{\, \|x\|_1 : \|Ax - b\|_p \leq \delta \,\}$ and for any k,

$$\begin{aligned} d_{X_\delta^*}(x^k) \;&\leq\; d_{X_\delta^p}(x^k) + d_{X_\delta^*}(\mathcal{P}_{X_\delta}(x^k)) \;\leq\; d_{X_0}(x^k) + \frac{\|\mathcal{P}_{X_\delta}(x^k)\|_1 - f^*}{\sqrt{n}} \\ &\leq\; d_{X_0}(x^k) + \frac{\|\mathcal{P}_{X_\delta}(x^k) - x^k\|_1}{\sqrt{n}} + \frac{\|x^k\|_1 - \varphi}{\sqrt{n}} \end{aligned}$$

$$\leq\ d_{X_0}(x^k) + d_{X_\delta}(x^k) + \frac{\|x^k\|_1 - \varphi}{\sqrt{n}}\ \leq\ 2\, d_{X_0}(x^k) + \frac{\|x^k\|_1 - \varphi}{\sqrt{n}}.$$

Thus, $\overline{d}_{X^*}(x)$ as given in Theorem 4.36 is a valid upper bound w.r.t. ($\mathrm{P}^{\delta}_{1,\,p}$) (or X^*_δ, respectively) as well. □

The practicality of the proposed adaptive approximate projection operators involving (F)ISTA depends on p; at least for $p \in \{1, 2, \infty\}$, a low iteration complexity of the iterative projection schemes can be achieved, see the comment at the end of Section 4.5.3.

4.6.1 Implementation Details

In the following, we give a detailed description of the ISAL1 implementation for (P_1) that was included in our extensive ℓ_1-solver comparison (see Chapter 3, and [174]).

We use several ideas to improve practical performance, regardless of the theoretical convergence conditions.

The theoretical convergence result in Theorem 4.36 (see also Theorem 4.7) hinges on dynamic bounds on the projection accuracies (ε_k) and thus on the number of CG steps needed to guarantee these accuracies, cf. Proposition 4.20. Surprisingly, this does not seem to be essential in practice. We observed that limiting the number of CG steps to a small constant suffices to achieve practical convergence. The actual number seems to depend on the density of A—the sparser the matrix, the more CG steps are apparently necessary; for dense matrices often two or three steps suffice. As a default, we use at most five CG iterations to approximate the projection in our implementation. Hence, the sequence (ε_k) or the associated computable (upper) bounds do not appear explicitly anywhere in our implementation. Moreover, the case $f_k \leq \varphi$ never occurred in any of our numerical experiments (even with $\varphi > 0$, where occurrence is theoretically be possible). Therefore, our implementation presently does not contain a projection accuracy refinement phase (Steps 6–11 of Algorithm 4.4).

A well-known practical property of subgradient methods is the so-called "zigzagging" of the iteration points, which is also a main cause for the slow local convergence often exhibited by such algorithms. To alleviate this effect, we apply the following stabilization scheme, suggested in [172]: Replace the subgradient h^k by the convex combination

$$\tilde{h}^k := 0.6h^k + 0.2h^{k-1} + 0.1(h^{k-2} + h^{k-3}).$$

Note that, as a consequence, $\tilde{h}^k \in \partial\|x^k\|_1$ does not generally hold. Other stabilization techniques have been proposed, e.g., in [46, 75, 170, 166].

Furthermore, as suggested in [129], instead of using a predetermined step size parameter sequence (λ_k) we reduce λ_k by a factor of $\frac{1}{2}$ after a number p of consecutive iterations without "relevant" improvement in the objective (i.e., a reduction of the previous best value by at least 10^{-6}), or after $\overline{p}$ iterations (independently of the improvement). Choosing improvement (as opposed to change) in the objective turned out to work well in practice. We start with an initial value of $\lambda_0 = 0.85$. Moreover, we performed a large number of experiments to estimate good rules for choosing p and $\overline{p}$; it turns out that they depend on the density of the $(m \times n)$ matrix A, which is given by

$$\rho(A) := \frac{\|A\|_0}{mn} = \frac{|\{\,\{i,j\} \,:\, a_{ij} \neq 0\,\}|}{mn}.$$

Specifically, we use

$$p := \begin{cases} \left\lceil \max\left\{5, \frac{n}{m\rho(A)^{3/4}}\right\}\right\rceil, & \text{if } \rho(A) \leq 0.1, \\ \left\lceil \max\left\{5, \frac{n}{m\rho(A)^{2}}\right\}\right\rceil, & \text{otherwise,} \end{cases}$$

and, with $s_p := \max\{500, 5p\}$,

$$\overline{p} := \max\left\{\left\lceil \frac{s_p}{\sqrt{2}} \right\rfloor, \left\lceil \frac{s_p}{(1+\rho(A))^2} \right\rfloor\right\}, \tag{4.63}$$

where $\lceil r \rfloor$ denotes the integer closest to a number r (rounded up or down, respectively).

Another well-known drawback of subgradient methods is the typical need for extensive parameter tuning to achieve practicality; ISA is no exception to this "rule". Indeed, despite having implemented various countermeasures and benchmarking the main algorithmic parameters, the above values can sometimes fail to yield convergence of our ISAL1 code. However, in such cases, this is often due to overly aggressive default parameter settings (which work well generally, but not always). Therefore, we apply a restart mechanism that resets some parameters to more conservative values[9] in case the step sizes become too small (close to numerical precision), or if the iterates fail to converge for the default parameter choices. The latter is assumed to be the case if a stagnation of the algorithmic process is detected for the first time. By stagnation, we mean that either the relevant objective improvement stalls over a

[9] Specifically, we reset $\lambda_k := \lambda_0$ and $p := 5p$, then recompute $s_p := \max\{10\,000, 10p\}$ and finally $\overline{p}$ via (4.63). (The current iterate x^k is the new "starting" point.)

span of s_p iterations, or the approximate support $S = \{\, i : |x_i^k| > \max\{10^{-6}, \epsilon_S\} \,\}$ does not change over 10 successive updates, which are performed every $\lfloor m/100 \rfloor$ iterations. Here, ϵ_S is chosen such that the entries x_j^k with $|x_j^k| \geq \epsilon_S$ account for at least 99.99% of $\|x^k\|_1$.

We terminate the method if either stagnation or too-small step sizes occur again after such a restart.

The dynamic step sizes use an estimate φ of the optimal value f^* of (P_1). This estimate is obtained as a dual lower bound. More precisely, given the starting point x^0 with (approximate) support S^0, we compute an approximate solution w^0 to $A_{S^0}^\top w = -\operatorname{sign}(x^0)$. The scaled vector $w^0/\|A^\top w^0\|_\infty$ is clearly a feasible solution to the dual problem of (P_1) (cf. Lemma 1.5),

$$\max -b^\top w \quad \text{s.t.} \quad -\mathbb{1} \leq A^\top w \leq \mathbb{1},$$

and thus implies the dual lower bound $-b^\top w^0/\|A^\top w^0\|_\infty =: \varphi \leq f^*$, which we use in our implementation. Note that other lower bounds are also easily available, e.g., 0 is always valid, and $b^\top b/\|A^\top b\|_\infty$ always yields a positive bound (since $b \neq 0$). However, we found the duality-based approach described above often gives a better bound and worked well in our experiments. (In fact, we automatically obtain such a dual bound from HOC—described in detail in Section 3.1—which serves as another stopping criterion in our ISAL1 implementation.)

Moreover, in the CG-based projection scheme given by Proposition 4.20, we usually compute products of the form $z = AA^\top q$ separately as $v = A^\top q$, $z = Av$, instead of directly computing $AA^\top$. For sparse A, this allows for faster evaluation of matrix-vector products (since $AA^\top$ may be dense), while for large dense matrices, computing $AA^\top$ explicitly can yield memory problems, so that in either case the separate computation is preferable. Nevertheless, for small dense matrices, $AA^\top$ is precomputed and used directly. (The computational results in [175, Section 5.2] show that, in this case, it still makes sense to approximate the projection instead of computing it exactly.)

Experiments showed that $x^0 := A^\top b$ generally seems to be a good starting point. If $AA^\top = I$, x^0 is feasible, since then $Ax^0 = AA^\top b = b$. This property of A allows for fast (exact) projections onto X and is (approximately) fulfilled by several types of matrices common in Compressed Sensing; see also the test set used for our ℓ_1-solver comparison, Section 3.3.

Two other aspects greatly improve the performance of ISAL1: Scaling the right hand side to unit Euclidean length, i.e., working with $b/\|b\|_2$ instead of b, helps in terms of numerical stability and convergence speed—for instance, such a scaling results in ISAL1 achieving more accurate solutions in less time for about 95% of the test instances with high dynamic range solutions (the geometric mean running time

reduced by about 81%, the average accuracy improved by 5 orders of magnitude). Intuitively, these improvements can be explained by the relative size of the components in a subgradient with respect to the scaled iterate vector: If $\|b\|_2$ is large, then a subgradient h at some point x is also a subgradient at $x/\|b\|_2$, and its entries are larger relative to the scaled point than to x itself, thus allowing for quicker progress toward the optimum. Similarly, if $\|b\|_2$ is very small, then scaling eventually results in reducing this relative size of a subgradient step (component-wise), thereby possibly avoiding unproductively large steps.

Finally, as a postprocessing step after termination, we try to improve the solution by solving (in a least-squares sense) the system $A_S x_S = b$ restricted to columns indexed by S, similarly to the "debiasing" step described in [256, Section II.I]. If the obtained point is feasible and yields a better objective function value than the best iterate, it is returned instead. (Here, S is the last support approximation obtained during the course of the algorithm.)

4.6.1.1 Recent Modifications and Additions to the ISAL1 Package

The above-described details pertain to ISAL1 version 0.91, which is the one used in the numerical experiments in the BP solver comparison in Chapter 3 (and in [174]). In the course of writing this thesis, we made a few minor, mostly cosmetic changes to the code. However, all of the above implementation details continue to hold.

One modification worth mentioning is the following: Recall that theoretically, ISAL1 will reach feasibility with respect to any fixed tolerance after finitely many iterations. As a practical safeguard, we implemented an additional high-accuracy projection (after regular termination) if the computed solution $\overline{x}$ does not already obey $\|A\overline{x} - b\|_\infty \leq 10^{-6}$.

More importantly, the ISAL1 package now also contains a prototypical implementation of the BP Denoising variant, i.e., ISA applied to (P_1^δ). Here, we realize the approximate projections onto $\{\, x \,:\, \|Ax - b\|_2 \leq \delta \,\}$ by means of the FISTA code from the UNLocBoX-Package previously mentioned in Section 4.5.3, Remark 4.33. (We used $\max\{\lambda_k, 10^{-5}\}$ as the feasibility tolerance of the projection algorithm, and start with a maximum allowed iteration number of 20 which is increased by 5 each time λ_k is halved until it reaches 100.) This ISAL1 variant also includes HOC for (P_1^δ), cf. Section 3.6.

However, as pointed out earlier, the many algorithmic parameters of the Denoising-ISAL1 code have not yet been tuned, and the experiments from Section 3.6.3 indicate that the current default values (from the original ISAL1 variant solving (P_1)) do not seem to be ideal choices w.r.t. (P_1^δ). Thus, the Denoising-ISAL1 implementation still has a somewhat "experimental", or prototypical, status.

The new version 1.0 of the ISAL1 package is available from the same webpage (http://wwwopt.mathematik.tu-darmstadt.de/spear) as version 0.91.

CHAPTER 5

Concluding Remarks

The main topics of this thesis were the computational complexity of sparse recovery conditions, a comparison of Basis Pursuit solvers as well as tools to improve their performance, and a general subgradient method for nonsmooth convex optimization (including a specialization to ℓ_1-minimization problems). Naturally, for each of these subject matters there remain open questions, further extensions or related problems to explore. In the following, we will give some pointers to possible future research along these lines.

5.1 Intractability of Recovery Conditions: Subtleties and Open Problems

Among other things, the results of Chapter 2 show that it is coNP-complete to answer the following questions in the case $\gamma = 1$:

Given a matrix A and order k, does the RIP or NSP hold with some constant $< \gamma$?

It is important to note that our results do *not* imply NP-hardness for *every* fixed constant $\gamma < 1$. For instance, Theorem 2.21 asserts that it is (co)NP-hard to certify the RIP for given A, k and $\delta_k \in (0, 1)$ *in general*; see also Corollary 2.22. The actual δ_k appearing in the proof, however, is very close to 1 and thus far from values of γ that yield actual sparse recovery guarantees. Similarly, while the NSP guarantees ℓ_0-ℓ_1-equivalence for $\underline{\alpha}_k < 1/2$, we proved NP-hardness for deciding whether $\underline{\alpha}_k < 1$. The complexity of these related questions therefore remains open.

(Note, however, that all such decision problems become solvable in time $\mathcal{O}(n^{\mathrm{poly}(k)})$ if k is *fixed*, regardless of the γ value.)

Nevertheless, our results do imply that *computing* the RIP and NSP constants $\underline{\delta}_k$ and $\underline{\alpha}_k$, respectively, is NP-hard in general. This provides a justification for investigating general approximation algorithms (that compute bounds on $\underline{\delta}_k$ or $\underline{\alpha}_k$) instead of searching for polynomial-time exact algorithms.

Indeed, to the best of our knowledge, the focus has so far been laid largely on relaxations or heuristics, see [77, 78, 145, 167] and other works. However, (co)NP-hardness does not necessarily exclude the possibility of *practically* efficient exact algorithms. In [158, 159], it was shown that one may sometimes do better than exhaustive search to certify the RIP, making use of the nondecreasing monotonicity of $\underline{\delta}_k$ with growing k. An exact procedure to compute the NSC $\underline{\alpha}_k$ was very recently proposed in [61] and empirically demonstrated to be faster than brute force. Unfortunately, neither method can *guarantee* a running time improvement with respect to simple enumeration. Moreover, the strong NP-hardness of computing the RIC $\underline{\delta}_k$ (cf. Corollary 2.32) shows that no general *pseudo*-polynomial-time algorithm (i.e., a method with running time polynomially bounded by the input size and the largest occurring numerical value) can exist for this task, unless P=NP [115]. On the other hand, the NSC $\underline{\alpha}_k$ is only shown to be weakly NP-hard to compute (based on the corresponding result about the spark of a matrix), so it might still be computable by (say) dynamic programming. More work on exact algorithms could certainly shed more light on the behavior of the spark, RIP and NSP.

Another interesting question is whether it is hard to *approximate* the constants associated with the RIP or NSP in polynomial time to within some factor. A first step in this direction was taken in [158, 159], where inapproximability of RIP parameters is shown under certain less common complexity assumptions (about dense subgraphs and the so-called "Hidden Clique problem", respectively), see also [22]. Moreover, our above-mentioned strong NP-hardness result implies that, given any $\epsilon > 0$, we cannot approximate $\underline{\delta}_k$ to within a factor of $1+\epsilon$ in time $\mathcal{O}(\mathrm{poly}(A, k, 1/\epsilon))$ (i.e., no FPTAS can exist) unless P=NP. Similarly, it would be good to know whether spark and NSC computations are actually strongly NP-hard as well, or if one of these problems admits an FPTAS (or, as mentioned above, a pseudo-polynomial-time algorithm), as weakly NP-hard problems often do. (Regarding the sparse reconstruction problem (P_0) itself, strong inapproximability results appear in [7].)

5.2 Test Sets, Solver Comparisons and HOC

With our test set and numerical experiments (see Chapter 3), we hope to have made a further step towards comprehensive, fair and meaningful comparisons of Basis Pursuit solvers. As is typical for basically any such computational tests, there are many aspects to consider, and there are often good arguments for both sides of opposing strategies. For instance, instead of the approach we chose—to assess the solvers in a black-box fashion and interpret the results under explicit consideration of different accuracy settings—one could have changed all codes to apply identical criteria, which would arguably yield the most directly comparable results. On the other hand, those findings could be misleading regarding the behavior of the original, unmodified solver implementation: Naturally, a convergent algorithm will eventually produce solutions that meet high accuracy demands, but oftentimes, convergence becomes considerably slower in the vicinity of the optimum. Thus, most solvers are tuned to balance the resulting trade-off between speed and accuracy, and may not be designed or intended for obtaining (numerically) exact solutions even if the underlying theory allows for it. This should arguably be taken into account—which eventually leads back to our black-box viewpoint.

Moreover, we saw that the heuristic optimality check we proposed for Basis Pursuit, i.e., HOC (Algorithm 3.2), often succeeds in identifying the exact optimum (within machine precision) and leads to early termination even before a solver's (medium-level) default target accuracy is reached. (Of course, the speed-ups—or overheads—would be even more pronounced if the solver was run with parameter settings aiming at high solution accuracy.) Importantly, HOC is independent of the inner workings of a given solver and can easily be integrated as an *additional* stopping criterion, and in case HOC is not successful, the induced runtime overhead is usually rather small.

With respect to both solver comparisons and further studies of the impact of HOC on solution speed and accuracy, many more numerical experiments can be made (regardless of which of the two above-described evaluation approaches is being pursued): Perhaps most importantly, our computational results would be well-complemented by a comparison on (very) large-scale instances, as these better reflect real-world problem dimensions. It would be interesting to see if the conclusions we reached from our tests continue to hold in such settings. In particular, the LP solvers Cplex and SoPlex (which turned out to be among the fastest and most reliable solution methods on our test set) will no longer be applicable once we deal with fast operators instead of explicit matrices. Note also that some solvers (e.g., YALL1 and NESTA) are designed or tuned for the case $AA^\top = I$, which should be taken into account for test set extensions and further comparisons.

Moreover, ℓ_1-Homotopy became arguably the best high-accuracy solver for (P_1) once we deviated from the theoretical requirements and actually applied it to the ℓ_1-regularized least-squares problem (QP_λ) with a very small regularization parameter λ. This naturally raises the question how the many more solvers that exist for (QP_λ) (e.g., SpaRSA [256], GPSR [107], or FISTA [18]) compare to the solvers for the pure BP problem (P_1) considered in Chapter 3 if run with similarly tiny λ values. (The idea to obtain high-accuracy approximately optimal solutions to (P_1) by solving (QP_λ) with small regularization parameter is, of course, not new, see for example [256].)

Remark 5.1. Note that most of the BP solvers considered in Chapter 3 are actively maintained; in particular, newer versions than the ones we used in our experiments are available for SPGL1 (now at version 1.8), YALL1 (1.4), ℓ_1-Homotopy (2.0), SoPlex (1.7.1) and CPLEX (12.5.1.0). Moreover, just recently, MATLAB version R2013b (8.2) has been released, and our own code ISAL1 is now at version 1.0 (though the changes to the BP code were minimal). However, presumably the "big picture" and conclusions will essentially remain the same, as most changes (compared to the versions we worked with) were minor bug-fixes or solver-package extensions to further problem variants not considered in this thesis.

The consideration of large-scale instances would clearly also be essential for a meaningful comparison of solvers for the *denoising* problems (P_1^δ) and (QP_λ): While the experiments in Section 3.6 indicate that the HOC variants adapted to these problems (i.e., Algorithms 3.3 and 3.4, respectively) are potentially beneficial, the solvers for which the influence was truly notable (ℓ_1-Magic and ISAL1 for (P_1^δ) and SolveBP/PDCO for (QP_λ)) were not competitive with the other solvers used in these tests (SPGL1 and ℓ_1-Homotopy). On the relatively small test instances from these experiments (the largest matrix was 1024×8192), the latter solvers were so fast that HOC hardly had a chance to improve anything. In particular, this can be understood literally for ℓ_1-Homotopy, which terminated after as many iterations as there are nonzeros in the optimum (i.e., with strictly increasing supports throughout the algorithmic process) on about 75% of the test instances considered there. Thus, to assess the potential of HOC more thoroughly, a rigorous solver comparison for (P_1^δ) or (QP_λ) along the lines of the one for (P_1) presented in Chapter 3 should focus on large-scale experiments. (Again, this also makes sense given that real-world instances usually are indeed large-scale.) In particular, Remark 3.5 is applicable for the denoising variants of HOC as well, i.e., the HOC schemes can be implemented to work with fast operators / implicit matrices (without the need for extracting columns explicitly) as are frequently encountered in the large-scale regime.

Moreover, in practical applications, precise values for δ or λ in (P_1^δ) and (QP_λ),

respectively, are usually not available. Therefore, it is worth investigating how the HOC variants behave in (or could be optimized for) situations with estimated parameter values. Given how HOC sometimes significantly improved the accuracy but not without introducing a slight overhead (see, for example, Figure 3.14), it may also be of interest to try HOC as a postprocessing routine and compare it with, e.g., the debiasing procedure from [256].

Concerning HOC for (P_1) (Algorithm 3.2), the construction of the dual certificate $\hat{w}$ may deserve further attention: The (least-squares) approach used in our HOC implementation is very specific and only guaranteed to work well if the BP solution is sufficiently sparse, see Section 3.1.3. Are there other sensible ways to quickly construct a dual solution to use within HOC for BP? One possible approach could be combining (P_1)-specifics with ideas from LP cross-over methods—note that essentially, HOC is a cross-over routine itself, although it attempts to construct a primal-dual optimal pair directly, and not via determining a nearby optimal basis (for an LP reformulation of (P_1)).

Remark 5.2. Within the research project SPEAR (which the author of this thesis participated in from 2011 to 2013), Christoph Brauer, presently a student at TU Braunschweig, has developed HOC for the ℓ_∞-constraint variant of Basis Pursuit Denoising, i.e.,

$$\min \|x\|_1 \quad \text{s.t.} \quad \|Ax - b\|_\infty \leq \delta.$$

Additionally, C. Brauer and Dirk Lorenz extended ISAL1 and HOC to the "ℓ_1-analysis" problem

$$\min \|Bx\|_1 \quad \text{s.t.} \quad Ax = b.$$

It will be interesting to see how these versions perform, or compare to other methods.

Finally, regarding test sets that admit objective comparisons, our approach was to construct instances with known (unique) optimal solutions. However, we reached the limits of the construction methods from L1TestPack (cf. [173]) for the larger matrices from our test set, and while our ERC-based procedure still worked, those instances are arguably easier in general (at least with respect to HOC). Thus, extending the applicability of (say) the alternating projections method from L1TestPack to higher dimensions remains a challenge. An interesting alternative to construct BP-recoverable solutions (of arbitrary support sizes up to the row number of the given matrix itself) was recently proposed in [162].

Moreover, with respect to unification and comparability of numerical solver evaluations for ℓ_1-minimization problems, one can envision setting up a large database of benchmark test problems comprising both synthetic and real-world data. Besides our own test set, several attempts in this direction have already been made over the

years, e.g., the Spot toolbox (see http://www.cs.ubc.ca/labs/scl/spot), which provides various matrix and fast operator constructions. So far, it seems that such collections are not very widely used. However, in view of the often-advocated "spirit of reproducible research" (cf. http://reproducibleresearch.net), a single large test problem collection arguably appears to be a logical next step.

5.3 Further Extensions and Applications of ISA

Several aspects regarding ISA remain subject to future research as well. For instance, one could investigate whether our framework can be extended to (infinite-dimensional) Hilbert space settings, or combined with incremental subgradient schemes, bundle methods (see, e.g., [134, 152]), or Nesterov's algorithm [199]. (In particular, one could explore the behavior of other algorithms that rely on projections if these methods are instead equipped with adaptive approximate projections as are used in our framework.) It is also of interest to consider how the ISA framework could be adapted to error-admitting settings such as those in [263, 194], i.e., incorporating random or deterministic (nonvanishing) noise and erroneous function or subgradient evaluations. Some of the fairly recent results in [194], which all require feasible iterates, seem conceptually close to our convergence analyses, so a blend of the two approaches might be promising.

Furthermore, it would also be interesting to investigate the convergence behavior of ISA-like subgradient schemes that employ other general, numerically motivated notions of "adaptive approximate projections", e.g., solving the projection problem with an approximation algorithm with additive or multiplicative performance guarantee.

From a practical viewpoint, the main question naturally is how the ISA variants compare with other solvers in terms of solution accuracy and runtime.

For the BP problem (P_1), the solver comparison in Section 3.4 demonstrated that ISAL1 is competitive on many test problems. Unfortunately, a typical drawback of basic subgradient schemes also pertains to ISAL1: Several algorithmic parameters needed to be benchmarked in extensive tests to achieve practicability of the method. As mentioned earlier, the parameter values that were found to work well for ISAL1 applied to (P_1) will most likely not be the best choices with respect to the denoising problem (P_1^δ). The experiments in Section 3.6 show that, in principle, ISAL1 works for (P_1^δ) as well, but the respective current implementation should be regarded as a prototype because the parameters have not yet been benchmarked specifically for this problem. Once a suitable test set for BP Denoising is available (cf. the

discussion in the previous subsection), such a parameter benchmarking is certainly a priority task with respect to improving the ISAL1 code for (P_1^δ).

Additionally, the practical application of ISAL1 to other denoising variants with constraints of the form $\|Ax - b\|_p \leq \delta$ for $2 \neq p \geq 1$—in particular, $p \in \{1, \infty\}$—is yet unexplored; the HOC procedure for ℓ_1-minimization under ℓ_∞-constraints (mentioned in Remark 5.2) may become valuable for such ISAL1 implementation extensions as well.

Beyond ℓ_1-minimization, specializations of the ISA framework to other problems and applications would certainly be of interest.

5.4 Sparse Recovery via Branch & Cut

In the introductory chapter, we mentioned the work [143] on solving general instances of the sparse representation problem

$$\min \|x\|_0 \quad \text{s.t.} \quad Ax = b \tag{P_0}$$

exactly, and preliminary further experiments by the author of this thesis with a different, but strongly related, approach. In order to address some challenges arising in this context, let us first informally outline the two approaches (without going into much detail).

In [143], a Branch & Cut method (cf., e.g., [224]) is applied to a binary integer programming (IP) formulation, exploiting the fact that (P_0) can be viewed (see [7]) as a special case of the *Maximum Feasible Subsystem* problem (MaxFS), which reads: Given an infeasible system $\Sigma := \{M\xi \leq f\}$, what is the largest possible number of satisfiable inequalities? Each binary variable in the resulting IP corresponds to one inequality of Σ and models whether it is satisfied or violated (given some ξ).

At first sight, a potential drawback of this approach is that the number of variables is doubled in the course of transforming the problem (see [143] for details). This can be avoided by a different IP reformulation of (P_0) in which the binary variables are in direct correspondence with the indices of entries of x. The resulting problem could be called *Minimum Support-Complement Cover* (MinSCC), since it is based on the observation that all supports of (feasible) solutions to $Ax = b$ need to contain at least one index from the respective complements of all inclusion-wise maximal infeasible supports. (The idea for this more direct approach, as well as a proof for the central observation just mentioned, were relayed to the author of this thesis by Marc Pfetsch, who coauthored [143] and developed the Branch & Cut code for the

MaxFS problem.)

Some experiments with a (prototypical) Branch & Cut code for MinSCC that we implemented showed, somewhat surprisingly, that the MaxFS-based approach seems superior despite the doubling of the variable number. Moreover, both algorithms could only solve very small instances of (P_0) within a reasonable time frame. Hence, there remain a lot of open questions and unresolved issues regarding these exact solution approaches; we sketch the most pressing aspects in the following.

Branch & Cut methods rely crucially on the ability to generate problem-specific cutting planes that cut off fractional solutions of intermediate LP relaxations of the integer program. Currently, for both of the above-mentioned IPs, only basic valid inequalities are known that come either directly from the initial formulations or from straightforward analysis of the zero-nonzero pattern of A and b. Thus, the most important challenge with respect to improving the algorithms is the identification of stronger cuts for the polytopes (integer hulls) underlying the respective problem formulations, and the development of corresponding separation routines that generate such cuts dynamically.

Regarding the relationship between the two IP models, a cursory analysis indicated that the doubling of variables may actually be beneficial here—essentially, the binary variables in the MaxFS-based variant actually correspond to indices in "split" supports that directly incorporate the signs of the associated x-variables in (P_0). Consequently, fixing a binary variable to (say) 1 in the IP from [143] amounts to fixing the sign of some entry in a solution x to $Ax = b$ to ± 1, and not only implies that its value must be nonzero (as is inferred by fixing a variable to 1 in the MinSCC-IP). A rigorous theoretical investigation of this aspect (e.g., from a polytope-theoretical viewpoint) should provide insights on whether one can indeed say that the MaxFS approach is somehow inherently better than the MinSCC formulation, as the existing computational results seem to indicate.

Several other implementational aspects (e.g., domain propagation) are also still largely unexplored and should be considered in the course of further development of these Branch & Cut algorithms.

5.5 Other Related Sparsity Problems

Exact methods for (P_0) could also be applied to compute the spark of a matrix, see Section 2.3.2. Instead of solving a sequence of n ℓ_0-minimization problems as suggested by the reduction sketched in that section, one could explore an alternative

randomized approach: Note that, since the spark is invariant w.r.t. scaling,

$$\operatorname{spark}(A) \;:=\; \min\{\,\|x\|_0 \,:\, Ax = 0,\; x \neq 0\,\} \;=\; \min\{\,\|x\|_0 \,:\, Ax = 0,\; \|x\|_2 = 1\,\}.$$

Clearly, if we knew some spark-defining vector x^*, we could replace the (quadratic) constraint $\|x\|_2 = 1$ by $(x^*)^\top x = 1$ and retain at least x^* as an optimal solution to the resulting problem, which therefore also yields the correct spark value. Since generally no such x^* is known, we could take a random vector ξ (with i.i.d. entries) and consider

$$\min\{\,\|x\|_0 \,:\, Ax = 0,\; \xi^\top x = 1\,\}.$$

Then, we may expect the solution value of this problem to equal $\operatorname{spark}(A)$, since with high probability, the vector ξ will be linearly independent of the rows of A (i.e., $\xi \notin \mathcal{R}(A^\top) = \mathcal{N}(A)^\perp$) and also will not be orthogonal to *every* vector $x \in \mathcal{N}(A)$ with minimal ℓ_0-norm.

Moreover, due to the optimality properties of greedy algorithms for matroid problems (see [205]), we can extend this idea to the problem of determining a sparse nullspace basis (cf. Section 2.3.2 and the related references therein): After solving the above problem, we could solve a sequence of problems, adding constraints in each step that ensure we always obtain a new solution vector that is linearly independent of the previously collected ones. If the true spark problem was correctly solved by the randomized approach, this procedure constructs a basis matrix for the nullspace with the fewest number of nonzeros possible.

As noted before, the Sparse Nullspace Basis problem is polynomially equivalent to Matrix Sparsification (finding a regular matrix T such that $B = TA$ is a sparsest-possible matrix with $\mathcal{R}(A) = \mathcal{R}(B)$ and $\mathcal{R}(A^\top) = \mathcal{R}(B^\top)$). Thus, similar approaches can be concocted for this problem, which essentially amount to sequentially finding sparse linear combinations of rows of A.

Naturally, instead of actually solving a series of (P_0) problems exactly, we can instead apply any of the heuristic methods known from Compressed Sensing, such as OMP or BP. It would be interesting to see how heuristics devised along these lines compare with the (surprisingly few) approaches proposed in the literature (e.g., [68, 117, 186, 55]). Moreover, there has apparently been little to no progress regarding matrix sparsification problems for a long time—the cited works are at least 20 years old—although clearly, many areas dealing with numerical linear algebra might profit greatly from new and improved developments. Arguably, the connections between these matrix problems and sparse vector recovery (see also [123]) may be particularly worth having a closer look at, given the advances the latter field has seen over the past decade.

Bibliography

[1] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[2] Y. I. Alber, A. N. Iusem, and M. V. Solodov, "On the Projected Subgradient Method for Nonsmooth Convex Optimization in a Hilbert Space," *Mathematical Programming*, vol. 81, no. 1, pp. 23–35, 1998.

[3] B. Alexeev, J. Cahill, and D. G. Mixon, "Full Spark Frames," *Journal of Fourier Analysis and Applications*, vol. 18, no. 6, pp. 1167–1194, 2012.

[4] E. Allen, R. Helgason, J. Kennington, and B. Shetty, "A Generalization of Polyak's Convergence Result for Subgradient Optimization," *Mathematical Programming*, vol. 37, no. 3, pp. 309–317, 1987.

[5] E. Amaldi and V. Kann, "The Complexity and Approximability of Finding Maximum Feasible Subsystems of Linear Relations," *Theoretical Computer Science*, vol. 147, no. 1–2, pp. 181–210, 1995.

[6] E. Amaldi and V. Kann, "On the Approximability of Some NP-hard Minimization Problems for Linear Systems," Cornell University, Ithaca, NY, USA, Tech. Rep. TR96-015, 1996.

[7] E. Amaldi and V. Kann, "On the Approximability of Minimizing Nonzero Variables or Unsatisfied Relations in Linear Systems," *Theoretical Computer Science*, vol. 209, no. 1–2, pp. 237–260, 1998.

[8] K. M. Anstreicher and L. A. Wolsey, "Two "Well-Known" Properties of Subgradient Optimization," *Mathematical Programming*, vol. 120, no. 1, pp. 213–220, 2009.

[9] M. S. Asif, "Primal Dual Pursuit—A Homotopy Based Algorithm for the Dantzig Selector," Master's thesis, Georgia Institute of Technology, Atlanta, GA, USA, 2008.

[10] M. S. Asif and J. Romberg, "On The LASSO and Dantzig Selector Equivalence," in *Proceedings of the 44th Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 2010.

[11] A. Auslaender and M. Teboulle, "Projected Subgradient Methods With Non-Euclidean Distances for Non-Differentiable Convex Minimization and Variational Inequalities," *Mathematical Programming*, vol. 120, no. 1, pp. 27–48, 2009.

[12] B. Bah and J. Tanner, "Improved Bounds on Restricted Isometry Constants for Gaussian Matrices," *SIAM Journal on Matrix Analysis and Applications*, vol. 31, no. 5, pp. 2882–2892, 2010.

[13] A. S. Bandeira, E. Dobriban, D. G. Mixon, and W. F. Sawin, "Certifying the Restricted Isometry Property is Hard," *IEEE Transaction on Information Theory*, vol. 59, no. 6, pp. 3448–3450, 2013.

[14] R. G. Baraniuk, M. A. Davenport, R. A. DeVore, and M. B. Wakin, "A Simple Proof of the Restricted Isometry Property for Random Matrices," *Constructive Approximation*, vol. 28, no. 3, pp. 253–263, 2008.

[15] R. H. Bartels, A. R. Conn, and J. W. Sinclair, "Minimization Techniques for Piecewise Differentiable Functions: The ℓ_1 Solution to an Overdetermined Linear System," *SIAM Journal of Numerical Analysis*, vol. 15, no. 2, pp. 224–241, 1978.

[16] H. H. Bauschke and J. M. Borwein, "On Projection Algorithms for Solving Convex Feasibility Problems," *SIAM Review*, vol. 38, no. 3, pp. 367–426, 1996.

[17] M. S. Bazaraa and H. D. Sherali, "On the Choice of Step Size in Subgradient Optimization," *European Journal of Operations Research*, vol. 7, no. 4, pp. 380–388, 1981.

[18] A. Beck and M. Teboulle, "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[19] S. Becker, J. Bobin, and E. J. Candès, "NESTA: A Fast and Accurate First-Order Method for Sparse Recovery," *SIAM Journal on Imaging Sciences*, vol. 4, no. 1, pp. 1–39, 2011.

[20] E. R. Berlekamp, R. J. McEliece, and H. C. A. van Tilborg, "On the Inherent Intractability of Certain Coding Problems," *IEEE Transactions on Information Theory*, vol. IT-24, no. 3, pp. 384–386, 1978.

[21] M. W. Berry, M. T. Heath, I. Kaneko, M. Lawo, R. J. Plemmons, and R. C. Ward, "An Algorithm to Compute a Sparse Basis of the Null Space," *Numerische Mathematik*, vol. 47, no. 4, pp. 483–504, 1985.

[22] Q. Berthet and P. Rigollet, "Computational Lower Bounds for Sparse PCA," arXiv:1304.0828 [math.ST], 2013.

[23] Q. Berthet and P. Rigollet, "Optimal Detection of Sparse Principal Components in High Dimensions," *The Annals of Statistics*, vol. 41, no. 4, pp. 1780–1815, 2013.

[24] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont, MA, USA: Athena Scientific, 1999.

[25] D. P. Bertsekas and S. K. Mitter, "A Descent Numerical Method for Optimization Problems With Nondifferentiable Cost Functionals," *SIAM Journal of Control*, vol. 11, no. 4, pp. 637–652, 1973.

[26] D. Bertsimas and X. Luo, "On the Worst Case Complexity of Potential Reduction Algorithms for Linear Programming," *Mathematical Programming*, vol. 77, no. 2, pp. 321–333, 1997.

[27] J. M. Bioucas-Dias and M. A. T. Figueiredo, "A New TwIST: Two-Step Iterative Shrinkage/Thresholding Algorithms for Image Restoration," *IEEE Transactions on Image Processing*, vol. 16, no. 12, pp. 2992–3004, 2007.

[28] J. R. Birge and F. Louveaux, *Introduction to Stochastic Programming*, ser. Springer Series in Operations Research and Financial Engineering. New York, NY, USA: Springer, 1999, corrected 2nd printing.

[29] E. G. Birgin, J. M. Martínez, and M. Raydan, "Nonmonotone Spectral Projected Gradient Methods on Convex Sets," *SIAM Journal on Optimization*, vol. 10, no. 4, pp. 1196–1211, 2000.

[30] R. E. Bixby and M. J. Saltzman, "Recovering an Optimal LP Basis from an Interior Point Solution," *Operations Research Letters*, vol. 15, no. 4, pp. 169–178, 1994.

[31] J. D. Blanchard, C. Cartis, and J. Tanner, "Compressed Sensing: How Sharp is the RIP?" *SIAM Review*, vol. 53, no. 1, pp. 105–125, 2011.

[32] T. Blumensath and M. E. Davies, "Iterative Hard Thresholding for Compressed Sensing," *Applied and Computational Harmonic Analysis*, vol. 27, no. 3, pp. 265–274, 2009.

[33] J. Bobin, J.-L. Starck, and R. Ottensamer, "Compressed Sensing in Astronomy," *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 5, pp. 718–726, 2008.

[34] R. I. Boţ, S.-M. Grad, and G. Wanka, "Generalized Moreau-Rockafellar Results for Composed Convex Functions," *Optimization*, vol. 58, no. 7, pp. 917–933, 2009.

[35] J. A. Bondy and U. S. R. Murty, *Graph Theory*, ser. Graduate Texts in Mathematics, vol. 44, corrected 2nd printing. New York, NY, USA: Springer, 2008.

[36] J. Bourgain, S. J. Dilworth, K. Ford, S. V. Konyagin, and D. Kutzarova, "Explicit Constructions of RIP Matrices and Related Problems," *Duke Mathematical Journal*, vol. 159, no. 1, pp. 145–185, 2011.

[37] S. Boyd and A. Mutapcic, "Stochastic Subgradient Methods," Lecture

notes, Stanford University, Stanford, CA, USA, 2008. Available online: http://see.stanford.edu/materials/lsocoee364b/04-stoch_subgrad_notes.pdf

[38] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[39] A. E. Brouwer and W. H. Haemers, *Spectra of Graphs*, ser. Universitext. New York, NY, USA: Springer, 2012.

[40] A. M. Bruckstein, D. L. Donoho, and M. Elad, "From Sparse Solutions of Systems of Equations to Sparse Modeling of Signals and Images," *SIAM Review*, vol. 51, no. 1, pp. 34–81, 2009.

[41] R. S. Burachik, V. Jeyakumar, and Z. Wu, "Necessary and Sufficient Conditions for Stable Conjugate Duality," *Nonlinear Analysis: Theory, Methods & Applications*, vol. 64, no. 9, pp. 1998–2006, 2006.

[42] J.-F. Cai, S. Osher, and Z. Shen, "Linearized Bregman Iterations for Compressed Sensing," *Mathematics of Computation*, vol. 78, no. 267, pp. 1515–1536, 2009.

[43] T. T. Cai and A. Zhang, "Compressed Sensing and Affine Rank Minimization Under Restricted Isometry," *IEEE Transactions on Signal Processing*, vol. 61, no. 13, pp. 3279–3290, 2013.

[44] T. T. Cai and A. Zhang, "Sharp RIP Bound for Sparse Signal and Low-Rank Matrix Recovery," *Applied and Computational Harmonic Analysis*, vol. 35, no. 1, pp. 74–93, 2013.

[45] T. T. Cai and A. Zhang, "Sparse Representation of a Polytope and Recovery of Sparse Signals and Low-Rank Matrices," *IEEE Transactions on Information Theory*, vol. 60, no. 1, pp. 122–132, 2014.

[46] P. M. Camerini, L. Fratta, and F. Maffioli, "On Improving Relaxation Methods by Modified Gradient Techniques," *Mathematical Programming Study*, vol. 3, pp. 26–34, 1975.

[47] E. J. Candès, "Compressive Sampling," in *Proceedings of the International Congress of Mathematicians (ICM)*, vol. III. Zürich, Switzerland: European Mathematical Society Publishing House, 2006, pp. 1433–1452.

[48] E. J. Candès, "The Restricted Isometry Property and Its Implications for Compressed Sensing," *Comptes Rendus Mathématique*, vol. 346, no. 9–10, pp. 589–592, 2008.

[49] E. J. Candès, J. Romberg, and T. Tao, "Robust Uncertainty Principles: Exact Signal Reconstruction from Highly Incomplete Frequency Information," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, 2006.

[50] E. J. Candès and T. Tao, "Decoding by Linear Programming," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[51] E. J. Candès and T. Tao, "The Dantzig Selector: Statistical Estimation When p Is Much Larger Than n," *The Annals of Statistics*, vol. 35, no. 6, pp. 2313–2351, 2007.

[52] E. J. Candès and M. B. Wakin, "An Introduction to Compressive Sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, 2008.

[53] I. Carron. (2013, September) Nuit Blanche. Available online: http://nuit-blanche.blogspot.co.at

[54] S. F. Chang and S. T. McCormick, "A Hierarchical Algorithm for Making Sparse Matrices Sparser," *Mathematical Programming*, vol. 56, no. 1–3, pp. 1–30, 1992.

[55] S. F. Chang and S. T. McCormick, "Implementation and Computational Results for the Hierarchical Algorithm for Making Sparse Matrices Sparser," *ACM Transactions on Mathematical Software*, vol. 19, no. 3, pp. 419–441, 1993.

[56] A. Charnes and W. W. Cooper, "Chance-Constrained Programming," *Management Science*, vol. 6, no. 1, pp. 73–79, 1959.

[57] G. H.-G. Chen and R. T. Rockafellar, "Convergence Rates in Forward-Backward Splitting," *SIAM Journal on Optimization*, vol. 7, no. 2, pp. 421–444, 1997.

[58] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic Decomposition by Basis Pursuit," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 33–61, 1998.

[59] A. Chistov, H. Fournier, L. Gurvits, and P. Koiran, "Vandermonde Matrices, NP-Completeness, and Transversal Subspaces," *Foundations of Computational Mathematics*, vol. 3, no. 4, pp. 421–427, 2003.

[60] J. J. Cho, Y. Chen, and Y. Ding, "On the (Co)Girth of a Connected Matroid," *Discrete Applied Mathematics*, vol. 155, no. 18, pp. 2456–2470, 2007.

[61] M. Cho and W. Xu, "Precisely Verifying the Null Space Conditions in Compressed Sensing: A Sandwiching Algorithm," arXiv:1306.2665 [cs.IT], 2013.

[62] V. Chvátal, *Linear Programming*. New York, NY, USA: W. H. Freeman and Company, 1983.

[63] A. Cohen, W. Dahmen, and R. A. DeVore, "Adaptive Wavelet Methods. II. Beyond the Elliptic Case," *Foundations of Computational Mathematics*, vol. 2, no. 3, pp. 203–245, 2002.

[64] A. Cohen, W. Dahmen, and R. A. DeVore, "Compressed Sensing and Best k-Term Approximation," *Journal of the American Mathematical Society*, vol. 22, no. 1, pp. 211–231, 2009.

[65] T. F. Coleman, "Sparse Null Bases and Marriage Theorems," Ph.D. dissertation, Cornell University, Ithaca, NY, USA, 1984.

[66] T. F. Coleman and A. Pothen, "The Sparse Null Space Basis Problem," Cornell University, Ithaca, NY, USA, Tech. Rep. TR 84-598, 1984.

[67] T. F. Coleman and A. Pothen, "The Null Space Problem I. Complexity," *SIAM Journal on Algebraic and Discrete Methods*, vol. 7, no. 4, pp. 527–537, 1986.

[68] T. F. Coleman and A. Pothen, "The Null Space Problem II. Algorithms," *SIAM Journal on Algebraic and Discrete Methods*, vol. 8, no. 4, pp. 544–563, 1987.

[69] P. L. Combettes, Đ. Dũng, and B. C. Vũ, "Dualization of Signal Recovery Problems," *Set-Valued and Variational Analysis*, vol. 18, no. 3–4, pp. 373–404, 2010.

[70] P. L. Combettes and J. Luo, "An Adaptive Level Set Method for Nondifferentiable Constrained Image Recovery," *IEEE Transactions on Image Processing*, vol. 11, no. 11, pp. 1295–1304, 2002.

[71] P. L. Combettes and J.-C. Pesquet, "Proximal Splitting Methods in Signal Processing," in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, ser. Optimization and Its Applications, vol. 49, H. H. Bauschke, R. S. Burachik, P. L. Combettes, V. Elser, D. R. Luke, and H. Wolkowicz, Eds. New York, NY, USA: Springer, 2011, ch. 10, pp. 185–212.

[72] P. L. Combettes and V. R. Wajs, "Signal Recovery by Proximal Forward-Backward Splitting," *Multiscale Modeling & Simulation*, vol. 4, no. 4, pp. 1168–1200, 2005.

[73] (2013, September) Compressive Sensing Resources. Available online: http://dsp.rice.edu/cs

[74] Y.-H. Dai, "Fast Algorithms for Projection on an Ellipsoid," *SIAM Journal on Optimization*, vol. 16, no. 4, pp. 986–1006, 2006.

[75] G. D'Antonio and A. Frangioni, "Convergence Analysis of Deflected Conditional Approximate Subgradient Methods," *SIAM Journal on Optimization*, vol. 20, no. 1, pp. 357–386, 2009.

[76] A. d'Aspremont, F. Bach, and L. E. Ghaoui, "Optimal Solutions for Sparse Principal Component Analysis," *Journal of Machine Learning Research*, vol. 9, no. Jul, pp. 1269–1294, 2008.

[77] A. d'Aspremont and L. E. Ghaoui, "Testing the Nullspace Property Using Semidefinite Programming," *Mathematical Programming*, vol. 127, no. 1, pp. 123–144, 2011.

[78] A. d'Aspremont, L. E. Ghaoui, M. I. Jordan, and G. R. G. Lanckriet, "A Direct Formulation for Sparse PCA Using Semidefinite Programming," *SIAM Review*, vol. 49, no. 3, pp. 434–448, 2007.

[79] I. Daubechies, M. Defrise, and C. De Mol, "An Iterative Thresholding Algorithm for Linear Inverse Problems with a Sparsity Constraint," *Commu-*

nications on Pure and Applied Mathematics, vol. 57, no. 11, pp. 1413–1457, 2004.

[80] M. A. Davenport and M. B. Wakin, "Analysis of Orthogonal Matching Pursuit Using the Restricted Isometry Property," *IEEE Transactions on Information Theory*, vol. 56, no. 9, pp. 4395–4401, 2010.

[81] M. E. Davies and R. Gribonval, "Restricted Isometry Constants where ℓ^p Sparse Recovery Can Fail for $0 < p \leq 1$," *IEEE Transactions on Information Theory*, vol. 55, no. 5, pp. 2203–2214, 2009.

[82] G. M. Davis, "Adaptive Nonlinear Approximations," Ph.D. dissertation, New York University, NY, USA, 1994.

[83] G. M. Davis, S. G. Mallat, and Z. Zhang, "Adaptive Time-Frequency Decompositions," *Optical Engineering*, vol. 33, no. 7, pp. 2183–2191, 1994.

[84] S. De Marchi, "Generalized Vandermonde Determinants, Toeplitz Matrices and Schur Functions," Universität Dortmund, Germany, Ergebnisberichte Angewandte Mathematik, vol. 176, 1999.

[85] R. A. DeVore, "Deterministic Constructions of Compressed Sensing Matrices," *Journal of Complexity*, vol. 23, no. 4–6, pp. 918–925, 2007.

[86] T. E. Dielman, "Least Absolute Value Regression: Recent Contributions," *Journal of Statistical Computation and Simulation*, vol. 75, no. 4, pp. 263–286, 2005.

[87] D. L. Donoho, "De-noising by Soft Thresholding," *IEEE Transactions on Information Theory*, vol. 41, no. 3, pp. 613–627, 1995.

[88] D. L. Donoho, "Neighborly Polytopes and Sparse Solutions to Underdetermined Linear Equations," Stanford University, Stanford, CA, USA, Tech. Rep. 2005-04, 2005.

[89] D. L. Donoho, "Compressed Sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[90] D. L. Donoho and M. Elad, "Optimally Sparse Representation in General (Non-Orthogonal) Dictionaries via ℓ^1 Minimization," *Proceedings of the National Academy of Sciences (USA)*, vol. 100, no. 5, pp. 2197–2202, 2003.

[91] D. L. Donoho, M. Elad, and V. N. Temlyakov, "Stable Recovery of Sparse Overcomplete Representations in the Presence of Noise," *IEEE Transactions on Information Theory*, vol. 52, no. 1, pp. 6–18, 2006.

[92] D. L. Donoho and X. Huo, "Uncertainty Principles and Ideal Atomic Decomposition," *IEEE Transactions on Information Theory*, vol. 47, no. 7, pp. 2845–2862, 2001.

[93] D. L. Donoho and Y. Tsaig, "Fast Solution of ℓ_1-Norm Minimization Problems when the Solution May Be Sparse," *IEEE Transactions on Information Theory*, vol. 54, no. 11, pp. 4789–4812, 2008.

[94] D. L. Donoho, Y. Tsaig, I. Drori, and J.-L. Starck, "Sparse Solution of Underdetermined Systems of Linear Equations by Stagewise Orthogonal Matching Pursuit," *IEEE Transactions on Information Theory*, vol. 58, no. 2, pp. 1094–1121, 2012.

[95] J. Duan, C. Soussen, D. Brie, J. Idier, and Y.-P. Wang, "A Sufficient Condition on Monotonic Increase of the Number of Nonzero Entry in the Optimizer of L1 Norm Penalized Least-Square Problem," arXiv:1104.3792 [stat.ML], 2011.

[96] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra, "Efficient Projections onto the ℓ_1-Ball for Learning in High Dimensions," in *Proceedings of the 25th International Conference on Machine Learning (ICML)*. New York, NY, USA: ACM, 2008, pp. 272–279.

[97] M. Dür, A. Martin, and S. Ulbrich, "Optimierung I – Einführung in die Optimierung," Lecture Notes, TU Darmstadt, Germany, 2009, in German. Available online: http://www.mathematik.tu-darmstadt.de/lehrmaterial/WS2008-2009/Opt_Einf/Skript/skript_Opt1.pdf

[98] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least Angle Regression," *The Annals of Statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[99] S. Egner and T. Minkwitz, "Sparsification of Rectangular Matrices," *Journal of Symbolic Computation*, vol. 26, no. 2, pp. 135–149, 1998.

[100] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Heidelberg, Germany: Springer, 2010.

[101] M. Elad and A. M. Bruckstein, "A Generalized Uncertainty Principle and Sparse Representations in Pairs of Bases," *IEEE Transactions on Information Theory*, vol. 48, no. 9, pp. 2558–2567, 2002.

[102] M. Elad, P. Milanfar, and R. Rubinstein, "Analysis versus Synthesis in Signal Priors," *Inverse Problems*, vol. 23, pp. 947–968, 2007.

[103] J. Erickson, "New Lower Bounds for Convex Hull Problems in Odd Dimensions," in *Proceedings of the 12th Annual ACM Symposium on Computational Geometry (SCG)*. New York, NY, USA: ACM, 1996, pp. 1–9.

[104] M. J. Fadili and J.-L. Starck, "Monotone Operator Splitting for Optimization Problems in Sparse Recovery," in *Proceedings of the 16th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2009, pp. 1461–1464.

[105] M. C. Ferris, "Weak Sharp Minima and Exact Penalty Functions," University of Wisconsin, Madison, WI, USA, Tech. Rep. 779, 1988.

[106] A. Feuer and A. Nemirovski, "On Sparse Representation in Pairs of Bases," *IEEE Transactions on Information Theory*, vol. 49, no. 6, pp. 1579–1581, 2003.

[107] M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright, "Gradient Projection

for Sparse Reconstruction: Applications to Compressed Sensing and Other Inverse Problems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 1, pp. 586–597, 2007.

[108] M. Fornasier, Ed., *Theoretical Foundations and Numerical Methods for Sparse Recovery*, ser. Radon Series on Computational and Applied Mathematics, vol. 9. Berlin, Germany: De Gruyter, 2010.

[109] M. Fornasier and H. Rauhut, "Compressive Sensing," in *Handbook of Mathematical Methods in Imaging*, vol. 1, O. Scherzer, Ed. Berlin, Heidelberg, New York: Springer, 2011, pp. 187–228.

[110] S. Foucart and M.-J. Lai, "Sparsest Solutions of Underdetermined Linear Systems via ℓ_q-Minimization for $0 \leq q \leq 1$," *Applied and Computational Harmonic Analysis*, vol. 26, no. 3, pp. 395–407, 2009.

[111] S. Foucart and H. Rauhut, *A Mathematical Introduction to Compressive Sensing*, ser. Applied and Numerical Harmonic Analysis. Basel, Switzerland: Birkhäuser, 2013.

[112] S. Friedland, "Bounds on the Spectral Radius of Graphs with e Edges," *Linear Algebra and its Applications*, vol. 101, pp. 81–86, 1988.

[113] M. P. Friedlander and P. Tseng, "Exact Regularization of Convex Programs," *SIAM Journal on Optimization*, vol. 18, no. 4, pp. 1326–1350, 2007.

[114] J.-J. Fuchs, "On Sparse Representations in Arbitrary Redundant Bases," *IEEE Transactions on Information Theory*, vol. 50, no. 6, pp. 1341–1344, 2004.

[115] M. R. Garey and D. S. Johnson, *Computers and Intractability. A Guide to the Theory of NP-completeness.* San Francisco, CA, USA: W. H. Freeman and Company, 1979.

[116] D. Ge, X. Jiang, and Y. Ye, "A Note on the Complexity of L_p Minimization," *Mathematical Programming*, vol. 129, no. 2, pp. 285–299, 2011.

[117] J. R. Gilbert and M. T. Heath, "Computing a Sparse Basis for the Null Space," *SIAM Journal on Algebraic and Discrete Methods*, vol. 8, no. 3, pp. 446–459, 1987.

[118] P. E. Gill, W. Murray, and M. H. Wright, *Numerical Linear Algebra and Optimization*, vol. 1. Redwood City, CA, USA: Addison-Wesley Publishing Company, 1991.

[119] GNU Scientific Library (GSL). Extension: Bundle. Available online: http://www.gnu.org/software/gsl

[120] J. L. Goffin and K. C. Kiwiel, "Convergence of a Simple Subgradient Level Method," *Mathematical Programming*, vol. 85, no. 1, pp. 207–211, 1999.

[121] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD, USA: Johns Hopkins University Press, 1996.

[122] C. C. Gonzaga, "An Algorithm for Solving Linear Programming Problems in $\mathcal{O}(n^3L)$ Operations," in *Progress in Mathematical Programming: Interior-point and related methods*, N. Megiddo, Ed. New York, NY, USA: Springer, 1988, pp. 1–28.

[123] L.-A. Gottlieb and T. Neylon, "Matrix Sparsification and the Sparse Null Space Problem," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (Proceedings APPROX 2010 and RANDOM 2010)*, ser. Lecture Notes in Computer Science, vol. 6302, M. Serna, R. Shaltiel, K. Jansen, and J. Rolim, Eds. Berlin, Heidelberg, Germany: Springer, 2010, pp. 205–218.

[124] S.-M. Grad, "New Insights into Conjugate Duality," Dissertation, Technische Universität Chemnitz, Germany, 2006.

[125] M. Grasmair, M. Haltmeier, and O. Scherzer, "Necessary and Sufficient Conditions for Linear Convergence of ℓ^1-Regularization," *Communications on Pure and Applied Mathematics*, vol. 64, no. 2, pp. 161–182, 2011.

[126] R. Gribonval and M. Nielsen, "Sparse Representations in Unions of Bases," *IEEE Transactions on Information Theory*, vol. 49, no. 12, pp. 3320–3325, 2003.

[127] R. Griesse and D. A. Lorenz, "A semismooth Newton method for Tikhonov functionals with sparsity constraints," *Inverse Problems*, vol. 24, no. 3, p. 035007, 2008.

[128] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, 2nd ed., ser. Algorithms and Combinatorics, vol. 2. Heidelberg, Germany: Springer, 1993.

[129] M. Held, P. Wolfe, and H. P. Crowder, "Validation of Subgradient Optimization," *Mathematical Programming*, vol. 6, pp. 62–88, 1974.

[130] C. Helmberg. Conic Bundle. Available online: http://www-user.tu-chemnitz.de/~helmberg/ConicBundle

[131] M. A. Herman and T. Strohmer, "High-Resolution Radar via Compressed Sensing," *IEEE Transactions on Signal Processing*, vol. 57, no. 6, pp. 2275–2284, 2009.

[132] F. J. Herrmann and G. Hennenfent, "Non-Parametric Seismic Data Recovery with Curvelet Frames," *Geophysical Journal International*, vol. 73, no. 1, pp. 233–248, 2008.

[133] M. R. Hestenes and E. Stiefel, "Methods of Conjugate Gradients for Solving Linear Systems," *Journal of Research of the National Bureau of Standards*, vol. 49, no. 6, pp. 409–436, 1952.

[134] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms. II (Advanced Theory and Bundle Methods)*, ser. Grundlehren

der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], vol. 306. Heidelberg, Germany: Springer, 1993.

[135] J.-B. Hiriart-Urruty and C. Lemaréchal, *Fundamentals of Convex Analysis*. Heidelberg, Germany: Springer, 2004, corrected 2nd printing.

[136] H. P. Hirst and W. T. Macey, "Bounding the Roots of Polynomials," *The College Mathematics Journal*, vol. 28, no. 4, pp. 292–295, 1997.

[137] A. J. Hoffman and S. T. McCormick, "A Fast Algorithm that Makes Matrices Optimally Sparse," in *Progress in Combinatorial Optimization*, W. R. Pulleyblank, Ed. Canada: Academic Press, 1984, pp. 185–196.

[138] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 2nd ed. New York, NY, USA: Cambridge University Press, 2013.

[139] E. Hübner, "A Bundle Method for Solving Convex Non-Smooth Minimization Problems," CiteSeerX preprint, DOI:10.1.1.92.2486, 2006.

[140] "IBM ILOG Cplex Optimization Studio 12.5," http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio.

[141] A. Itai and M. Rodeh, "Finding a Minimum Circuit in a Graph," *SIAM Journal on Computing*, vol. 7, no. 4, pp. 413–423, 1978.

[142] M. A. Iwen, "Simple Deterministically Constructible RIP Matrices with Sublinear Fourier Sampling Requirements," in *Proceedings of the 43rd Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 2009, pp. 870–875.

[143] S. Jokar and M. E. Pfetsch, "Exact and Approximate Sparse Solutions of Underdetermined Linear Equations," *SIAM Journal on Scientific Computing*, vol. 31, no. 1, pp. 23–44, 2008.

[144] M. Journée, Y. E. Nesterov, P. Richtárik, and R. Sepulchre, "Generalized Power Method for Sparse Principal Component Analysis," *Journal of Machine Learning Research*, vol. 11, pp. 517–533, 2010.

[145] A. Juditsky and A. Nemirovski, "On Verifiable Sufficient Conditions for Sparse Signal Recovery via ℓ_1 Minimization," *Mathematical Programming*, vol. 127, no. 1, pp. 57–88, 2011.

[146] P. Kall and J. Mayer, *Stochastic Linear Programming. Models, Theory, and Computation*, 2nd ed., ser. International Series in Operations Research & Management Science, vol. 156. New York, Dordrecht, Heidelberg, London: Springer, 2011.

[147] R. M. Karp, "Reducibility among Combinatorial Problems," in *Complexity of Computer Computations*, R. Miller and J. W. Thatcher, Eds. New York, NY, USA: Plenum Press, 1972, pp. 85–103.

[148] L. Khachiyan, "On the Complexity of Approximating Extremal Determinants in Matrices," *Journal of Complexity*, vol. 11, pp. 138–153, 1995.

[149] M. Khorramizadeh and N. Mahdavi-Amiri, "An Efficient Algorithm for Sparse Null Space Basis Problem Using *ABS* Methods," *Numerical Algorithms*, vol. 62, no. 3, pp. 469–485, 2013.

[150] S. Kim, H. Ahn, and S.-C. Cho, "Variable Target Value Subgradient Method," *Mathematical Programming*, vol. 49, no. 3, pp. 359–369, 1991.

[151] Y. N. Kiseliov, "Algorithms of Projection of a Point onto an Ellipsoid," *Lithuanian Mathematical Journal*, vol. 34, no. 2, pp. 141–159, 1994.

[152] K. C. Kiwiel, "Proximity Control in Bundle Methods for Convex Nondifferentiable Minimization," *Mathematical Programming*, vol. 46, no. 1, pp. 105–122, 1990.

[153] K. C. Kiwiel, "Subgradient Method with Entropic Projections for Convex Nondifferentiable Minimization," *Journal on Optimization Theory and Applications*, vol. 96, no. 1, pp. 159–173, 1998.

[154] K. C. Kiwiel, "Convergence of Approximate and Incremental Subgradient Methods for Convex Optimization," *SIAM Journal on Optimization*, vol. 14, no. 3, pp. 807–840, 2004.

[155] V. Klee and G. J. Minty, "How Good is the Simplex Algorithm?" in *Inequalities, III*, O. Shisha, Ed. New York, NY, USA: Academic Press, 1972, pp. 159–175.

[156] W. K. Klein Haneveld, *Duality in Stochastic Linear and Dynamic Programming*, ser. Lecture Notes in Economics and Mathematical Systems, vol. 274. New York, NY, USA: Springer, 1986.

[157] W. K. Klein Haneveld and M. H. van der Vlerk, "Integrated Chance Constraints: Reduced Forms and an Algorithm," *Computational Management Science*, vol. 3, no. 4, pp. 245–269, 2006.

[158] P. Koiran and A. Zouzias, "On the Certification of the Restricted Isometry Property," arXiv:1103.4984 [cs.CC], 2011.

[159] P. Koiran and A. Zouzias, "Hidden Cliques and the Certification of the Restricted Isometry Property," arXiv:1211.0665 [cs.CC], 2012.

[160] T. G. Kolda and B. W. Bader, "Tensor Decompositions and Applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.

[161] B. Korte and J. Vygen, *Combinatorial Optimization. Theory and Algorithms*, 5th ed., ser. Algorithms and Combinatorics, vol. 21. Berlin, Germany: Springer, 2011.

[162] C. Kruschel and D. A. Lorenz, "Computing and Analyzing Recoverable Supports for Sparse Reconstruction," arXiv:1309.2460 [math.OC], 2013.

[163] J. B. Kruskal, "Three-Way Arrays: Rank and Uniqueness of Trilinear Decompositions, with Application to Arithmetic Complexity and Statistics," *Linear Algebra and its Applications*, vol. 18, no. 2, pp. 95–138, 1977.

[164] D. Kuhn, "Convergent Bounds for Stochastic Programs with Expected Value Constraints," *Journal of Optimization Theory and Applications*, vol. 141, no. 3, pp. 597–618, 2009.

[165] G. Kutyniok and Y. C. Eldar, Eds., *Compressed Sensing: Theory and Applications*. New York, NY, USA: Cambridge University Press, 2012.

[166] T. Larsson, M. Patriksson, and A.-B. Strömberg, "Conditional Subgradient Optimization – Theory and Applications," *European Journal of Operations Research*, vol. 88, no. 2, pp. 382–403, 1996.

[167] K. Lee and Y. Bressler, "Computing Performance Guarantees for Compressed Sensing," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2008, pp. 5129–5132.

[168] A. S. Lewis, D. R. Luke, and J. Malick, "Local Linear Convergence for Alternating and Averaged Nonconvex Projections," *Foundations of Computational Mathematics*, vol. 9, no. 4, pp. 485–513, 2009.

[169] X. Li and S. Luo, "A Compressed Sensing-based Iterative Algorithm for CT Reconstruction and Its Possible Application to Phase Contrast Imaging," *BioMedical Engineering OnLine*, vol. 10, no. 1 (paper no. 73), 2011.

[170] C. Lim and H. D. Sherali, "Convergence and Computational Analyses for Some Variable Target Value and Subgradient Deflection Methods," *Computational Optimization and Applications*, vol. 34, no. 3, pp. 409–428, 2005.

[171] L.-H. Lim and P. Comon, "Multiarray Signal Processing: Tensor Decomposition Meets Compressed Sensing," *Comptes Rendus Mécanique*, vol. 338, no. 6, pp. 311–320, 2010.

[172] A. Löbel, "Optimal Vehicle Scheduling in Public Transit," Dissertation, Technische Universität Berlin, Germany, 1998.

[173] D. A. Lorenz, "Constructing Test Instances for Basis Pursuit Denoising," *IEEE Transactions on Signal Processing*, vol. 61, no. 5, pp. 1210–1214, 2013.

[174] D. A. Lorenz, M. E. Pfetsch, and A. M. Tillmann, "Solving Basis Pursuit: Heuristic Optimality Check and Solver Comparison," 2011. Available online: http://www.optimization-online.org/DB_HTML/2011/08/3132.html

[175] D. A. Lorenz, M. E. Pfetsch, and A. M. Tillmann, "An Infeasible-Point Subgradient Method Using Adaptive Approximate Projections," *Computational Optimization and Applications*, vol. 57, no. 2, pp. 271–306, 2014.

[176] D. A. Lorenz, S. Schiffler, and D. Trede, "Beyond Convergence Rates: Exact Inversion with Tikhonov Regularization with Sparsity Constraints," *Inverse Problems*, vol. 27, no. 8, p. 085009, 2011.

[177] R. Luss and M. Teboulle, "Conditional Gradient Algorithms for Rank-One Matrix Approximations with a Sparsity Constraint," *SIAM Review*, vol. 55, no. 1, pp. 65–98, 2013.

[178] M. Lustig, D. L. Donoho, and J. M. Pauly, "Sparse MRI: The Application of Compressed Sensing for Rapid MR Imaging," *Magnetic Resonance in Medicine*, vol. 58, no. 6, pp. 1182–1195, 2007.

[179] M. Lustig, D. L. Donoho, J. M. Santos, and J. M. Pauly, "Compressed Sensing MRI," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 72–82, 2008.

[180] J. Mairal and B. Yu, "Complexity Analysis of the Lasso Regularization Path," in *Proceedings of 29th International Conference on Machine Learning (ICML)*, J. Langford and J. Pineau, Eds. Madison, WI, USA: Omnipress, 2012, pp. 353–360.

[181] D. Malioutov, M. Çetin, and A. Willsky, "Homotopy Continuation for Sparse Signal Representation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 5. IEEE, 2005, pp. 733–736.

[182] S. G. Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way*, 3rd ed. USA: Academic Press, 2008.

[183] S. G. Mallat and Z. Zhang, "Matching Pursuits with Time-Frequency Dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[184] O. L. Mangasarian, "Minimum-Support Solutions of Polyhedral Concave Programs," *Optimization*, vol. 45, no. 1-4, pp. 149–162, 1999.

[185] S. T. McCormick, "A Combinatorial Approach to some Sparse Matrix Problems," Ph.D. dissertation, Stanford University, CA, USA, 1983.

[186] S. T. McCormick, "Making Sparse Matrices Sparser: Computational Results," *Mathematical Programming*, vol. 49, no. 1, pp. 91–111, 1990.

[187] N. Megiddo, "On Finding Primal- and Dual-Optimal Bases," *ORSA Journal on Computing*, vol. 3, no. 1, pp. 63–65, 1991.

[188] A. Milzarek and M. Ulbrich, "A Semismooth Newton Method with Multi-Dimensional Filter Globalization for ℓ_1-Optimization," 2013. Available online: http://www-m1.ma.tum.de/foswiki/pub/M1/Lehrstuhl/PublikationenUlbrich/MilzarekUlbrich-v2.pdf

[189] Q. Mo and Y. Shen, "A Remark on the Restricted Isometry Property in Orthogonal Matching Pursuit," *IEEE Transactions on Information Theory*, vol. 58, no. 6, pp. 3654–3656, 2012.

[190] T. S. Motzkin and I. J. Schoenberg, "The Relaxation Method for Linear Inequalities," *Canadian Journal of Mathematics*, vol. 6, pp. 393–404, 1954.

[191] S. Nam, M. E. Davies, M. Elad, and R. Gribonval, "The Cosparse Analysis Model and Algorithms," *Applied and Computational Harmonic Analysis*, vol. 34, no. 1, pp. 30–56, 2013.

[192] B. K. Natarajan, "Sparse Approximate Solutions to Linear Systems," *SIAM Journal on Computing*, vol. 24, no. 2, pp. 227–234, 1995.

[193] A. Nedić and D. P. Bertsekas, "Incremental Subgradient Methods for Nondifferentiable Optimization," *SIAM Journal on Optimization*, vol. 12, no. 1, pp. 109–138, 2001.

[194] A. Nedić and D. P. Bertsekas, "The Effect of Deterministic Noise in Subgradient Methods," *Mathematical Programming*, vol. 125, no. 1, pp. 75–99, 2010.

[195] D. Needell and J. A. Tropp, "CoSaMP: Iterative Signal Recovery from Incomplete and Inaccurate Samples," *Applied and Computational Harmonic Analysis*, vol. 26, no. 3, pp. 301–321, 2009.

[196] D. Needell and R. Vershynin, "Uniform Uncertainty Principle and Signal Recovery via Regularized Orthogonal Matching Pursuit," *Foundations of Computational Mathematics*, vol. 9, no. 3, pp. 317–334, 2009.

[197] A. S. Nemirovskiy and B. T. Polyak, "Iterative Methods for Solving Linear Ill-Posed Problems Under Precise Information. I," *Izvestiya Akademii Nauk SSSR. Tekhnicheskaya Kibernetika 5*, vol. 203, no. 2, pp. 13–2, 1984.

[198] Y. E. Nesterov, "A Method for Solving the Convex Programming Problem with Convergence Rate $O(1/k^2)$," *Soviet Mathematics Doklady*, vol. 27, no. 2, pp. 372–376, 1983.

[199] Y. E. Nesterov, "Smooth Minimization of Non-Smooth Functions," *Mathematical Programming*, vol. 103, no. 1, pp. 127–152, 2005.

[200] Y. E. Nesterov, "Gradient Methods for Minimizing Composite Objective Function," CORE discussion paper 2007/76, 2007. Available online: http://www.optimization-online.org/DB_HTML/2007/09/1784.html

[201] E. S. H. Neto and A. R. D. Pierro, "Incremental Subgradients for Constrained Convex Optimization: A Unified Framework and New Methods," *SIAM Journal on Optimization*, vol. 20, no. 3, pp. 1547–1572, 2009.

[202] T. Neylon, "Sparse Solutions for Linear Predictor Problems," Ph.D. dissertation, New York University, NY, USA, 2006.

[203] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed., ser. Springer Series in Operations Research and Financial Engineering. New York, NY, USA: Springer, 2006.

[204] M. Osbourne, B. Presnell, and B. Turlach, "A New Approach to Variable Selection in Least Squares Problems," *IMA Journal of Numerical Analysis*, vol. 20, no. 3, pp. 389–402, 2000.

[205] J. G. Oxley, *Matroid Theory*. New York, NY, USA: Oxford Graduate Texts in Mathematics, 1992.

[206] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Upper Saddle River, NJ, USA: Prentice-Hall, 1982.

[207] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal Matching Pursuit: Recursive Function Approximation with Applications to Wavelet Decomposition," in *Proceedings of the 27th Annual Asilomar Conference on Signals, Systems and Computers*, vol. 1. Pacific Grove, CA, USA: IEEE Computer Society Press, 1993, pp. 40–44.

[208] M. J. Piff and D. J. A. Welsh, "On the Vector Representation of Matroids," *Journal of the London Mathematical Society*, vol. 2, no. 2, pp. 284–288, 1970.

[209] A. Pinar, E. Chow, and A. Pothen, "Combinatorial Algorithms for Computing Column Space Bases That Have Sparse Inverses," *Electronic Transactions on Numerical Analysis*, vol. 22, pp. 122–145, 2006.

[210] M. D. Plumbley, T. Blumensath, L. Daudet, R. Gribonval, and M. E. Davies, "Sparse Representations in Audio and Music: From Coding to Source Separation," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 995–1005, 2010.

[211] B. T. Polyak, "A General Method for Solving Extremal Problems," *Doklady Akademii Nauk SSSR*, vol. 174, no. 1, pp. 33–36, 1967.

[212] B. T. Polyak, "Minimization of Nonsmooth Functionals," *USSR Computational Mathematics and Mathematical Physics*, vol. 9, no. 3, pp. 14–29, 1969.

[213] B. T. Polyak, "Subgradient Methods: A Survey of Soviet Research," in *Nonsmooth Optimization*, ser. IIASA Proceedings Series, vol. 3, C. Lemaréchal and R. Mifflin, Eds. Oxford, UK: Pergamon Press, 1978, pp. 5–29.

[214] L. C. Potter, E. Ertin, J. T. Parker, and M. Çetin, "Sparsity and Compressed Sensing in Radar Imaging," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1006–1020, 2010.

[215] A. Prékopa, "Contributions to the Theory of Stochastic Programming," *Mathematical Programming*, vol. 4, no. 1, pp. 202–221, 1973.

[216] J. F. Queiró, "On the Interlacing Property for Singular Values and Eigenvalues," *Linear Algebra and Its Applications*, vol. 97, pp. 23–28, 1987.

[217] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed Minimum-Rank Solutions of Linear Matrix Equations via Nuclear Norm Minimization," *SIAM Review*, vol. 52, no. 3, pp. 471–501, 2010.

[218] R. T. Rockafellar, "Duality and Stability in Extremum Problems Involving Convex Functions," *Pacific Journal of Mathematics*, vol. 21, no. 1, pp. 167–187, 1967.

[219] R. T. Rockafellar, *Conjugate Duality and Optimization*, ser. CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 16. Philadelphia, PA, USA: SIAM, 1974.

[220] R. T. Rockafellar, "Monotone Operators and the Proximal Point Algorithm," *SIAM Journal on Control and Optimization*, vol. 14, no. 5, pp. 877–898, 1976.

[221] A. Ruszczyński, *Nonlinear Optimization*. Princeton, NJ, USA: Princeton University Press, 2006.

[222] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. SIAM, 2003.

[223] S. Salzo and S. Villa, "Inexact and Accelerated Proximal Point Algorithms," *Journal of Convex Analysis*, vol. 19, no. 4, pp. 1167–1192, 2012.

[224] A. Schrijver, *Theory of Linear and Integer Programming*, ser. Wiley-Interscience Series in Discrete Mathematics and Optimization. New York, NY, USA: John Wiley & Sons, 1986.

[225] S. R. Searle, *Matrix Algebra Useful for Statistics*, ser. Wiley Series in Probability and Statistics. New York, Chichester, Brisbane: John Wiley & Sons, 1982.

[226] H. D. Sherali, G. Choi, and C. H. Tuncbilek, "A Variable Target Value Method for Nondifferentiable Optimization," *Operations Research Letters*, vol. 26, no. 1, pp. 1–8, 2000.

[227] J. R. Shewchuk, "An Introduction to the Conjugate Gradient Method Without the Agonizing Pain," Carnegie Mellon University, Pittsburgh, PA, USA, Tech. Rep. CMU-CS-TR-94-125, 1994.

[228] N. Z. Shor, *Minimization Methods for Non-Differentiable Functions*. New York, NY, USA: Springer, 1985.

[229] M. V. Solodov and S. K. Zavriev, "Error Stability Properties of Generalized Gradient-Type Algorithms," *Journal of Optimization Theory and Applications*, vol. 98, no. 3, pp. 663–680, 1998.

[230] (2013, September) Sparse- and Low-Rank Approximation Wiki. Available online: http://www.ugcs.caltech.edu/~srbecker/wiki

[231] J.-L. Starck, F. Murtagh, and J. M. Fadili, *Sparse Image and Signal Processing: Wavelets, Curvelets, Morphological Diversity*. New York, NY, USA: Cambridge University Press, 2010.

[232] T. Strohmer and R. W. H. Jr., "Grassmannian Frames with Applications to Coding and Communications," *Applied and Computational Harmonic Analysis*, vol. 14, no. 3, pp. 257–275, 2003.

[233] B. L. Sturm, B. Mailhé, and M. D. Plumbley, "On Theorem 10 in "On Polar Polytopes and the Recovery of Sparse Representations"," *IEEE Transactions on Information Theory*, vol. 59, no. 8, pp. 5206–5209, 2013.

[234] M. A. Sustik, J. A. Tropp, I. S. Dhillon, and R. W. H. Jr., "On the Existence of Equiangular Tight Frames," *Linear Algebra and its Applications*, vol. 426, pp. 619–635, 2007.

[235] R. Tibshirani, "Regression Shrinkage and Selection via the Lasso," *Journal of the Royal Statistical Society B*, vol. 58, no. 1, pp. 267–288, 1996.

[236] A. M. Tillmann, R. Gribonval, and M. E. Pfetsch, "Projection onto the k-Cosparse Set is NP-hard," arXiv:1303.5305 [cs.CC], 2013, accepted for Proc. ICASSP 2014.

[237] A. M. Tillmann and M. E. Pfetsch, "The Computational Complexity of the Restricted Isometry Property, the Nullspace Property, and Related Concepts in Compressed Sensing," *IEEE Transactions on Information Theory*, vol. 60, no. 2, pp. 1248–1259, 2014.

[238] I. Tošić and P. Frossard, "Dictionary Learning," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 27–38, 2011.

[239] J. A. Tropp, "Greed is Good: Algorithmic Results for Sparse Approximation," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2231–2242, 2004.

[240] J. A. Tropp, "Just Relax: Convex Programming Methods for Identifying Sparse Signals in Noise," *IEEE Transactions on Information Theory*, vol. 52, no. 3, pp. 1030–1051, 2006.

[241] J. A. Tropp and A. C. Gilbert, "Signal Recovery from Random Measurements via Orthogonal Matching Pursuit," *IEEE Transactions on Information Theory*, vol. 53, no. 12, pp. 4655–5666, 2007.

[242] J. A. Tropp and S. J. Wright, "Computational Methods for Sparse Solution of Linear Inverse Problems," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 948–958, 2010.

[243] P. Tseng, "Applications of a Splitting Algorithm to Decomposition in Convex Programming and Variational Inequalities," *SIAM Journal on Control and Optimization*, vol. 29, no. 1, pp. 119–138, 1991.

[244] P. M. Vaidya, "An Algorithm for Linear Programming which Requires $\mathcal{O}(((m+n)n^2 + (m+n)^{1.5}n)L)$ Arithmetic Operations," *Mathematical Programming*, vol. 47, no. 1–3, pp. 175–201, 1990.

[245] E. van den Berg and M. P. Friedlander, "Probing the Pareto Frontier for Basis Pursuit Solutions," *SIAM Journal on Scientific Computing*, vol. 31, no. 2, pp. 890–912, 2008.

[246] E. van den Berg, M. Schmidt, M. P. Friedlander, and K. Murphy, "Group Sparsity via Linear-Time Projection," University of British Columbia, Tech. Rep. TR-2008-09, 2008.

[247] J. H. van Lint, *Introduction to Coding Theory*, 3rd ed., ser. Graduate Texts in Mathematics, vol. 86. Berlin, Heidelberg, Germany: Springer, 1999.

[248] C. van Nuffelen, "On the Incidence Matrix of a Graph," *IEEE Transactions on Circuits and Systems*, vol. 23, no. 9, p. 572, 1976.

[249] A. Vardy, "The Intractability of Computing the Minimum Distance of a Code," *IEEE Transactions on Information Theory*, vol. 43, no. 6, pp. 1757–1766, 1997.

[250] S. Villa, S. Salzo, L. Baldassarre, and A. Verri, "Accelerated and Inexact Forward-Backward Algorithms," *SIAM Journal on Optimization*, vol. 23, no. 3, pp. 1607–1633, 2011.

[251] Y. Wang and W. Yin, "Sparse Signal Reconstruction via Iterative Support Detection," *SIAM Journal on Imaging Sciences*, vol. 3, no. 3, pp. 462–491, 2010.

[252] L. R. Welch, "Lower Bounds on the Maximum Cross Correlation of Signals," *IEEE Transactions on Information Theory*, vol. 20, no. 3, pp. 397–399, 1974.

[253] Z. Wen, W. Yin, D. Goldfarb, and Y. Zhang, "A Fast Algorithm for Sparse Reconstruction Based on Shrinkage, Subspace Optimization and Continuation," *SIAM Journal on Scientific Computing*, vol. 32, no. 4, pp. 1832–1857, 2010.

[254] S. Wenger, M. Ament, S. Guthe, D. A. Lorenz, A. M. Tillmann, D. Weiskopf, and M. Magnor, "Visualization of Astronomical Nebulae via Distributed Multi-GPU Compressed Sensing Tomography," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2188–2197, 2012.

[255] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust Face Recognition via Sparse Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210–227, 2009.

[256] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo, "Sparse Reconstruction by Separable Approximation," *IEEE Transactions on Signal Processing*, vol. 57, no. 7, pp. 2479–2493, 2009.

[257] R. Wunderling, "Paralleler und objektorientierter Simplex-Algorithmus," Dissertation, Technische Universität Berlin, Germany, 1996, in German.

[258] A. Y. Yang, A. G. Balasubramanian, Z. Zhou, S. S. Sastry, and Y. Ma, "Fast ℓ_1-Minimization Algorithms for Robust Face Recognition," *IEEE Transactions on Image Processing*, vol. 22, no. 8, pp. 3234–3246, 2013.

[259] J. Yang and Y. Zhang, "Alternating Direction Algorithms for ℓ_1-Problems in Compressive Sensing," Rice University, Houston, TX, USA, Tech. Rep. TR09-37, 2009.

[260] Y. Ye, "An $\mathcal{O}(n^3L)$ Potential Reduction Algorithm for Linear Programming," *Mathematical Programming*, vol. 50, no. 1–3, pp. 239–258, 1991.

[261] W. Yin, "Analysis and Generalizations of the Linearized Bregman Model," *SIAM Journal on Imaging Sciences*, vol. 3, no. 4, pp. 856–877, 2010.

[262] W. Yin, S. J. Osher, D. Goldfarb, and J. Darbon, "Bregman Iterative Algorithms for ℓ^1-Minimization with Applications to Compressed Sensing," *SIAM Journal on Imaging Sciences*, vol. 1, no. 1, pp. 143–168, 2008.

[263] A. J. Zaslavski, "The Projected Subgradient Method for Nonsmooth Convex Optimization in the Presence of Computational Error," *Numerical Functional Analysis and Optimization*, vol. 31, no. 5, pp. 616–633, 2010.

[264] H. Zhang, M. Yan, and W. Yin, "One Condition for All: Solution Uniqueness and Robustness of ℓ_1-Synthesis and ℓ_1-Analysis Minimizations," arXiv:1304.5038 [cs.IT], 2013.

[265] H. Zhang, W. Yin, and L. Cheng, "Necessary and Sufficient Conditions of Solution Uniqueness in ℓ_1 Minimization," 2012, arXiv:1209.0652 [cs:IT].

[266] Y. Zhang, "Theory of Compressive Sensing via ℓ_1-Minimization: A Non-RIP Analysis and Extensions," Department of Computational and Applied Mathematics, Rice University, Houston, TX, USA, Tech. Rep. TR08-11 (revised), 2008.

[267] Y.-B. Zhao, "RSP-Based Analysis for Sparsest and Least ℓ_1-Norm Solutions to Underdetermined Linear Systems," *IEEE Transactions on Signal Processing*, vol. 61, no. 22, pp. 5777–5788, 2013.

[268] L. Zhu, L. Lee, Y. Ma, Y. Ye, R. Mazzeo, and L. Xing, "Using Total-Variation Regularization for Intensity Modulated Radiation Therapy Inverse Planning with Field-Specific Numbers of Segments," *Physics in Medicine and Biology*, vol. 53, no. 23, pp. 6653–6672, 2008.

[269] Z. Zhu, A. Man-Cho So, and Y. Ye, "Fast and Near-Optimal Matrix Completion via Randomized Basis Pursuit," in *Proceedings of the Fifth International Congress of Chinese Mathematicians*, ser. AMS/IP Studies in Advanced Mathematics, vol. 51, L. Ji, Y. Sun Poon, L. Yang, and S.-T. Yau, Eds. Cambridge, MA, USA: AMS and International Press, 2012, pp. 859–882.

[270] H. Zou, T. Hastie, and R. Tibshirani, "Sparse Principal Component Analysis," *Journal of Computational and Graphical Statistics*, vol. 15, no. 2, pp. 265–286, 2006.

List of Figures

List of Tables

List of Algorithms