WHEN CAN YOU TRUST FEATURE SELECTION? – I: A CONDITION-BASED ANALYSIS OF LASSO AND GENERALISED HARDNESS OF APPROXIMATION

ALEXANDER BASTOUNIS, FELIPE CUCKER, AND ANDERS C. HANSEN

ABSTRACT. The arrival of AI techniques in computations, with the potential for hallucinations and non-robustness, has made trustworthiness of algorithms a focal point. However, trustworthiness of the many classical approaches are not well understood. This is the case for feature selection, a classical problem in the sciences, statistics, machine learning etc. Here, the LASSO optimisation problem is standard. Despite its widespread use, it has not been established when the output of algorithms attempting to compute support sets of minimisers of LASSO in order to do feature selection can be trusted. In this paper we establish how no (randomised) algorithm that works on all inputs can determine the correct support sets (with probability $> 1/2$) of minimisers of LASSO when reading approximate input, regardless of precision and computing power. However, we define a LASSO condition number and design an efficient algorithm for computing these support sets provided the input data is well-posed (has finite condition number) in time polynomial in the dimensions and logarithm of the condition number. For ill-posed inputs the algorithm runs forever, hence, it will never produce a wrong answer. Furthermore, the algorithm computes an upper bound for the condition number when this is finite. Finally, for any algorithm defined on an open set containing a point with infinite condition number, there is an input for which the algorithm will either run forever or produce a wrong answer. Our impossibility results stem from *generalised hardness of approximation* – within the Solvability Complexity Index (SCI) hierarchy framework – that generalises the classical phenomenon of hardness of approximation.

1. INTRODUCTION

In the wake of the many AI-based algorithms throughout the society and the sciences, potentially yielding hallucinations and instabilities [4, 23, 23, 24, 31, 34, 38, 40, 44, 46], the question of trustworthiness of algorithms is now becoming a crucial topic. This is particularly true for government regulators, where the European Commission [52] has been particularly vocal about its demand for trust in algorithms. However, with this new focus on trust in algorithms comes an important question: Which of the classical (non-AI-based) approaches are trustworthy? For example, can solutions to the *unconstrained LASSO feature/model selection* problem be trustworthily computed? See Remark 1.3 for a discussion of what one means by a trustworthy algorithm. The now classical LASSO feature selection approach, initiated by Tibshirani [60] is now standard in large parts of the sciences and is defined as follows, see also [15, 36, 37, 41].

Definition 1.1. For fixed $\lambda \in \mathbb{Q}$, $\lambda > 0$, the *unconstrained LASSO feature selection problem* is the following: given input $\iota = (y, A)$ with $y \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times N}$, find the support of some vector in

$$\mathsf{Sol}^{\mathsf{UL}}(b, U) := \operatorname*{argmin}_{\hat{x} \in \mathbb{R}^N} \|A\hat{x} - y\|_2^2 + \lambda \|\hat{x}\|_1. \tag{1.1}$$

The output set for input (y, A) is therefore

$$\Xi(y, A) = \{\mathsf{supp}(x) \,|\, x \in \operatorname*{argmin}_{\hat{x} \in \mathbb{R}^N} \|A\hat{x} - y\|_2^2 + \lambda \|\hat{x}\|_1\} \tag{1.2}$$

and we have $\Xi(y, A) \subseteq \mathbb{B}^N$ where $\mathbb{B} = \{0, 1\}$.

Remark 1.2 (Model of computation – Inexact input)**.** In practice, when trying to compute an element of $\Xi(y, A)$ in (1.2), we must assume that the A and y are given inexactly. This is because either we have: (1) an irrational input; or (2) the input is rational (for example $1/3$), but our computer expresses numbers in a certain base (typically base-2); (3) the computer uses floating-point arithmetic for which – in many cases – the common backward-error

analysis (popularized by Wilkinson [64]) translates the accumulation of round-off in a computation into a single-perturbation of the input data. Hence, in the sequel, we assume that algorithms access the input to whatever finite precision desired and that all computational operations are done exactly.

The key questions we address in this paper are the following:

> *When do there exist algorithms that can compute a support set in $\Xi(y, A)$, when (y, A) are given inexactly (yet with arbitrary large precision)? If trustworthy algorithms cannot exist for all inputs, when can we guarantee trustworthy computations?*

These questions touch on condition numbers, generalised hardness of approximation and robust optimisation.

Remark 1.3 (Trustworthiness of algorithms)**.** By 'trustworthy algorithm' for a computational problem, we mean the following. If the computational problem takes only discrete values (as is the case when computing support sets of minimisers of optimisation problems), a trustworthy algorithm will always produce a correct answer – if it halts. If the computational problem takes non-discrete values, a trustworthy algorithm produces an output – if it halts – that is at least $\epsilon \geq 0$ accurate in some predefined metric – for any $\epsilon > 0$ given as input to the algorithm.

1.1. **Condition and trustworthiness.** The phenomenon of small imprecisions on the input data leading to substantial errors on the output of an algorithm is a classical challenge in numerical analysis. Indeed, this magnification of the input error can in many cases be gauged by a notion of *condition*, which needs to be defined for each individual problem. This is crucial in the developments of trustworthy algorithms — that is algorithms with guaranteed error bounds. A *condition number* Cond is a map from the space of inputs to the interval $[0, \infty]$ that defines the sensitivity to small perturbations, with high values indicating sensitive inputs. Condition numbers were introduced independently by Turing [62] and von Neumann and Goldstine [63] in a pair of papers widely considered the birth certificate of contemporary numerical analysis. The goal was to understand the effects of finite precision in linear system solving to ensure trustworthy algorithms. Shortly after, condition numbers took also a role in the computation of complexity bounds. See [16, Overture] for a global picture about this.

1.2. **Generalised hardness of approximation (GHA) and robust optimisation.** The program on robust optimisation, pioneered by A. Ben-Tal, L. El Ghaoui & Nemirovski [11, 12, 48] among others, provides a powerful mathematical framework that addresses the challenge of optimisation given inaccuracies and uncertainty in the input. This model, as argued in Remark 1.2, is more realistic in view of real life computations — compared to models assuming exact computations — and aligns with S. Smale's call for "[Computational] models which process approximate inputs and which permit round-off computations" in the list of problems for the 21st century [59]. Recent developments in this area intersect with generalisations of the phenomenon of hardness of approximation [5], namely generalised hardness of approximation (GHA), initiated in [7] (see also [1, 3, 26, 29] and Problem 5 (J. Lagarias) in [28] for further results on GHA). GHA in optimisation is the phenomenon where one can easily compute an ϵ-approximation to a minimiser of the optimisation problem when $\epsilon > \epsilon_0 > 0$, but for $\epsilon < \epsilon_0$ (the approximation threshold) it suddenly becomes impossible regardless of computing power and accuracy on the input. This phase transition phenomenon was recently established [7, 28] for computing minimisers of the LASSO problem (1.1), and our impossibility results build on this framework and extend these results. Robust optimisation is classically concerned with the problem of approximating the optimal value of the objective function. However, a theory of robust optimisation for computing minimisers will necessarily include the GHA phenomenon.

Remark 1.4 (**Condition and GHA).** Condition numbers are well-known to often provide sufficient conditions for the existence of trustworthy algorithms in optimisation [16, 19, 53, 55–58]. J. Renegar's seminal work [53, 55–58] on condition numbers in optimisation start with a definition of ill-posed problems, and then a natural condition number is 1/(distance to the closest ill-posed problem). New developments demonstrate how, in certain situations, the necessity of having finite standard condition numbers in order to obtain trustworthy algorithms is not needed.

Indeed, the theory of GHA [1,3,7,26,29] demonstrate examples of classes of optimisation problems in the sciences that have infinite condition numbers that are standard in optimisation, yet efficient and trustworthy algorithms can handle the problems.

For linear programs and basis pursuit problems restricted to an input class Ω with inputs $\iota = (y, A)$, where $A \in \mathbb{R}^{m\times N}$ satisfies the robust nullspace condition [1] of order $s \geq 1$, with fixed parameters $\tau > 0$ and $\rho \in (0, 1)$, and $y = Ax$, where x is s-sparse, we have the following phenomenon [7]. There exists an algorithm that can compute approximate minimisers of the optimisation problem when A and y are given with inexact input. Moreover, such approximate minimisers can be computed in polynomial time in the number of variables and $\log(\epsilon^{-1})$ where $\epsilon > 0$ is the bound on the error of the approximate minimiser produced by the algorithm. As is standard in linear programming [16], the set of ill-conditioned problems is the collection of problems with several minimisers, yielding a condition number Cond for minimisers of linear programs. However, the set Ω contains input elements $\iota = (y, A)$ for which the condition number $\mathrm{Cond}(\iota) = \infty$. In particular, there exist polynomial time algorithms for vast input classes in the sciences for which classical condition numbers are infinite.

This suggests that the marriage of condition theory and GHA provide a much more refined theoretical framework designed to understand when trustworthy and efficient algorithms can be designed. This paper is the first step in this direction.

2. Main Result

Our paper continues the developments of condition in connection with GHA and provides both upper and lower bounds. We define in Section 6 a condition number $\mathscr{C}_{\mathrm{UL}}$ that gauges the impact of numerical errors when using feature selection via unconstrained LASSO. This quantity follows classical ideas of condition as the inverse of the distance to ill-posedness (see for instance [20] for a close ancestor of $\mathscr{C}_{\mathrm{UL}}$ for linear programming). We provide an alternative definition of $\mathscr{C}_{\mathrm{UL}}$ based on the Karush-Kuhn-Tucker (KKT) conditions in Section 7. Alongside the condition of the data at hand, the cost of solving LASSO also depends of the size of the data. We will measure this size with the following "truncated norms:"

$$[\![(b, U)]\!]_{\max} := \max\left\{\|U\|_{\max},\, \|b\|_\infty,\, 1\right\}, [\![(y, A)]\!]_{\mathrm{S}} := \max\left\{\sum_{i=1}^{m}\sum_{j=1}^{N}|A_{ij}|,\, \sum_{i=1}^{m}|y_i|,\, 1\right\}$$

where $\|U\|_{\max} = \max_{i,j}|U_{ij}|$ and $\|b\|_\infty = \max_i |b_i|$.

Our main result is the following (we will make the meaning of "variable-precision approximations" precise in Section 4).

Theorem 2.1. *Consider the condition number $\mathscr{C}_{\mathrm{UL}}(b, U)$ defined in* (6.1)*.*

(1) We exhibit an algorithm Γ which, for any input pair $(b, U) \in \mathbb{R}^m \times \mathbb{R}^{m\times N}$, reads variable-precision approximations of (b, U). If $\mathscr{C}_{\mathrm{UL}}(b, U) < \infty$ then the algorithm halts and returns a correct value in $\Xi(b, U)$. The cost of this computation is

$$\mathcal{O}\left\{N^3\left[\log_2\left(N^2[\![(b, U)]\!]^2_{\max}\mathscr{C}_{\mathrm{UL}}(b, U)\right)\right]^2\right\}$$

and the maximum number of digits the algorithm accesses is bounded by

$$\mathcal{O}(\lceil\log_2\left(\max\{\lambda + \lambda^{-1}, N, [\![(b, U)]\!]_{\mathrm{S}}, \mathscr{C}_{\mathrm{UL}}(b, U)\}\right)\rceil).$$

If, instead, $\mathscr{C}_{\mathrm{UL}}(b, U) = \infty$ then the algorithm runs forever.

(2) The condition number $\mathscr{C}_{\mathrm{UL}}(b, U)$ can be estimated in the following sense: There exists an algorithm that provides an upper bound on $\mathscr{C}_{\mathrm{UL}}(b, U)$, when it is finite, and runs forever when $\mathscr{C}_{\mathrm{UL}}(b, U) = \infty$.

(3) If $\Omega \subseteq \mathbb{R}^m \times \mathbb{R}^{m\times N}$ is an open set and there is a $(b, U) \in \Omega$ with $\mathscr{C}_{\mathrm{UL}}(b, U) = \infty$ then there is no algorithm that, for all inputs $(y, A) \in \Omega$, computes an element of $\Xi(y, A)$ given approximations to $(y, A) \in \Omega$. Moreover, for any randomised algorithm Γ^{ran} that always halts and any $p > 1/2$, there exists a $(y, A) \in \Omega$ and an approximate representation $(\tilde{y}, \tilde{A})$ (see §9) of (y, A) so that $\Gamma^{\mathrm{ran}}(\tilde{y}, \tilde{A}) \notin \Xi(y, A)$ with probability at least p.

If $(b, U) \in \Omega$ is computable, then the failure point $(y, A) \in \Omega$ above can be made computable.

Remark 2.2 (**Our algorithm never produces wrong outputs).** In terms of trustworthiness – which is the main topic of this paper – our algorithm will never make a mistake. In the cases where it fails, it will simply run forever. Note that according to Theorem 2.1, this is optimal for any algorithm that will work on open sets of inputs, as the alternative to not halting is producing a wrong output when $\mathscr{C}_{\mathrm{UL}}(b, U) = \infty$.

The complexity bound presented in Theorem 2.1 is pleasantly low, exhibiting a cubic dependence on N and logarithmic dependence on both the size of the data and its condition number. It is worth highlighting that (3) within Theorem 2.1 implies that our defined condition number effectively captures the challenging aspects of solving unconstraint LASSO with open input sets. Indeed, open sets of inputs containing elements with an infinite condition number inherently preclude the application of reliable algorithms. Our current approach does not provide a means to determine when the condition number becomes infinite. Generally, it is widely believed that discerning with finite precision whether a data has finite or infinite condition number is, in most cases, not possible (cf. [54]). The estimate of this condition number is, for a broad class of problems, as challenging as solving the underlying problem itself [21]. This insight was promptly recognized by von Neumann and Goldstine, prompting them to address this issue in their sequel [30] to [63].

The proposed solution in that sequel, which has since become a commonly followed approach, involves imbuing the data space with a probability measure and deriving probabilistic estimates for the condition number. These estimates, in turn, lead to probabilistic bounds on complexity and accuracy. A sequel [6] of our paper proceeds along these lines.

Remark 2.3 (**Condition number and robust optimisation).** Both the fields of condition in optimisation and robust optimisation share a common objective: ensuring stable and precise computations even in the presence of imprecise input data. However, historically, these two domains have often remained somewhat distinct within mathematics. A reason for this is that robust optimisation has focused on computing the optimal value and not the minimisers. As our negative results in Theorem 2.1 demonstrate, providing a theory about properties of minimisers within robust optimisation, necessarily involves condition numbers. In particular, our results illustrate how a theory of robust feature selection through optimization inherently links robust optimization and the theory of condition.

2.1. **Connection to previous work.** Below follows an account of the connection to different areas and works that are crucial for the paper.

Condition in optimisation: Condition numbers in computational mathematics and numerical analysis have been a mainstay [27, 39] in order to secure trustworthy algorithms that are accurate and stable. In optimisation J. Renegar has pioneered the theory of condition numbers [53, 55–58] both from the perspective of stability and accuracy, but also from the perspective of efficiency of algorithms. All of these perspectives are covered in detail in [16]. In addition, we want to mention the work of J. Peña [50, 51] as well as D. Amelunxen, M. Lotz, J. Walvin [43], and D. Amelunxen, M. Lotz, M. McCoy, J. Tropp [2] see also [19, 20, 22].

GHA and robust optimisation: GHA [1, 3, 7, 26, 29] is in spirit (although mathematically very different) close to hardness of approximation in computer science [5]. However, GHA in optimisation can be viewed as a part of the program on robust optimisation (A. Ben-Tal, L. El Ghaoui & Nemirovski [11, 12, 48]) for computing minimisers. It is also a part of the greater program on the mathematics behind the Solvability Complexity Index (SCI) hierarchy, see for example the work by J. Ben-Artzi, M. Colbrook, M. Marletta [9, 10, 25, 35].

Trustworthy algorithms and computer assisted proofs: Trustworthy algorithms in optimisation go beyond scientific computing and have important implications in computer assisted proofs in mathematics, where T. Hales' proof of Kepler's conjecture [32, 33] is a star example. The intricate computer assisted proof relies on computing around 50,000 linear programs with irrational inputs, which leads to the crucial problem of computing with inexact inputs. The reader may also want to consult [28], in particular Problem 2 (T. Hou) and Problem 5 (J. Lagarias) on

ϵ	x_1	x_2	Features computed	Features correctly computed
10^{-1}	0	0.95	$\{2\}$	Yes
10^{-2}	0	0.95	$\{2\}$	Yes
10^{-3}	0	0.95	$\{2\}$	Yes
10^{-4}	0.45	0.5001	$\{1,2\}$	No
10^{-5}	0.9	0.0500	$\{1,2\}$	No
10^{-6}	0.945	0	$\{1\}$	No

TABLE 1. The computation done in Example 3.1. Note that there is an error for $\epsilon = 10^{-4}, 10^{-5}$ or 10^{-6}.

a paper that discusses the tradition of developing algorithms that are 100% trustworthy and even suitable for computer assisted proofs.

Algorithms for computing minimisers of LASSO: There is, of course, a variety of algorithms suitable for the LASSO problem. We refer to the review articles by Nesterov & Nemirovski [49] and Chambolle & Pock [18], and the references therein for a more complete overview. See also the work by Beck & Teboulle [8] and Wright, Nowak & Figueiredo [66], and the references therein, as well as [1, 15, 65]. However, while these algorithms can compute approximations to the objective function, they cannot – in general – compute the support sets of minimisers of LASSO.

3. NUMERICAL EXAMPLES – FAILURE OF MODERN ALGORITHMS

Our first numerical example shows how this idea can work in practice.

Example 3.1. Assume that a dependent variable b depends on two features b_1, b_2 in the following way: $b = u_1/(1-\epsilon) = u_2$ where the parameter $\epsilon \in (0,1)$. In particular, b is strongly correlated with u_2 and this is the best predictor (using the lasso) for b. Suppose that we observe b, u_1 and u_2 and obtain the measurements $u = 1$, and $U = \begin{pmatrix} u_1 & u_2 \end{pmatrix} = \begin{pmatrix} 1-\epsilon & 1 \end{pmatrix}$. The LASSO problem with input (b, U) has a unique solution x^* with $\mathsf{supp}(x^*) = \{2\}$ provided $\lambda < 2$. We tested the accuracy of LASSO solvers under finite precision by computing solutions to the LASSO problem with $\lambda = 10^{-1}$ in the following way: first, we used MATLAB's lasso routine to attempt to find a minimiser. Next, we set any values of this minimiser that were smaller in magnitude than 10^{-2} to 0 (not doing so would systematically lead to minimisers with full support). We then computed the support of the resulting vector x. Table 1 presents the results for the computed vector x and its support. For small values of ϵ, the execution was unable to identify the zero component of the true solution x^* of the unconstrained LASSO problem. There is, of course, a variety of algorithms suitable for the LASSO problem [17, 18], however, Theorem 2.1 is universal, and thus for any algorithm there will be inputs for which the algorithm will fail.

An obvious extension of Example 3.1 is to move from the deterministic inputs of Example 3.1 to inputs chosen according to a random distribution. We thus look at a simple random example, where we can once again compute by hand the true solution and thus contrast with the output of the algorithm.

Example 3.2. Again we consider the single measurement case and set $b = 1$, and $\lambda = 10^{-2}$ but this time we assume that we observe N features, $u_1, \ldots, u_N$, which are randomly and independently drawn according to either the exponential distribution with parameter 1, the normal distribution with mean 1 variance 10^{-4}, and the uniform distribution on $(0,1)$. This purposefully simplified situation is considered because it is easy to derive the true solution to the feature set induced by the solution of (1.1). We compare this answer (the ground truth) with the following procedure: we use Matlab's lasso routine to attempt to compute an element x in $\mathsf{Sol}^{\mathsf{UL}}(b, U)$ and then set any values of x larger in absolute value than a parameter 'threshold' to 0 and consider the resulting vector's support.

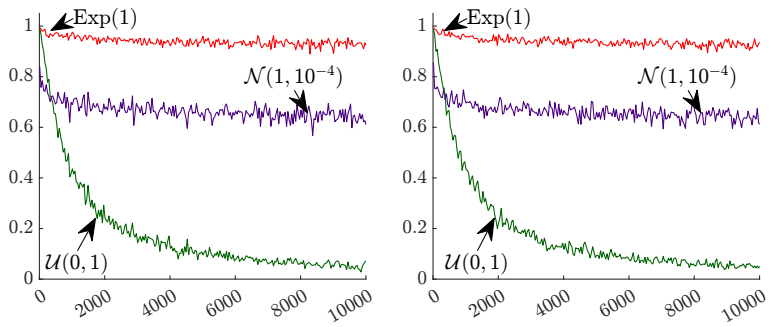

FIGURE 1. Outputs of the computation done in Example 3.2 – where each entry of U is IID according to the distributions $\mathcal{U}(a, b)$ (uniform), $\mathrm{Exp}(\nu)$ (exponential) and $\mathcal{N}(\mu, \sigma^2)$ (normal). The task is to compute the support set of a LASSO minimiser i.e. an element in $\Xi(b, U)$ from (1.2) with $\lambda = 10^{-2}$. The horizontal axis represents the dimension N. The vertical axis represents the success rate $\frac{\#\text{ of successes}}{\#\text{ of trials}}$ with threshold value – see the text accompanying in Example 3.2 – set to 10^{-3} and 10^{-12} for the left and right figures respectively.

We do this over five hundred randomly generated instances $U = (u_1, \ldots, u_N)$ for each choice of $N \in \{10, 20, \ldots, 10010\}$ inclusive. The results are presented in Figure 1. We see that for large N, the algorithm is more accurate when data are drawn from an exponential distribution instead of this normal distribution and similarly the algorithm is more accurate when data are drawn from a normal distribution instead of a uniform one.

4. PRELIMINARIES ON THE COMPUTATIONAL MODEL

To talk about algorithms and complexity it is necessary to fix a computational model. In our case, we can take either a BSS machine [13, 14] or a Turing machine. The only assumption we will do on these machines is that they can access arbitrarily precise approximations to the input data (and that the machine "knows" the precision of these approximations). It is clear that otherwise, there would be no hope to return a correct output. It is also clear that the cost of accessing one such approximation will have to depend of the precision required. We next specify this.

We will assume that, for any pair $(b, U) \in \mathbb{R}^m \times \mathbb{R}^{m \times N}$ we want to solve, the machine is given as input two procedures $\mathsf{GetMatrix}^U(\,)$, $\mathsf{GetVector}^b(\,)$ taking themselves as input a natural number $n \in \mathbb{N}$ which return some matrix U^n and vector b^n respectively such that

$$|U^n_{ij} - U_{ij}| \leq 2^{-n}, \quad |b^n_i - b_i| \leq 2^{-n} \quad \text{for } i = 1, 2, \ldots, m \text{ and } j = 1, 2, \ldots, N. \tag{4.1}$$

The procedures $\mathsf{GetMatrix}^U(\,)$, $\mathsf{GetVector}^b(\,)$ are *black boxes* or *oracles*. We do not suppose any specific implementation for them, merely their correctness.

Of course, for any fixed (b, U) there are multiple choices of b^n and U^n satisfying (4.1). Crucially, the algorithm must work for any such choice. That is to say, the algorithm has to work with whichever choice of b^n or U^n the oracles return, it can only rely on the fact that (b^n, U^n) approximates (b, U) with a precision given by (4.1). But to be consistent with both the BSS and Turing choices, we want U^n and b^n to have rational entries[1]. And for complexity reasons (we will rely on existing algorithms for convex quadratic programming that take rational inputs) we also want to control the bit-size of the entries of b^n and U^n. We now observe that for any $x \in \mathbb{R}$ and $n \geq 1$, we can compute a rational number $\tilde{x}$ such that $|\tilde{x} - x| \leq 2^{-n}$ with

$$\text{bit-size}(\tilde{x}) \leq \lceil \log_2(1 + |x|) \rceil + n. \tag{4.2}$$

[1]This consistency refers to our algorithm in Theorem 2.1(1) which can be thought either as a Turing or a BSS algorithm as it works over rational numbers. We note, however, that the complexity bound in the statement refers to the BSS setting. Also, part (3) of the theorem holds for both Turing and BSS algorithms.

Hence, we will strengthen assumption (4.1) with the bounds (4.2) for the entries of U_{ij}^n and b_i^n.

Furthermore, the computation of $\tilde{x}$ above can be done with a cost of $\mathcal{O}(\lceil \log_2(1+|x|)\rceil + n)$ arithmetic operations. This suggests a natural cost of $\mathcal{O}(mN(\log_2(\|U\|_{\max}+1)+n))$ for a call to $\mathsf{GetMatrix}^U(n)$ and one of $\mathcal{O}(m(\log_2(\|b\|_\infty+1)+n))$ for a call to $\mathsf{GetVector}^b(n)$. We will therefore adopt these costs.

We also make the assumption that $\lambda \in \mathbb{Q}$ and that the algorithm can access the value of λ exactly. This is because this parameter is usually configurable by an end user working on the feature selection problem, and can thus be chosen to be a rational number. This is not the case for b and U which are usually taken from real world datasets and may be irrational. It goes without saying, at least in the Turing model, the bit-size $p(\lambda)$ of λ affects the cost of the computation. But for a fixed λ the bound in Theorem 2.1 holds. Nonetheless, we will explicitly quantify this effect in Section 8.

In addition to $[\![(y,A)]\!]_{\max}$ we will also use the ℓ^p-norm $\|y\|_p$ of y, the operator norms $\|A\|_{qr} = \sup_{\|x\|_q=1}\|Ax\|_r$ (writing $\|A\|_q$ when $q=r$), and the additional 'truncated norm' $[\![(y,A)]\!]_2 := \max\{\|A\|_2, \|y\|_2, 1\}$.

5. Standard facts about the unconstrained LASSO

In this section we will describe some well-known facts about the unconstrained LASSO selector. We make no claims to novelty — instead, this section exists only to supplement the rest of the article.

For a pair (y,A), let $\mathsf{Sol}^{\mathsf{UL}}(y,A)$ be the set of solutions in $\mathbb{R}^N$ to unconstrained LASSO with input (y,A). More precisely, we set

$$\mathsf{Sol}^{\mathsf{UL}}(y,A) := \operatorname*{argmin}_{\hat{x}\in\mathbb{R}^N} \|A\hat{x}-y\|_2^2 + \lambda\|\hat{x}\|_1.$$

Note that $\mathsf{Sol}^{\mathsf{UL}}(y,A)$ is a set and not necessarily single valued: a priori, multiple vectors may minimise the unconstrained LASSO functional. Then, we recall, $\Xi(y,A) = \{\mathsf{supp}(x) \mid x \in \mathsf{Sol}^{\mathsf{UL}}(y,A)\}$.

The following facts are well-known:

(UL1): If $x \in \mathsf{Sol}^{\mathsf{UL}}(y,A)$ we must have $\|Ax-y\|_2^2, \lambda\|x\|_1 \le \|Ax-y\|_2^2 + \lambda\|x\|_1 \le \|y\|_2^2$. Thus $\mathsf{Sol}^{\mathsf{UL}}(y,A)$ can be written equivalently as

$$\mathsf{Sol}^{\mathsf{UL}}(y,A) = \operatorname*{argmin}_{\hat{x}\in\mathbb{R}^N,\, \|\hat{x}\|_1 \le \lambda^{-1}\|y\|_2^2} \|A\hat{x}-y\|_2^2 + \lambda\|\hat{x}\|_1.$$

For each fixed (y,A), this is a minimisation problem of a continuous function (of $\hat{x}$) over a non-empty compact region. Thus $\mathsf{Sol}^{\mathsf{UL}}(y,A)$ is always non-empty and compact. This situation is different from that of linear programming, where there exist inputs without optimal solutions.

(UL2): As the objective function (that is to say, the function mapping $\hat{x}$ to $\|A\hat{x}-y\|_2^2 + \lambda\|\hat{x}\|_1$) is convex, the unconstrained LASSO problem is convex.

(UL3): As mentioned before, it is not necessary that the unconstrained LASSO problem has a unique minimiser. However, as the problem is convex and the feasible region is non-empty, either there is a unique minimiser or there are infinitely many minimisers. Understanding uniqueness will prove to be important to this paper and has been examined in detail in [61].

(UL4): As argued in [61], if $x^1, x^2 \in \mathsf{Sol}^{\mathsf{UL}}(y,A)$ then $Ax^1 = Ax^2$. Thus $\|Ax^1-y\|_2^2 = \|Ax^2-y\|_2^2$ and so $\|x^1\|_1 = \|x^2\|_1$.

(UL5): The KKT conditions are both necessary and sufficient. That is,

$$x \in \mathsf{Sol}^{\mathsf{UL}}(y,A) \iff 2A^*(Ax-y) = -\lambda\,\mathfrak{s}(x)$$

where

$$(\mathfrak{s}(x))_i \begin{cases} = 1 & \text{if } x_i > 0 \\ = -1 & \text{if } x_i < 0 \\ \in [-1,1] & \text{if } x_i = 0. \end{cases}$$

In particular, combining this with (UL4) we see that $\mathfrak{s}(x)$ is constant over all $x \in \mathsf{Sol}^{\mathsf{UL}}(y,A)$.

Remark 5.1. Note that $\mathfrak{s}(x)$ is closely related to the sign function sgn, but is multivalued when applied to 0 whereas $\mathsf{sgn}(0)$ is traditionally defined to be 0. In fact, $\mathfrak{s}(x)$ is the convex subdifferential of the ℓ^1 norm applied at x.

6. Conditioning of unconstrained LASSO

It is sensible to define a quantity that measures how variations in the input to a LASSO problem affect the support of a solution. We thus define the *stability support*.

Definition 6.1. The *stability support* of a pair (y, A) is defined as

$$\mathsf{stsp}(y,A) := \inf\Big\{\delta \geq 0 \,\Big|\, \exists\, \tilde{y} \in \mathbb{R}^m, \tilde{A} \in \mathbb{R}^{m\times N}, x \in \mathsf{Sol}^{\mathsf{UL}}(y,A), \text{ and } \tilde{x} \in \mathsf{Sol}^{\mathsf{UL}}(\tilde{y},\tilde{A}) \\ \text{such that } \|\tilde{y}-y\|_\infty, \|A-\tilde{A}\|_{\max} \leq \delta \text{ and } \mathsf{supp}(x) \neq \mathsf{supp}(\tilde{x})\Big\}.$$

The stability support is therefore the *distance to support change*. If $\mathsf{stsp}(y, A) > 0$ then there exists $S \in \mathbb{B}^N$ such that $\Xi(y, A) = \{S\}$. Furthermore, for all pairs (y', A') in a ball (w.r.t. the max distance) of radius $\mathsf{stsp}(y, A)$ around (y, A) we have $\Xi(y', A') = \{S\}$. If, instead, $\mathsf{stsp}(y, A) = 0$ then there are arbitrarily small perturbations of (y, A) which yield LASSO solutions with different support.

The notion of stability support leads to a natural definition of condition for the unconstrained LASSO feature selection problem.

Definition 6.2. For an input (y, A) to UL feature selection we define the *condition number* $\mathscr{C}_{\mathrm{UL}}(y, A)$ to be

$$\mathscr{C}_{\mathrm{UL}}(y,A) = \begin{cases} (\mathsf{stsp}(y,A))^{-1} & \text{if } \mathsf{stsp}(y,A) \neq 0 \\ \infty & \text{otherwise.} \end{cases} \tag{6.1}$$

The set $\Sigma_{\mathrm{UL}} := \{(y, A) \mid \mathsf{stsp}(y, A) = 0\}$ is the *set of ill-posed inputs*.

Remark 6.3. For scale-invariant problems it is common to define condition as the normalized inverse of the distance to ill-posedness. That is, for a data a, as $\|a\|/d(a, \Sigma)$ where Σ denotes the set of ill-posed inputs, $\|\ \|$ is some norm measuring the size of the input data and d is some distance metric, usually the one induced by $\|\ \|$ (see [16, §6.1] for a discussion on this). The fact that UL is not scale invariant explains why the condition number $\mathscr{C}_{\mathrm{UL}}$ is not normalized. To better understand this issue it is worth considering a simple example. Let us consider the input $(y, A) = (0, 0)$. For any $\lambda > 0$, it is easy to see that $\mathsf{Sol}^{\mathsf{UL}}(y, A) = \{0\}$. Moreover, if both $\|y' - y\|_\infty = \|y'\|_\infty \leq \epsilon$ and $\|A' - A\|_{\max} = \|A'\|_{\max} \leq \epsilon$ then $\|A'0 - A'y'\|_\infty = \|A'y'\|_\infty < \lambda$, provided that $\epsilon \leq \sqrt{\lambda N^{-1}}$. Thus 0 satisfies the KKT conditions and so $0 \in \mathsf{Sol}^{\mathsf{UL}}(y', A')$. Since all LASSO solutions have the same ℓ^1 norm (by (UL4)), we conclude that $\mathsf{Sol}^{\mathsf{UL}}(y', A') = \{0\}$. Thus $\mathsf{stsp}(y, A) > 0$ and so the input $(0, 0)$ should be considered well-posed. If we were to define the condition number as a norm over a distance, we would have that $\mathscr{C}_{\mathrm{UL}}(y, A) = 0$. Such perfect conditioning would be unhelpful in understanding how close the data is to instability. Hence, we do not define the condition number in the traditional way. We instead opt to define $\mathscr{C}_{\mathrm{UL}}$ as in (6.1) so that the condition of $(0, 0)$ is both positive and finite.

From the definition, it is immediately obvious that if $(y, A) \notin \Sigma_{\mathrm{UL}}$ then $\Xi(y, A)$ must be single valued. We conclude this section by noting that this property transfers to multivaluedness of $\mathsf{Sol}^{\mathsf{UL}}$. We will prove this result in Section 8.

Proposition 6.4. *If* $(y, A) \notin \Sigma_{\mathrm{UL}}$ *then* $|\mathsf{Sol}^{\mathsf{UL}}(y, A)| = 1$.

7. An alternative characterisation of the condition number

In this section we present a different characterisations of ill-posedness based on three values, $\sigma_1, \sigma_2, \sigma_3$, which we can use to computationally approximate the condition number.

Definition 7.1. For a pair (y, A), we write (with the convention that if M is non-invertible, $\|M^{-1}\|_2 := \infty$ and so $\|M^{-1}\|_2^{-1} = 0$),

$$\sigma_1(y, A) := \inf\{t \,|\, \exists x \in \mathsf{Sol}^{\mathsf{UL}}(y, A) \text{ with } \|A^*_{S^c}(Ax - y)\|_\infty = \lambda/2 - t, S = \mathsf{supp}(x)\},$$
$$\sigma_2(y, A) := \inf\{\|(A^*_S A_S)^{-1}\|_2^{-1} \,|\, \exists x \in \mathsf{Sol}^{\mathsf{UL}}(y, A) \text{ with } S = \mathsf{supp}(x)\},$$
$$\sigma_3(y, A) := \inf\{t \,|\, \exists i \in \{1, 2, \ldots, N\} \text{ and } x \in \mathsf{Sol}^{\mathsf{UL}}(y, A) \text{ such that } 0 < |x_i| \leq t\}.$$

where, for the empty-set $\varnothing$, we interpret $\|A^*_\varnothing(Ax-y)\|_\infty = 0$, we treat $A^*_\varnothing A_\varnothing$ as invertible with $\|(A^*_\varnothing A_\varnothing)^{-1}\|_2^{-1} = \infty$, and we set $\inf \varnothing = \infty$.

To get an intuition of what these σs gauge assume momentarily that Proposition 6.4 holds and let $\mathsf{Sol}^{\mathsf{UL}}(y, A) = \{x\}$. Then σ_1 gauges how close the non-support set of x is to violating the KKT conditions; it is small if $\|A^*_{S^c}(Ax - y)\|_\infty$ is close to $\lambda/2$. Similarly, σ_2 is small if $\|(A^*_S A_S)^{-1}\|_2$ is large for the support set S of x (and thus $A^*_S A_S$ is close to being non-invertible), and σ_3 is small if x has a small component on its support.

We combine each of σ_1, σ_2 and σ_3 into a single quantity as follows,

$$\sigma(y, A) := \min\left\{\sigma_1(y, A), \sigma_2(y, A)^2, \sigma_3(y, A)\right\}$$

At first glance, it might seem that computing σ is difficult owing to the infima in Definition 7.1. The next proposition shows how these infima can be removed so that σ can be easily calculated given an $x \in \mathsf{Sol}^{\mathsf{UL}}(y, A)$.

Proposition 7.2. *Let $x \in \mathsf{Sol}^{\mathsf{UL}}(y, A)$ have support S.*

(1) If $\|A^*_{S^c}(Ax - y)\|_\infty < \lambda/2$ and $A^*_S A_S$ is invertible, then $|\mathsf{Sol}^{\mathsf{UL}}(y, A)| = 1$ and so

$$\sigma_1(y, A) = \lambda/2 - \|A^*_{S^c}(Ax - y)\|_\infty, \quad \sigma_2(y, A) = \|(A^*_S A_S)^{-1}\|_2^{-1}$$
$$\sigma_3(y, A) = \inf\{|x_i| \,|\, i \in S\}.$$

(2) If instead $\|A^*_{S^c}(Ax - y)\|_\infty = \lambda/2$ or $A^*_S A_S$ is not invertible, then $\sigma(y, A) = 0$.

We are now interested in the relation between σ and stsp. The following proposition provides a bound from above from stsp in terms of σ.

Proposition 7.3. *For $(y, A) \in \mathbb{R}^m \times \mathbb{R}^{m\times N}$, $\mathsf{stsp}(y, A)$ is bounded above by $\sigma(y, A)$ as follows,*

(1) For $\sigma_1(y, A) < \lambda/4$, we have $\mathsf{stsp}(y, A) \leq \frac{4\|A\|_{\max}\sigma_1(y,A)}{\lambda}$.
(2) We have $\mathsf{stsp}(y, A) \leq \sqrt{\sigma_2(y, A)}$.
(3) We have $\mathsf{stsp}(y, A) \leq \|A\|_{\max}\sigma_3(y, A)$.

Similarly, a lower bound which makes use of the following polynomial (defined on positive ν, ξ)

$$q(\nu, \xi) := 96\nu^5 + 12\nu^3(1 + \lambda\sqrt{N})\sqrt{\xi} + \xi\left(\frac{2\nu^3}{\lambda} + 3\nu\right) \tag{7.1}$$

is given below.

Proposition 7.4. *Set $\alpha = [\![(y, A)]\!]_2$ and $\sigma = \sigma(y, A)$. Then*

$$\mathsf{stsp}(y, A) \geq (mN)^{-\frac{1}{2}} \min\left\{\frac{\sigma^2}{q(\alpha, \sigma)}, \frac{\sqrt{\sigma}}{6\alpha}, \alpha\right\}.$$

Propositions 7.2 and 7.3 allow us to compute upper bounds for the condition number $\mathscr{C}_{\mathrm{UL}}(y, A)$, whilst Proposition 7.4 ensures that these estimates are accurate. There is also another consequence of these results: they allow us to provide an alternative definition for the condition number.

Definition 7.5. For $(y, A) \in \mathbb{R}^m \times \mathbb{R}^{m\times N}$, (y, A) is said to be *σ-ill-posed* for the UL feature selection problem if $\sigma(y, A) = 0$.

Propositions 7.3 and 7.4 then show that the set of σ-ill-posed problems is exactly the set Σ_{UL}. Thus an alternative definition of the condition number for (y, A) is to define it as the reciprocal of the distance to the set of σ-ill-posed problems. Whilst σ is convenient from a computational perspective, it is more intuitive to define condition in terms of stsp.

8. Proofs of the stated results

8.1. **Proof of Proposition 6.4.** Before proving Proposition 6.4, we need to introduce the concept of the set of solutions with minimal support.

Definition 8.1. The set of *solutions with minimal support* of a pair (y, A) is

$$\mathsf{Sol}^{\mathsf{ms}}(y, A) := \{x \in \mathsf{Sol}^{\mathsf{UL}}(y, A) \,|\, \forall x' \in \mathsf{Sol}^{\mathsf{UL}}(y, A),\, \mathsf{supp}(x') \subseteq \mathsf{supp}(x) \Rightarrow x' = x\}. \tag{8.1}$$

Lemma 8.2. *If* $|\mathsf{Sol}^{\mathsf{UL}}(y, A)| \neq 1$ *then* $|\mathsf{Sol}^{\mathsf{ms}}(y, A)| \geq 2$.

Proof. The set $\mathsf{Sol}^{\mathsf{UL}}(y, A)$ is compact, convex, and non-empty (from (UL1-2)). In particular, by the Krein-Milman Theorem [47, Theorem 9.4.6], $\mathsf{Sol}^{\mathsf{UL}}(y, A)$ is the closed convex hull of its extreme points (that is, points $p \in \mathsf{Sol}^{\mathsf{UL}}(y, A)$ so that if $p' \in \mathbb{R}^N$ is such that $p+p' \in \mathsf{Sol}^{\mathsf{UL}}(y, A)$ and $p-p' \in \mathsf{Sol}^{\mathsf{UL}}(y, A)$ then $p' = 0$ [47, Theorem 9.2.2(d)]). Therefore there must be at least two extreme points of $\mathsf{Sol}^{\mathsf{UL}}(y, A)$; otherwise, $|\mathsf{Sol}^{\mathsf{UL}}(y, A)| = 1$.

To complete the proof, we now show that every extreme point of $\mathsf{Sol}^{\mathsf{UL}}(y, A)$ is in $\mathsf{Sol}^{\mathsf{ms}}(y, A)$. Suppose otherwise, that is, that there is an extreme point $x \in \mathsf{Sol}^{\mathsf{UL}}(y, A)$ and an $x' \in \mathsf{Sol}^{\mathsf{UL}}(y, A)$ with $x' \neq x$ and $\mathsf{supp}(x') \subseteq \mathsf{supp}(x)$. Then (by (UL4-5)), $Ax = Ax'$, $\mathfrak{s}(x) = \mathfrak{s}(x')$ and $\|x\|_1 = \|x'\|_1$. For $\epsilon > 0$, let $v = (1+\epsilon)x - \epsilon x'$. If ϵ is sufficiently small we have that $\mathfrak{s}(v) = \mathfrak{s}(x)$ and hence

$$\begin{aligned}\|v\|_1 = \sum_{i \in \mathsf{supp}(v)} v_i \mathfrak{s}(v_i) = \sum_{i \in \mathsf{supp}(x)} v_i \mathfrak{s}(x_i) = \sum_{i \in \mathsf{supp}(x)} (1+\epsilon)|x_i| - \epsilon|x'_i| \\ = (1+\epsilon)\|x\|_1 - \epsilon\|x'\|_1 = \|x\|_1.\end{aligned}$$

Moreover, $Av = Ax$. We thus conclude that $v \in \mathsf{Sol}^{\mathsf{UL}}(y, A)$. But this contradicts the extremality of x (take $p' = \epsilon(x - x') \neq 0$; then $x + p' = v \in \mathsf{Sol}^{\mathsf{UL}}(y, A)$ and $x - p' = (1-\epsilon)x + \epsilon x' \in \mathsf{Sol}^{\mathsf{UL}}(y, A)$ since this set is convex) and thus all extreme points of $\mathsf{Sol}^{\mathsf{UL}}(y, A)$ are in $\mathsf{Sol}^{\mathsf{ms}}(y, A)$. We have already shown that there are multiple extreme points in $\mathsf{Sol}^{\mathsf{UL}}(y, A)$, thus completing the proof. □

This immediately implies Proposition 6.4: indeed, if $|\mathsf{Sol}^{\mathsf{UL}}(y, A)| \neq 1$ then by Lemma 8.2 there exist $v^1, v^2 \in \mathsf{Sol}^{\mathsf{UL}}(y, A)$ with $\mathsf{supp}(v^1) \neq \mathsf{supp}(v^2)$. By the definition of stsp, we must have $\mathsf{stsp}(y, A) = 0$.

8.2. **Proof of Proposition 7.2.** To prove part (1), first note that the result is trivial if $S = \varnothing$: indeed, in this case $x = 0$ and hence by (UL4) the solution is unique. We thus consider the case where $S \neq \varnothing$. Assume that the vector $\tilde{x}$ is such that $\tilde{x} \in \mathsf{Sol}^{\mathsf{UL}}(y, A)$. Then $A\tilde{x} = Ax$ (note that this was stated in Section 5 as (UL4)) and so $\|A^*_{S^{\mathsf{c}}}(A\tilde{x} - y)\|_\infty < \lambda/2$. By the KKT conditions this implies that $\mathsf{supp}(\tilde{x}) \subseteq S$ (note, in the case $S^{\mathsf{c}} = \varnothing$ this is trivial). But then $A_S\tilde{x}_S = A\tilde{x} = Ax = A_S x_S$. Since $A^*_S A_S$ is invertible by assumption, it must have a trivial kernel and hence $x_S = \tilde{x}_S$. Finally, since both $\mathsf{supp}(x), \mathsf{supp}(\tilde{x}) \subseteq S$ we must have $x = \tilde{x}$. Thus $|\mathsf{Sol}^{\mathsf{UL}}(y, A)| = 1$. The result about σ_1, σ_2 and σ_3 in this circumstance follows from the fact that in this case, the infima in Definition 7.1 are taken over a single vector with finitely many entries.

Part 2 follows immediately from the definition of σ: note that $\|A^*_{S^{\mathsf{c}}}(Ax - y)\|_\infty = \lambda/2$ implies that $\sigma_1 = 0$ and non-invertibility of $A^*_S A_S$ implies that $\sigma_2 = 0$. □

8.3. **Proof of Proposition 7.3.** We begin with the following Lemma.

Lemma 8.3. *If* $\sigma_2(y, A) = 0$ *then* $(y, A) \in \Sigma_{\mathrm{UL}}$.

Proof. By the definition of σ_2, if $\sigma_2(y, A) = 0$ then there is a minimiser $x \in \mathsf{Sol}^{\mathsf{UL}}(y, A)$ and a set $S \neq \varnothing$ such that $\mathsf{supp}(x) = S$ and $A^*_S A_S$ is non-invertible. In particular, since $A^*_S A_S \in \mathbb{R}^{|S|\times|S|}$ it must also have a non-trivial nullspace. Let $v \in \mathbb{R}^N$ be such that $A^*_S A_S v_S = 0$ (so that $\|A_S v_S\|_2^2 = \langle v_S, A^*_S A_S v_S\rangle = 0$) and $v_{S^{\mathsf{c}}} = 0$. For $\epsilon > 0$ sufficiently small we must have $\mathfrak{s}(x_S + \epsilon v_S) = \mathfrak{s}(x_S)$ and so $\|x_S \pm \epsilon v_S\|_1 = \langle \mathfrak{s}(x_S), x_S \pm \epsilon v_S\rangle = \|x\|_1 \pm \epsilon\langle \mathfrak{s}(x_S), v_S\rangle$. Thus at least one of $\|x_S + \epsilon v_S\|_1, \|x_S - \epsilon v_S\|_1$ is bounded above by $\|x\|_1$. Assume that $\|x_S + \epsilon v_S\|_1 \leq \|x\|_1$ (the argument for $\|x_S - \epsilon v_S\|_1 \leq \|x\|_1$ is identical). In this case, we have $A^*_S A_S(x_S + \epsilon v_S) = A^*_S A_S x_S = -\lambda \mathfrak{s}(x_S)/2 = -\lambda \mathfrak{s}(x_S + \epsilon v_S)/2$ and $\|A^*_{S^{\mathsf{c}}} A_S(x_S + \epsilon v_S)\|_\infty = \|A^*_{S^{\mathsf{c}}} A_S x_S\|_\infty \leq \lambda/2$. Therefore

$(x + \epsilon v)$ obeys the KKT conditions and so $(x + \epsilon v) \in \mathsf{Sol}^{\mathsf{UL}}(y, A)$. But then $|\mathsf{Sol}^{\mathsf{UL}}(y, A)| \geq 2$ and so by Proposition 6.4 we have $(y, A) \in \Sigma_{\mathrm{UL}}$. □

We can now prove Proposition 7.3.

Proof of Proposition 7.3. All parts are trivial if $\mathsf{stsp}(y, A) = 0$. We then assume that $\mathsf{stsp}(y, A) > 0$. Proposition 6.4 then shows that there exists a unique point x in $\mathsf{Sol}^{\mathsf{UL}}(y, A)$. Let $S := \mathsf{supp}(x)$. We now prove each of the three parts separately. Note that, because x is the only point in $\mathsf{Sol}^{\mathsf{UL}}(y, A)$, we don't need to take infima in the definition of σ_1, σ_2 and σ_3. Furthermore, if $S = \varnothing$ then parts (2) and (3) are trivial since $\sigma_2(y, A) = \sigma_3(y, A) = \infty$, so we assume $S \neq \varnothing$ for parts (2) and (3).

Proof of part (1): Note that the condition that $\sigma_1(y, A) < \lambda/4$ implies that A is not the zero matrix and so $\|A\|_{\max} \neq 0$. It also implies that $S^{\mathsf{c}} \neq \varnothing$ since otherwise $\sigma_1(y, A) = \lambda/2$. We will argue by contradiction and assume that the statement does not hold. Let t be a real number such that

$$\sigma_1(y, A) < t < \min\left\{\frac{\lambda\,\mathsf{stsp}(y, A)}{4\|A\|_{\max}}, \frac{\lambda}{4}\right\}$$

(note that this interval is non-empty by our assumption). Then there is an $i \in \{1, 2, \ldots, N\}$ such that $|[A_i^*(Ax - y)]| > \lambda/2 - t$ and $x_i = 0$. Set $\tilde{A} = A(\mathrm{I}_N + \delta P_i)$ where $\delta = 2t/(\lambda - 2t) < 4t/\lambda$ and where P_i is an $N \times N$ matrix consisting only of 0s except the entry on the ith row, ith column, which is set to 1. Clearly $\|A - \tilde{A}\|_{\max} \leq \delta\|A\|_{\max} \leq 4t\|A\|_{\max}/\lambda < \mathsf{stsp}(y, A)$. Thus $\mathsf{supp}(\tilde{x}) = \mathsf{supp}(x)$ for all $\tilde{x} \in \mathsf{Sol}^{\mathsf{UL}}(y, \tilde{A})$.

We claim that $x \in \mathsf{Sol}^{\mathsf{UL}}(y, \tilde{A})$. Indeed, since $x_i = 0$ and $\mathsf{supp}(\tilde{x}) = \mathsf{supp}(x)$, we must have $\tilde{x}_i = 0$. Thus, again as $x_i = 0$,

$$\|\tilde{A}x - y\|_2^2 + \lambda\|x\|_1 = \|Ax - y\|_2^2 + \lambda\|x\|_1 \leq \|A\tilde{x} - y\|_2^2 + \lambda\|\tilde{x}\|_1 = \|\tilde{A}\tilde{x} - y\|_2^2 + \lambda\|\tilde{x}\|_1,$$

the inequality by the optimality of x and the last equality as $\tilde{x}_i = 0$. This shows the claim.

By the assumption that $|[A^*(Ax - y)]_i| > \lambda/2 - t$ we have

$$|\tilde{A}_i^*(\tilde{A}x - y)| = (1 + \delta)|A_i^*(\tilde{A}x - y)| = (1 + \delta)|A_i^*(Ax - y)| > \left(1 + \frac{2t}{\lambda - 2t}\right)\left(\frac{\lambda}{2} - t\right) = \lambda/2$$

but this contradicts the fact that $\|\tilde{A}^*(\tilde{A}x - y)\|_\infty \leq \lambda/2$ by the KKT conditions for the LASSO problem and the fact that $x \in \mathsf{Sol}^{\mathsf{UL}}(y, \tilde{A})$.

Proof of part (2): Suppose for the sake of contradiction that for some $\epsilon > 0$ we have $\sigma_2(y, A) = t$ but $\mathsf{stsp}(y, A) > \sqrt{t + \epsilon}$. Then $\|(A_S^* A_S)^{-1}\|_2 = \frac{1}{t} > 1/(t + \epsilon)$. This implies that there is a $v \in \mathbb{R}^N$ such that $v_{S^{\mathsf{c}}} = 0$, $\|v\|_2 = 1$ and $\|A_S^* A_S v_S\|_2 \leq t + \epsilon$. Therefore, $\|A_S v_S\|_2^2 = \langle A_S^* A_S v_S, v_S\rangle \leq \|v_S\|_2 \|A_S^* A_S v_S\|_2$ and we obtain $\|A_S v_S\|_2 \leq \sqrt{t + \epsilon}$.

Let $B := A - Avv^{\mathrm{T}}$. Then

$$\|B - A\|_{\max} \leq \|B - A\|_2 = \|Avv^{\mathrm{T}}\|_2 \leq \|Av\|_2\,\|v^{\mathrm{T}}\|_2 = \|Av\|_2 \leq \sqrt{t + \epsilon}.$$

Hence, since $\mathsf{stsp}(y, A) > \sqrt{t + \epsilon}$, $(y, B) \notin \Sigma_{\mathrm{UL}}$ and there exists $\hat{x} \in \mathsf{Sol}^{\mathsf{UL}}(y, B)$ with $\mathsf{supp}(\hat{x}) = S$ (this comes from the definition of stsp). But

$$B_S v_S = A_S v_S - Avv_S^{\mathrm{T}} v_S = A_S v_S - Av\|v_S\|^2 = A_S v_S - Av = 0$$

and hence, $B_S^* B_S v_S = 0$. It follows that $B_S^* B_S$ is not invertible. This contradicts Lemma 8.3.

Proof of part (3): For shorthand, let $t := \sigma_3(y, A)$. Then there exists an index $i \in \{1, 2, \ldots, N\}$ such that $|x_i| = t$. Let $\tilde{y} = y - x_i A e_i$ where e_i is the ith vector in the standard basis on $\mathbb{R}^N$. We claim that $\tilde{x} = x - x_i e_i$ is such that $\tilde{x} \in \mathsf{Sol}^{\mathsf{UL}}(\tilde{y}, A)$. Since the KKT conditions are both necessary and sufficient for unconstrained LASSO, it suffices to show that $\tilde{x}$ obeys the KKT conditions. Let W be the support of $\tilde{x}$. Since $\tilde{x}_W = x_W$, we have $A\tilde{x} = A_W \tilde{x}_W = A_W x_W$ so that

$$A^*(A\tilde{x} - \tilde{y}) = A^*(A_W x_W - \tilde{y}) = A^*(A_W x_W + x_i A e_i - y) = A^*(Ax - y).$$

Therefore $A_W^*(A\tilde{x} - \tilde{y}) = -\lambda\mathfrak{s}(x)_W/2 = -\lambda\mathfrak{s}(\tilde{x})_W/2$ (since x obeys the KKT conditions) and on W^{c} we have $\|A_{W^{\mathsf{c}}}^*(A\tilde{x} - \tilde{y})\|_\infty = \|A_{W^{\mathsf{c}}}^*(Ax - y)\|_\infty \leq \lambda/2$ (again, since x obeys the KKT conditions). Thus $\tilde{x}$ obeys

the KKT conditions and so $\tilde{x} \in \mathsf{Sol}^{\mathsf{UL}}(\tilde{y}, A)$ as claimed. But then $\mathsf{supp}(\tilde{x}) \neq \mathsf{supp}(x)$ and so $\mathsf{stsp}(y, A) \leq d_{\max}([(y, A), (\tilde{y}, A)]) \leq \|x_i A e_i\|_\infty \leq t\|A\|_{\max}$. □

8.4. **Proof of Proposition 7.4.** We assume that $\sigma > 0$, otherwise there is nothing to prove. Let $x \in \mathsf{Sol}^{\mathsf{UL}}(y, A)$ and $S = \mathsf{supp}(x)$. Additionally, let $\Delta := \min\left(\frac{\sigma^2}{q(\alpha,\sigma)}, \frac{\sqrt{\sigma}}{6\alpha}, \alpha\right)$ and $\delta := (mN)^{-1/2}\Delta$. Finally, let $(\tilde{y}, B)$ be such that $\|\tilde{y} - y\|_\infty \leq \delta$ and $\|A - B\|_{\max} \leq \delta$.

We will use the following results derived from the classic bounds (see [16, §1.1] or [39, §6.2]) between norms and the definition of Δ:

$$\|B - A\|_2 \leq \sqrt{mN}\|B - A\|_{\max} \leq \sqrt{mN}\delta \leq \Delta, \tag{8.2}$$

$$\|B\|_2 \leq \|A\|_2 + \Delta \leq 2\alpha, \tag{8.3}$$

$$\|\tilde{y} - y\|_2 \leq \sqrt{m}\delta \leq \Delta, \tag{8.4}$$

$$\|\tilde{y}\|_2 \leq \|y\|_2 + \|\tilde{y} - y\|_2 \leq \alpha + \Delta \leq 2\alpha. \tag{8.5}$$

We start with the case where both S and S^{c} are non empty. We initially establish a sequence of basic inequalities. We begin by observing that

$$B_S^* B_S = A_S^* A_S + (B_S - A_S)^* A_S + A_S^*(B_S - A_S) + (B_S - A_S)^*(B_S - A_S)$$

so that $B_S^* B_S = A_S^* A_S(\mathrm{I} + X)$ where

$$X = (A_S^* A_S)^{-1}\left[(B_S - A_S)^* A_S + A_S^*(B_S - A_S) + (B_S - A_S)^*(B_S - A_S)\right].$$

Note that X is well defined since $(A_S^* A_S)^{-1}$ exists by the assumption that $\sigma > 0$ and Proposition 7.2. In addition,

$$\begin{aligned}
\|X\|_2 &\leq \|(A_S^* A_S)^{-1}\|_2 \left(\|(B_S - A_S)^* A_S\|_2 + \|A_S^*(B_S - A_S)\|_2\right. \\
&\quad \left. + \|(B_S - A_S)^*(B_S - A_S)\|_2\right) \\
&\leq \|(A_S^* A_S)^{-1}\|_2 \left(2\|B_S - A_S\|_2 \|A_S\|_2 + \|B_S - A_S\|_2^2\right) \\
&\leq \frac{2\Delta\|A\|_2 + \Delta^2}{\sigma_2(y, A)} \leq \frac{3\Delta\alpha}{\sqrt{\sigma}}
\end{aligned} \tag{8.6}$$

where we used the definition of σ_2, (8.2) and $\Delta, \|A\|_2 \leq \alpha$. It follows from the hypothesis on Δ that $\|X\|_2 < 1/2$. Hence, $\mathrm{I} + X$ is invertible with inverse satisfying $(\mathrm{I} + X)^{-1} = \sum_{r=0}^{\infty}(-1)^r X^r$ and consequently, so is $B_S^* B_S$ and we have $(B_S^* B_S)^{-1} = (\mathrm{I} + X)^{-1}(A_S^* A_S)^{-1}$. Furthermore,

$$\begin{aligned}
\|(B_S^* B_S)^{-1} - (A_S^* A_S)^{-1}\|_2 &= \|(\mathrm{I} + X)^{-1}(A_S^* A_S)^{-1} - (A_S^* A_S)^{-1}\|_2 \\
&\leq \|(\mathrm{I} + X)^{-1} - \mathrm{I}\|_2 \, \|(A_S^* A_S)^{-1}\|_2 \\
&\leq \left\|\sum_{r=1}^{\infty}(-1)^r X^r\right\|_2 \|(A_S^* A_S)^{-1}\|_2 \leq \frac{\|X\|_2 \|(A_S^* A_S)^{-1}\|_2}{1 - \|X\|_2} \\
&< \frac{2\|X\|_2}{\sigma_2(y, A)} \leq \frac{6\Delta\alpha}{\sigma},
\end{aligned} \tag{8.7}$$

where the last inequality following from (8.6). In particular,

$$\begin{aligned}
&\left\|(B_S^* B_S)^{-1} B_S^* \tilde{y} - (A_S^* A_S)^{-1} A_S^* y\right\|_2 \\
&\leq \left\|\left[(B_S^* B_S)^{-1} B_S^* - (A_S^* A_S)^{-1} B_S^*\right]\tilde{y}\right\|_2 + \left\|\left[(A_S^* A_S)^{-1} B_S^* - (A_S^* A_S)^{-1} A_S^*\right]\tilde{y}\right\|_2 && (8.8) \\
&+\left\|(A_S^* A_S)^{-1} A_S^*(y - \tilde{y})\right\|_2 \\
&\leq \|(B_S^* B_S)^{-1} - (A_S^* A_S)^{-1}\|_2 \|B_S^*\|_2 \|\tilde{y}\|_2 && (8.9) \\
&+\|(A_S^* A_S)^{-1}\|_2\left(\|B_S^* - A_S^*\|_2\|\tilde{y}\|_2 + \|A_S^*\|_2\|y - \tilde{y}\|_2\right) \\
&\leq \frac{6\Delta\alpha(2\alpha)(2\alpha)}{\sigma} + \frac{\Delta(2\alpha) + \Delta\alpha}{\sigma_2(y, A)} \qquad \text{by (8.2), (8.7), (8.4), and (8.5)} \\
&= \frac{24\Delta\alpha^3}{\sigma} + \frac{3\Delta\alpha}{\sqrt{\sigma}} =: \Upsilon. && (8.10)
\end{aligned}$$

Since $\sigma > 0$ and x obeys the KKT conditions for unconstrained LASSO with input (y, A), we see that $x_S = (A_S^* A_S)^{-1}(A_S^* y - \lambda \mathfrak{s}(x)_S/2)$ and $x_{S^\mathsf{c}} = 0$. Hence, if we let $\tilde{x} \in \mathbb{R}^N$ be given by $\tilde{x}_S := (B_S^* B_S)^{-1}(B_S^* \tilde{y} - \lambda\, \mathfrak{s}(x)_S/2)$ and $\tilde{x}_{S^\mathsf{c}} := 0$ then

$$\begin{aligned}
2\|\tilde{x} - x\|_2 &\leq \|(B_S^* B_S)^{-1}(2B_S^*\tilde{y} - \lambda\, \mathfrak{s}(x)_S) - (A_S^* A_S)^{-1}(2A_S^* y - \lambda\, \mathfrak{s}(x)_S)\|_2 \\
&\leq 2\left\|(B_S^* B_S)^{-1} B_S^* \tilde{y} - (A_S^* A_S)^{-1} A_S^* y\right\|_2 \qquad (8.11)\\
&\quad + \lambda \left\|\left[(B_S^* B_S)^{-1} - (A_S^* A_S)^{-1}\right] \mathfrak{s}(x)_S\right\|_2 \\
&\underset{(8.8)}{\leq} 2\Upsilon + \lambda \|(B_S^* B_S)^{-1} - (A_S^* A_S)^{-1}\|_2 \|\mathfrak{s}(x)_S\|_2 \underset{(8.7)}{\leq} 2\Upsilon + \frac{6\Delta\alpha\lambda\sqrt{|S|}}{\sigma} \\
&\leq 6\Delta\alpha\left(\frac{8\alpha^2}{\sigma} + \frac{1 + \lambda\sqrt{|S|}}{\sqrt{\sigma}}\right) \qquad (8.12)
\end{aligned}$$

Also, since $x \in \mathsf{Sol}^{\mathsf{UL}}(y, A)$ we have (by (UL1)) $\|Ax - y\|_2^2, \lambda\|x\|_1 \leq \|y\|_2^2$. Thus

$$\begin{aligned}
&\|B_{S^\mathsf{c}}^*(B\tilde{x} - \tilde{y}) - A_{S^\mathsf{c}}^*(Ax - y)\|_2 \\
&\leq \|B_{S^\mathsf{c}}^*(B\tilde{x} - \tilde{y}) - B_{S^\mathsf{c}}^*(Bx - \tilde{y})\|_2 + \|B_{S^\mathsf{c}}^*(Bx - \tilde{y}) - B_{S^\mathsf{c}}^*(Ax - \tilde{y})\|_2 \\
&\quad + \|B_{S^\mathsf{c}}^*(Ax - \tilde{y}) - B_{S^\mathsf{c}}^*(Ax - y)\|_2 + \|B_{S^\mathsf{c}}^*(Ax - y) - A_{S^\mathsf{c}}^*(Ax - y)\|_2 \\
&\leq \|B\|_2\left(\|B\|_2\|\tilde{x} - x\|_2 + \|B - A\|_2\|x\|_2 + \|y - \tilde{y}\|_2\right) + \|B - A\|_2\|Ax - y\|_2 \\
&\leq 2\alpha\left[2\alpha\|(\tilde{x} - x)\|_2 + \frac{\|B - A\|_2\|y\|_2^2}{\lambda} + \|y - \tilde{y}\|_2\right] + \Delta\|y\|_2 \\
&\leq 2\alpha\left[6\Delta\alpha^2\left(\frac{8\alpha^2}{\sigma} + \frac{1 + \lambda\sqrt{|S|}}{\sqrt{\sigma}}\right) + \frac{\Delta\alpha^2}{\lambda} + \Delta\right] + \Delta\alpha \\
&= \quad 12\Delta\alpha^3\left(\frac{8\alpha^2}{\sigma} + \frac{1 + \lambda\sqrt{|S|}}{\sqrt{\sigma}}\right) + \frac{2\Delta\alpha^3}{\lambda} + 3\Delta\alpha \\
&\leq \Delta\left(\frac{96\alpha^5}{\sigma} + \frac{12\alpha^3 + 12\alpha^3\lambda\sqrt{|S|}}{\sqrt{\sigma}} + \frac{2\alpha^3}{\lambda} + 3\alpha\right) \leq \Delta q(\alpha, \sigma)/\sigma. \qquad (8.13)
\end{aligned}$$

We can now conclude the proof. Using (8.11) and the definition of Δ we obtain that

$$\begin{aligned}
\|\tilde{x} - x\|_\infty \quad &\leq \quad \|\tilde{x} - x\|_2 \;\leq\; 3\Delta\alpha\left(\frac{8\alpha^2}{\sigma} + \frac{1 + \lambda\sqrt{|S|}}{\sqrt{\sigma}}\right) \;\leq\; 3\frac{\sigma^2}{q(\alpha, \sigma)}\alpha\left(\frac{8\alpha^2}{\sigma} + \frac{1 + \lambda\sqrt{|S|}}{\sqrt{\sigma}}\right) \\
&= \quad \sigma\left(\frac{24\alpha^3 + 3\alpha\sqrt{\sigma}(1 + \lambda\sqrt{|S|})}{96\alpha^5 + 12\alpha^3(1 + \lambda\sqrt{N})\sqrt{\sigma} + \left(\frac{2\alpha^3}{\lambda} + 3\alpha\right)}\right) \;<\; \sigma
\end{aligned}$$

and therefore, if we set $S^+ = \{i \,|\, x_i > 0\}$, then for each $i \in S^+$ we have that $\tilde{x}_i \geq x_i - |\tilde{x}_i - x_i| > \sigma_3(y, A) - \sigma \geq 0$. Similarly, if we set $S^- = \{i \,|\, x_i < 0\}$ then for each $i \in S^-$ we have $\tilde{x}_i < 0$. It follows that each entry of $\tilde{x}_S$ is non-zero and that $\mathfrak{s}(\tilde{x})_S = \mathfrak{s}(x)_S$ and so,

$$2B_S^*(B\tilde{x} - \tilde{y}) = \lambda\, \mathfrak{s}(x)_S = \lambda\, \mathfrak{s}(\tilde{x})_S. \qquad (8.14)$$

Using (8.13) and the definition of Δ we show as above that $\|B_{S^\mathsf{c}}^*(B\tilde{x} - \tilde{y}) - A_{S^\mathsf{c}}^*(Ax - y)\|_2 < \sigma$ and thus

$$\begin{aligned}
\|B_{S^\mathsf{c}}^*(B\tilde{x} - \tilde{y})\|_\infty &\leq \|B_{S^\mathsf{c}}^*(B\tilde{x} - \tilde{y}) - A_{S^\mathsf{c}}^*(Ax - y)\|_\infty + \|A_{S^\mathsf{c}}^*(Ax - y)\|_\infty \\
&< \sigma + \lambda/2 - \sigma_1(y, A) \leq \lambda/2. \qquad (8.15)
\end{aligned}$$

Inequalities (8.14) and (8.15), together with the fact that each entry of $\tilde{x}_S$ is non-zero show that $\tilde{x}$ satisfies the unconstrained LASSO KKT conditions for $(\tilde{y}, B)$ and that the hypothesis of Proposition 7.2 part (1) holds. We therefore have $\{\tilde{x}\} = \mathsf{Sol}^{\mathsf{UL}}(\tilde{y}, B)$. Since each entry of $\tilde{x}_S$ is non-zero and $\tilde{x}_{S^\mathsf{c}} = 0$, we have $\mathsf{supp}(\tilde{x}) = \mathsf{supp}(x)$ and therefore all vectors in $\mathsf{Sol}^{\mathsf{UL}}(\tilde{y}, B)$ have the same support as x.

We now argue that the same result holds for $S = \varnothing$ or $S^\mathsf{c} = \varnothing$. In the former case, we set $\tilde{x} = 0$. Then the bound in (8.11) holds trivially and (8.13) and (8.15) follow as before. Thus $\tilde{x}$ satisfies the unconstrained LASSO

KKT conditions for $(\tilde{y}, B)$ and we are done. In the later case where $S^c = \varnothing$, the only difference is that there is no need to compute (8.13): it suffices to use (8.11) to conclude (8.14), which shows that $\tilde{x}$ satisfies the unconstrained LASSO KKT conditions.

We have thus shown in each case for S that all vectors in $\mathsf{Sol}^{\mathsf{UL}}(\tilde{y}, B)$ have the same support as x, for every pair $(\tilde{y}, B)$ with $\|\tilde{y} - y\|_\infty \leq \delta$ and $\|A - B\|_{\max} \leq \delta$. We deduce that $\mathsf{stsp}(y, A) \geq \delta$ which completes the proof.

8.5. **A convex quadratic routine for unconstrained LASSO.** In addition to the subroutines $\mathsf{GetMatrix}^U$ and $\mathsf{GetVector}^b$, which are assumed within our computational model we will use the following subroutine:

$\mathsf{ULasso}(y, A, \lambda)$**::** given $y \in \mathbb{Q}^m$, $A \in \mathbb{Q}^{m \times N}$, and $\lambda \in \mathbb{Q}$, it returns a vector $x \in \mathsf{Sol}^{\mathsf{UL}}(y, A)$ in time $O(N^3 \log_2(L))$ where L is the total number of bits of A, y and λ.

The existence of ULasso follows from the fact that unconstrained LASSO can be written as a convex quadratic program. To see this, if we write $x = x^+ - x^-$ and $\tilde{x} = \begin{pmatrix} x^+ & x^- \end{pmatrix}^{\mathrm{T}}$ with $x^+ \geq 0$ and $x^- \geq 0$ (where these inequalities are taken entrywise) we have

$$\begin{aligned} \|Ax - y\|_2^2 + \lambda \|x\|_1 &= \langle Ax, Ax \rangle - 2\langle A^* y, x \rangle + \lambda \sum_{i=1}^{N} (x_i^+ + x_i^-) + \|y\|_2^2 \\ &= \tilde{x}^* M \tilde{x} + (\lambda \mathbf{1}_{2N} - 2By)^* \tilde{x} + \|y\|_2^2 \end{aligned}$$

where $\mathbf{1}_N$ is a vector of length N with each entry equal to 1, $M = \begin{pmatrix} A^*A & -A^*A \\ -A^*A & A^*A \end{pmatrix}$ and $B = \begin{pmatrix} A^* \\ -A^* \end{pmatrix}$. Note also that M is positive semi-definite. Since $\|y\|_2^2$ is constant, we conclude that the solutions to the quadratic program in standard form

$$\operatorname*{argmin}_{\tilde{x} \in \mathbb{R}^N} \{ \tilde{x}^* M \tilde{x} + (\lambda \mathbf{1}_{2N} - 2By)^* \tilde{x} \mid -\mathrm{I}_{2N} \tilde{x} \leq 0 \}$$

can be converted to solutions of the unconstrained LASSO problem with inputs A and y. Thus using the algorithm proposed in e.g. [45], we obtain an algorithm that works in $\mathcal{O}(N^3 L) = \mathcal{O}(N^3 p + N^3 \log_2(m+N)) = \mathcal{O}(N^3(p + \log_2(N)))$ arithmetic operations, where L is the total number of bits required to store A, y and λ and p is the maximal number of bits required to store any entry of A, y and λ.

8.6. **A subroutine for testing upper bounds for σ.** Our aim in this section is to produce an algorithm that can tell us, for a given value of C, whether or not $\sigma \leq C^2$. More precisely, we aim to produce the following subroutine to add to the one discussed above,

$\mathsf{Sigma}(y, A, x, \lambda, C)$**::** with input $y \in \mathbb{Q}^m$, $A \in \mathbb{Q}^{m \times N}$, $x \in \mathbb{Q}^N$, and $\lambda, C \in \mathbb{Q}$, returns true if $\sigma(y, A) \leq C^2$ and false otherwise. The precondition is that $x \in \mathsf{Sol}^{\mathsf{UL}}(y, A)$ and λ is the unconstrained LASSO parameter. The cost of running this procedure is $\mathcal{O}(N^3)$.

Executing $\mathsf{Sigma}(y, A, x, \lambda, C)$ requires deciding if $\sigma_1(y, A) \leq C^2$, $\sigma_2(y, A) \leq C$ and $\sigma_3(y, A) \leq C^2$. Let $S := \mathsf{supp}(x)$. Note that computing σ_1 can be done by computing $A_{S^{\mathsf{c}}}^*(Ax - y)$ (at a cost of $\mathcal{O}(mN)$ operations), finding the maximum absolute value across all rows (taking $\mathcal{O}(|S^{\mathsf{c}}|) = \mathcal{O}(N)$ operations) and then subtracting $\lambda/2$. Hence σ_1 can be computed in $\mathcal{O}(mN)$ operations. We compute σ_3 via a simple maximum argument requiring $\mathcal{O}(N)$ operations. The hardest of the three to compute is σ_2; we will approximate the smallest singular value of $A_S^* A_S$ by testing the positive definiteness of $A_S^* A_S - t\mathrm{I}$ for various values of t. Note that if $A_S^* A_S - t\mathrm{I}$ is not positive definite then $\|(A_S^* A_S)^{-1}\|_2^{-1} \leq t$. Conversely, if $A_S^* A_S - t\mathrm{I}$ is postiive definite then $\|(A_S^* A_S)^{-1}\|_2^{-1} > t$. Indeed, the following chain of equivalences hold since $A_S^* A_S$ is symmetric: $A_S^* A_S - t\mathrm{I}$ is positive definite $\iff$ each eigenvalue of $A_S^* A_S$ is larger than t $\iff$ each eigenvalue of $(A_S^* A_S)^{-1}$ is smaller than t^{-1} $\iff$ $\|(A_S^* A_S)^{-1}\|_2 < t^{-1}$.

Remark 8.4. It is well known that there is an algorithm PosDef, that can check if a symmetric $r \times r$ matrix is positive definite or not. Such an algorithm runs with $\mathcal{O}(r^3)$ operations.

We can now precisely state our algorithm for computing σ (where we take by convention $\|A^*_\varnothing(Ax-y)\|_\infty = 0$ and $A^*_\varnothing A_\varnothing - C\mathrm{I}$ as positive definite for any $C > 0$).

Algorithm: $\mathsf{Sigma}(y, A, x, \lambda, C)$
Data: $A \in \mathbb{R}^{m\times N}$, $y \in \mathbb{R}^m$, $x \in \mathsf{Sol}^{\mathsf{UL}}(y, A) \subseteq \mathbb{R}^N$, $C > 0$, and the LASSO parameter λ.
Result: true if $\sigma \le C^2$, otherwise, false
$S := \mathsf{supp}(x)$;
if $\|A^*_{S^c}(Ax - y)\|_\infty < \lambda/2$ **and** $A^*_S A_S$ is invertible **then**
  $\sigma_1 := \lambda/2 - \max_{i\in S^c}(|A^*_i(Ax - y)|)$;
  $\sigma_3 := \|x_{S^c}\|_\infty$;
  $X := A^*_S A_S - C\mathrm{I}$;
  if $\mathsf{PosDef}(X)$ **and** $\sigma_1 \le C^2$ **and** $\sigma_3 \le C^2$ **then**
    return true;
  else
    return false;
  end
else
  return true
end

Proposition 8.5. *Algorithm* Sigma *is correct (the returned value is* true *iff* $\sigma(y, A) \le C^2$*). Its running time is bounded by* $\mathcal{O}(N^3)$.

Proof. We begin proving correctness. Because $x \in \mathsf{Sol}^{\mathsf{UL}}(y, A)$, by Proposition 7.2(1), if $A^*_{S^c}(Ax - y) < \lambda/2$ and $A^*_S A_S$ is invertible, then x is the only solution of (y, A). In this case the infima in the definition of σ_1, σ_2 and σ_3 reduces to the corresponding values at S and the correctness of the estimates for the σs has already been argued above. If, instead, either $A^*_{S^c}(Ax - y) \ge \lambda/2$ or $A^*_S A_S$ is invertible, then, by Proposition 7.2(2), $\sigma = 0$ and, clearly, $\sigma \le C^2$.

We now show the complexity bound. As mentioned earlier, computing σ_1 takes $\mathcal{O}(mN)$ operations and computing σ_3 takes $\mathcal{O}(N)$ operations. Computing X can be done using $\mathcal{O}(|S|^2 N) = \mathcal{O}(N^3)$ operations. Also, as discussed earlier, the computation of PosDef takes $|S|^3/3 = \mathcal{O}(N^3)$ operations. Thus the total complexity of the algorithm is $\mathcal{O}(N^3 + mN + N + N^3) = \mathcal{O}(N^3)$. □

8.7. **Proof of Theorem 2.1, parts (1) & (2).** Our argument for both parts (1) and (2) will involve increasing the precision of the approximations (y, A) of the true input, which we denote throughout the proof by (b, U), until $\sigma(y, A)$ is sufficiently large. This strategy will give us both the solution to unconstrained LASSO feature selection (part 1) and an upper bound for the condition number (part 2). This is done in the algorithm FSUL.

There are two things that need to be shown: firstly, that the algorithm is correct in the sense that if the **repeat** loop terminates, the output S is correct and η is a bound for $\mathscr{C}_{\mathrm{UL}}(b, U)$. Secondly, that the runtime is bounded as stated, which will also give us a guarantee that the **repeat** loop does in fact terminate when $\mathscr{C}_{\mathrm{UL}}(b, U) < \infty$. The correctness proof will require the use of Proposition 7.4 and the runtime bound the use of Proposition 7.3. We split the proof of the runtime into two further sections: firstly, we evaluate the cost of each iteration of the loop, and secondly we show a bound on the number of iterations that are executed.

8.7.1. *Proof of correctness:* Assume that the repeat loop terminates. We use the notation $S, n, \delta, A, y, x, G, H$ and C to denote the values of these variables set by the algorithm when the **repeat** loop has terminated. We claim that $\mathsf{stsp}(y, A) \ge 2\delta$. Assume for now that this is the case. Then by the properties (4.1) of $\mathsf{GetMatrix}^U$ and $\mathsf{GetVector}^b$, at the nth iteration of the repeat loop we must have $d_\infty(b, y), d_{\max}(A, U) \le 2^{-4n} = 16^{-n} = \delta$.

Thus when the **repeat** loop terminates, $\mathsf{stsp}(y, A) \ge 2\delta > d_{\max}[(y, A), (b, U)]$ and so if $w \in \mathsf{Sol}^{\mathsf{UL}}(b, U)$ and $v \in \mathsf{Sol}^{\mathsf{UL}}(y, A)$ then $\mathsf{supp}(w) = \mathsf{supp}(v)$. But since $x \in \mathsf{Sol}^{\mathsf{UL}}(y, A)$ by the correctness of ULasso, we must have

Algorithm: FSUL
Data: $\lambda > 0$ as well as oracles $\mathsf{GetMatrix}^U$ and $\mathsf{GetVector}^b$ that approximate (b, U) to arbitrary precision
Result: a support set $S \subseteq \{1, \dots, N\}$ and η such that $\mathsf{Cond}(b, U) \le \eta$
$\delta := 1, \quad n := 0, \quad \delta^{1/4} := 1;$
repeat
  $n := n + 1;$
  $\delta := \delta/16;$
  $A := \mathsf{GetMatrix}^U(4n);$
  $y := \mathsf{GetVector}^b(4n);$
  $x := \mathsf{ULasso}(y, A, \lambda);$
  $\delta^{1/4} := \delta^{1/4}/2;$
  $G := [\![(y, A)]\!]_{\max};$
  $H := [\![(y, A)]\!]_{\mathrm{S}};$
  $C := 6 \cdot \delta^{1/4} N(\lambda + \lambda^{-1})H^2;$
until not $\mathsf{Sigma}(y, A, x, \lambda, C)$ **and** $G^2(mN)^{-1} \ge 4\delta^2$;
return $S := \mathsf{supp}(x)$ and $\eta = \delta^{-1}$.

$\mathsf{supp}(w) = \mathsf{supp}(x) = S$. Furthermore, since $\mathsf{stsp}(b, U) \ge \mathsf{stsp}(y, A) - d_{\max}[(y, A), (b, U)] \ge 2\delta - \delta = \delta$, we must have $\mathscr{C}_{\mathrm{UL}}(b, U) \le \delta^{-1} = \eta$.

Therefore we have shown that if the algorithm terminates then it terminates with the correct result, provided that we can show that $\mathsf{stsp}(y, A) \ge 2\delta$. To that end, we will make use of the following lemma.

Lemma 8.6. *Let $H \ge \nu \ge 1$, $\delta \le 1$, and $\lambda > 0$. Let $C = 6\delta^{1/4}N(\lambda + \lambda^{-1})H^2$. For all $\xi \ge C^2$ we have*

$$f(\xi) := \xi^2 - 2\delta\sqrt{mN}q(\nu, \xi) \ge 0 \quad \textit{and} \quad g(\xi) := \frac{\sqrt{\xi}(mN)^{-1/2}}{6\nu} - 2\delta \ge 0$$

where, we recall, $q(\nu, \xi)$ is defined in (7.1).

Assuming for now that Lemma 8.6 holds, we will complete the proof that $\mathsf{stsp}(y, A) \ge 2\delta$. Indeed, when the algorithm terminates $\sigma(y, A) > C^2$ because $\mathsf{Sigma}(y, A, x, \lambda, C)$ does not hold. Now we apply Lemma 8.6 with $H = [\![(y, A)]\!]_{\mathrm{S}}$, $\alpha = [\![(y, A)]\!]_2$ and $\sigma = \sigma(y, A)$. Using the inequality for f in this lemma with $\xi = \sigma$ and $\nu = \alpha$ we obtain $\sigma^2 \ge 2\delta\sqrt{mN}q(\alpha, \sigma)$ and thus $(mN)^{-1/2}\sigma^2/q(\alpha, \sigma) \ge 2\delta$. Next, using the inequality for g, we obtain $(mN)^{-1/2}\sqrt{\sigma}/(6\alpha) \ge 2\delta$. Finally, when the **repeat** loop terminates we must have $G^2(mN)^{-1} \ge 4\delta^2$ and in particular, since $\alpha \ge G$, we get $(mN)^{-1/2}\alpha \ge 2\delta$. Combining these inequalities yields

$$\mathsf{stsp}(y, A) \ge (mN)^{-1/2} \min(\sigma^2/q(\alpha, \sigma), \sqrt{\sigma}/(6\alpha), \alpha) \ge 2\delta \tag{8.16}$$

where the first inequality follows from Proposition 7.4.

All that remains is to prove Lemma 8.6. This is done as follows.

Proof of Lemma 8.6. We make frequent use of the inequality $H \ge \nu$. We start by proving that $f(\xi)$ is increasing on $[C^2, \infty)$. We have $f'(\xi) = 2\xi - 2\delta\sqrt{mN}[6\nu^3(1 + \lambda\sqrt{N})/\sqrt{\xi} + 2\nu^3/\lambda + 3]$. The form of $f'(\xi)$ makes it clear to see that for positive ξ, $f'(\xi)$ is increasing and so to show that $f(\xi)$ is increasing on $[C^2, \infty)$ it suffices to show that $f'(C^2) \ge 0$. Because $\delta \le 1$, we have $\delta^{2/4} \ge \delta^{3/4}$. Furthermore, $H \ge \nu \ge 1$ and $N \ge 1$. Thus

$$\delta^{2/4}N^2(\lambda^2 + 2 + \lambda^{-2})H^4 \ge 2\delta^{2/4}N^2H \ge \delta^{2/4}\left(1 + \frac{\lambda\sqrt{N}}{\lambda + \lambda^{-1}}\right)\nu \ge \frac{\nu\delta^{3/4}(1 + \lambda\sqrt{N})}{\lambda + \lambda^{-1}} \tag{8.17}$$

Furthermore, $(\lambda^2 + 2 + \lambda^{-2})\lambda > 2\lambda + \lambda^{-1} > \lambda + \lambda^{-1} \ge 2$ by the AM-GM inequality and the assumption that $\lambda > 0$. Thus

$$\delta^{2/4}N^2(\lambda^2 + 2 + \lambda^{-2})H^4 \ge 2\delta^{2/4}N^2\nu^3\lambda^{-1} \ge 2\nu^3\delta N\lambda^{-1}. \tag{8.18}$$

Our final simple inequality is the following: using $\lambda^2 + \lambda^{-2} \geq 2$ and $H, N \geq 1 \geq \delta$, we get

$$\delta^{2/4} N^2 (\lambda^2 + 2 + \lambda^{-2}) H^4 \geq 4\delta^{2/4} N^2 \nu \geq 3\delta N \nu. \tag{8.19}$$

Using the definitions of f, C and the bounds $H \geq \nu$, $H, N \geq 1$, and $m \leq N$, we get

$$\begin{aligned}
f'(C^2) &= 2C^2 - \delta\sqrt{mN}\left[\frac{6\nu^3(1+\lambda\sqrt{N})}{C} + \frac{2\nu^3}{\lambda} + 3\nu\right] \\
&\geq (2\cdot 6^2)\delta^{2/4} N^2(\lambda^2+2+\lambda^{-2})H^4 - 2\delta N\left[\frac{6\nu^3(1+\lambda\sqrt{N})}{6\delta^{1/4}N(\lambda+\lambda^{-1})H^2} + \frac{2\nu^3}{\lambda} + 3\nu\right] \\
&\geq 6\delta^{2/4} N^2(\lambda^2+2+\lambda^{-2})H^4 - 2\left[\frac{\nu\delta^{3/4}(1+\lambda\sqrt{N})}{(\lambda+\lambda^{-1})} + \frac{2\nu^3\delta N}{\lambda} + 3\nu\delta N\right] \geq 0,
\end{aligned}$$

where the subtraction in the last line decomposes into three subtractions each of them being non-negative by (8.17–8.19). We conclude that f is increasing on $[C^2, \infty)$. Thus to show that f is positive it is sufficient to show that $f(C^2) > 0$. Indeed,

$$\begin{aligned}
\frac{f(C^2)}{(NH^2)^4\delta} &= (6(\lambda+\lambda^{-1}))^4 - \frac{2\sqrt{mN}}{(NH^2)^4}\Big[96\nu^5 + 12\nu^3(1+\lambda\sqrt{N})(6\delta^{1/4}N(\lambda+\lambda^{-1})H^2) \\
&\qquad + 6^2\delta^{2/4}N^2(\lambda+\lambda^{-1})^2H^4\left(\frac{2\nu^3}{\lambda}+3\nu\right)\Big] \\
&\geq 6^4(\lambda+\lambda^{-1})^4 - 2\Big(96 + 12(1+\lambda)(6(\lambda+\lambda^{-1})) + 6^2(\lambda+\lambda^{-1})^2\left(\frac{2}{\lambda}+3\right)\Big) \\
&= [6^4(\lambda+\lambda^{-1})^4 - 2(180\lambda^2 + 144\lambda + 384\lambda^0 + 216\lambda^{-1} + 108\lambda^{-2} + 72\lambda^{-3})] \\
&\geq 6^4[\lambda^4 + 4\lambda^2 + 6 + 4\lambda^{-2} + \lambda^{-4} - (\lambda^2 + \lambda + 1 + \lambda^{-1} + \lambda^{-2} + \lambda^{-3})] \\
&\geq 6^4[\lambda^4 + 4\lambda^2 + 6 + 4\lambda^{-2} + \lambda^{-4} - (2\lambda^2 + 3 + 3\lambda^{-2} + \lambda^{-4})] \geq 0.
\end{aligned}$$

where in the last line we have made use of the facts that $\lambda < \lambda^2 + 1$, $\lambda^{-1} < \lambda^{-2} + 1$ and $\lambda^{-3} < \lambda^{-4} + \lambda^{-2}$ all of which following from $\lambda > 0$. Since f is increasing on $[C^2, \infty)$ and $f(C^2) \geq 0$, we get that $f(\xi) \geq 0$ for $\xi \geq C^2$, completing the proof for f.

We finish the argument by showing that $g(\xi) > 0$. This is somewhat simpler: since $\xi \geq C^2$ by assumption, we have $\xi \geq 36\delta^{2/4}N^2(\lambda+\lambda^{-1})^2H^4 \geq 36 \cdot 4\delta^2(mN)\nu^2$ since $\delta \leq 1$, $H \geq \nu$, $N \geq m$ and $\lambda + \lambda^{-1} \geq 2$. Taking square roots yields $\sqrt{\xi} \geq 12\delta\nu(mN)^{1/2}$, or, equivalently, $\sqrt{\xi}(mN)^{-1/2}/(6\nu) \geq 2\delta$. □

8.7.2. *A bound on the number of iterations & precision that the repeat loop requires.* The number of iterations & the maximum precision of the **repeat** loop is bounded in the following lemma.

Lemma 8.7. *There exists a universal constant D independent of all parameters such that for any $n \in \mathbb{N}$ with*

$$n \geq D \log_2\left(\max\{\lambda + \lambda^{-1}, N, [\![(b,U)]\!]_{\max}, \mathscr{C}_{\mathrm{UL}}(b,U)\}\right) \tag{8.20}$$

and any (y, A) and (b, U) such that $d_{\max}[(y,A),(b,U)] \leq 2^{-n} =: \delta$, we have $[\![(y,A)]\!]_{\max}(mN)^{-1/2} \geq \delta$ and $\sigma(y,A) \geq C^2$ where $C = 6\delta^{1/4}N(\lambda+\lambda^{-1})[\![(y,A)]\!]_{\mathrm{S}}^2$.

As a consequence, the number of iterations of the **repeat** *loop in* FSUL *(and thus, the maximum digits of precision used by* FSUL*) on input (b, U) is bounded by*

$$\mathcal{O}(\lceil \log_2\left(\max\{\lambda + \lambda^{-1}, N, [\![(b,U)]\!]_{\mathrm{S}}, \mathscr{C}_{\mathrm{UL}}(b,U)\}\right)\rceil).$$

Proof. As we don't need to give a precise value for D in the bound on the number of iterations we will give a shorter proof without making this value explicit using big $\mathcal{O}$ and big Ω notation [42]. To show that $\sigma(y,A) \geq C^2$ we need to show that $\sigma_1(y,A) \geq C^2$, $\sigma_2(y,A) \geq C$ and $\sigma_3(y,A) \geq C^2$. We start by noting that using well-known norm inequalities [39, §6.2] and the fact that $m \leq N$ we get $[\![(y,A)]\!]_{\mathrm{S}} \leq N^2[\![(y,A)]\!]_{\max}$. Moreover, $[\![(y,A)]\!]_{\max} \leq [\![(b,U)]\!]_{\max} + \delta$ and since $\delta \leq 1 \leq [\![(y,A)]\!]_{\max}$ we obtain $[\![(y,A)]\!]_{\max} \leq 2[\![(b,U)]\!]_{\max}$. Hence

$$[\![(y,A)]\!]_{\mathrm{S}} \leq 2N^2[\![(b,U)]\!]_{\max}. \tag{8.21}$$

Since $\lambda + \lambda^{-1} \geq 2$, the bound (8.20) implies that $n \geq D\max\{2, \log_2 \mathscr{C}_{\mathrm{UL}}(b,U)\} \geq D\max\{1, \log_2 \mathscr{C}_{\mathrm{UL}}(b,U)\}$. Hence, since $\mathsf{stsp}(y,A) \geq \mathsf{stsp}(b,U) - \delta$, we obtain

$$\mathsf{stsp}(y,A) \geq \mathsf{stsp}(b,U) - \delta \geq \mathsf{stsp}(b,U) - 2^{-D\max\{1,\log_2 \mathscr{C}_{\mathrm{UL}}(b,U)\}}. \tag{8.22}$$

We now claim that for $D > 2$ we have

$$\mathscr{C}_{\mathrm{UL}}(y,A) \leq 2\mathscr{C}_{\mathrm{UL}}(b,U). \tag{8.23}$$

Indeed, if $\mathscr{C}_{\mathrm{UL}}(b,U) \geq 2$ we see (recall, $\mathscr{C}_{\mathrm{UL}}(b,U) = \mathsf{stsp}(b,U)^{-1}$) that

$$\mathsf{stsp}(b,U) - 2^{-D\max\{1,\log_2 \mathscr{C}_{\mathrm{UL}}(b,U)\}} \geq \mathsf{stsp}(b,U) - \mathsf{stsp}(b,U)^D \geq \mathsf{stsp}(b,U)/2$$

whereas if $\mathscr{C}_{\mathrm{UL}}(b,U) \leq 2$ we have (since in this case $2^{-D} \leq 1/4 \leq \mathsf{stsp}(b,U)/2$)

$$\mathsf{stsp}(b,U) - 2^{-D\max\{1,\log_2 \mathscr{C}_{\mathrm{UL}}(b,U)\}} = \mathsf{stsp}(b,U) - 2^{-D} \geq \mathsf{stsp}(b,U)/2.$$

In both cases, $\mathsf{stsp}(b,U) - 2^{-D\max\{1,\log_2 \mathscr{C}_{\mathrm{UL}}(b,U)\}} \geq \mathsf{stsp}(b,U)/2$. This, together with (8.22), proves the claim (8.23). The bound (8.20), together with (8.21) and (8.23), thus implies that

$$n \geq DX/4 \tag{8.24}$$

with

$$X := \log_2\left(\max\{\lambda + \lambda^{-1}, N, [\![(y,A)]\!]_{\mathrm{S}}, \mathscr{C}_{\mathrm{UL}}(y,A)\}\right).$$

Because $\log_2 \max\{\alpha,\beta\} = \Omega(\log_2\alpha + \log_2\beta)$ we have

$$X = \Omega(\log_2((\lambda+\lambda^{-1})N[\![(y,A)]\!]_{\mathrm{S}}\mathscr{C}_{\mathrm{UL}}(y,A)) \tag{8.25}$$

and

$$\begin{aligned} X = \Omega(\log_2((\lambda+\lambda^{-1})N[\![(y,A)]\!]_{\mathrm{S}}) &= \Omega(\log_2((\lambda+\lambda^{-1})N[\![(y,A)]\!]_{\mathrm{S}}^2)) \\ &= \Omega\left(\log_2 \frac{C}{\delta^{1/4}}\right). \end{aligned} \tag{8.26}$$

By Proposition 7.3(1), for $\sigma_1(y,A) < \lambda/4$, we have $\log_2(\sigma_1(y,A)) \geq \log_2(\mathsf{stsp}(y,A)) - \log_2\left(\frac{4\|A\|_{\max}}{\lambda}\right)$. Hence, using that $[\![(y,A)]\!]_{\mathrm{S}} \geq \|A\|_{\max}$ and that $\mathscr{C}_{\mathrm{UL}}(y,A) = \mathsf{stsp}(y,A)^{-1}$,

$$\begin{aligned} \log_2(\sigma_1(y,A)) + n &\underset{(8.24)}{\geq} -\log_2 \mathscr{C}_{\mathrm{UL}}(y,A) - \log_2\left(\frac{4\|A\|_{\max}}{\lambda}\right) + DX/4 \\ &\underset{(8.25)}{\geq} \mathcal{O}(X) - DX/4 \geq KX \underset{(8.26)}{\geq} 2\log_2(C) + n \end{aligned}$$

where the constant K is chosen so that $KX \geq 2\log_2(C) + 2n$ (in particular, this implies the last inequality) and then the penultimate inequality is ensured by choosing D sufficiently large. If instead $\sigma_1(y,A) \geq \lambda/4$, we use that $X \geq |\log_2\lambda|$ to deduce that

$$\log_2(\sigma_1(y,A)) + n \geq \log_2(\lambda/4) + DX/4 \geq KX \geq 2\log_2(C) + n$$

the last part of the reasoning being identical to the above. In both cases, we get $\sigma_1(y,A) \geq C^2$.

We next use Proposition 7.3(2–3) to get, reasoning as above,

$$\log_2(\sigma_2(y,A)) + n \geq -2\log_2 \mathscr{C}_{\mathrm{UL}}(y,A) + \frac{DX}{4} \geq \frac{KX}{2} \geq \log_2(C/\delta^{\frac{1}{4}}) = \log_2(C) + n$$

$$\log_2(\sigma_3(y,A)) + n \geq -\log_2 \mathscr{C}_{\mathrm{UL}}(y,A) - \log_2(\|A\|_{\max}) + \frac{DX}{4} \geq KX \geq 2\log_2(C) + n$$

which shows $\sigma_2(y,A) \geq C$ and $\sigma_3(y,A) \geq C^2$, as we wanted to prove. From the definition of σ we conclude that $\sigma(y,A) \geq C^2$.

All that remains is to show that $[\![(y,A)]\!]_{\max}(mN)^{-1/2} \geq 2\delta$. But for any $D \geq 2$, it follows from (8.20) that

$$n - 1 \geq n/2 \geq \log_2 N \geq \log_2((mN)^{1/2}) \geq \log_2((mN)^{1/2}) - \log_2[\![(y,A)]\!]_{\max}$$

the third inequality as $m \leq N$ and the last as $[\![(y, A)]\!]_{\max} \geq 1$. This completes the proof by noting that $2^{1-n} = 2\delta$. □

8.7.3. *A bound on the number of arithmetic operations performed at the nth iteration of the repeat loop.* We analyse each section of the **repeat** loop line by line, calculating the number of operations performed in big $\mathcal{O}$ notation. The first two lines, $n := n + 1$ and $\delta := \delta/16$, can be done using two arithmetic operations.

Next we analyse the calls to oracles and subroutines. Firstly, $\mathsf{GetMatrix}^U(n)$ and $\mathsf{GetVector}^b(n)$ are executed with cost $\mathcal{O}(mN(\log_2(\|U\|_{\max} + 1) + n))$ and $\mathcal{O}(m(\log_2(\|b\|_\infty + 1) + n))$, respectively. Both quantities are bounded by $\mathcal{O}(N^2(\log_2[\![(b, U)]\!]_{\max} + n))$. The call to $\mathsf{ULasso}(y, A, \lambda)$ takes $\mathcal{O}(N^3(p + \log_2(N))$ operations where p is the largest number of bits for the entries of y, A and λ. Note that $p \leq \max\{\log_2(\|A\|_{\max} + 1) + n, \log_2(\|y\|_\infty + 1) + n, p(\lambda)\} \leq \max\{\log_2(3[\![(b, U)]\!]_{\max}) + n, p(\lambda)\}$ and thus the runtime of $\mathsf{ULasso}(y, A, \lambda)$ is $\mathcal{O}[N^3(\max\{\log_2([\![(b, U)]\!]_{\max}) + n, p(\lambda)\} + \log_2(N))]$. Calculating $\delta^{1/4}$ can be done in one arithmetic operation. We can calculate H and G in $\mathcal{O}(mN) = \mathcal{O}(N^2)$ operations. To calculate C requires five multiplications and one addition of already calculated quantities and thus takes $\mathcal{O}(1)$ time. Finally, the call to Sigma takes $\mathcal{O}(N^3)$ operations (Proposition 8.5).

Thus the total runtime of the nth iteration of the loop is

$$\begin{aligned}&\mathcal{O}(N^2(\log_2[\![(b, U)]\!]_{\max} + n)\\ &+ \mathcal{O}[N^3(\max\{\log_2([\![(b, U)]\!]_{\max}) + n + 1, p(\lambda)\} + \log_2(N))] + \mathcal{O}(N^3)\end{aligned}$$

which is equal to

$$\mathcal{O}[N^3(\max\{\log_2([\![(b, U)]\!]_{\max}) + n, p(\lambda)\} + \log_2(N))].$$

8.7.4. *The overall runtime of the algorithm.* By Lemma 8.7, the number of iterations of FSUL is bounded above by some r with

$$r = \mathcal{O}(\lceil \log_2\left(\max\{\lambda + \lambda^{-1}, N, [\![(b, U)]\!]_{\max}, \mathscr{C}_{\mathrm{UL}}(b, U)\}\right)\rceil).$$

Thus, the total runtime is bounded above by

$$\begin{aligned}&\sum_{n=1}^{r} \mathcal{O}[N^3(\max\{\log_2([\![(b, U)]\!]_{\max}) + n + 1, p(\lambda)\} + \log_2(N))]\\ &= \mathcal{O}\left\{N^3\left[r\log_2([\![(b, U)]\!]_{\max}) + r^2 + rp(\lambda) + r\log_2(N)\right]\right\}.\end{aligned}$$

Clearly, $\log_2(N) = \mathcal{O}(r)$ and $r\log_2([\![(b, U)]\!]_{\max}) = \mathcal{O}(r^2)$. Hence the total runtime is bounded above by $\mathcal{O}(N^3r^2 + N^3rp(\lambda))$ i.e.

$$\begin{aligned}&\mathcal{O}\big\{N^3\left[\log_2\left(N^2(\lambda + \lambda^{-1})^2[\![(b, U)]\!]^2_{\max}\mathscr{C}_{\mathrm{UL}}(b, U)\right)\right]^2\\ &\quad + N^3\log_2\left(N^2(\lambda + \lambda^{-1})^2[\![(b, U)]\!]^2_{\max}\mathscr{C}_{\mathrm{UL}}(b, U)\right)p(\lambda)\big\}\end{aligned}$$

Coupled with the earlier argument to show correctness for the algorithm, we have completed the proof of Theorem 2.1 parts (1) and (2). □

8.8. **Proof of Theorem 2.1, part (3).** We start by proving the following:

Lemma 8.8. *Let $(y, U) \in \Omega$ and $x \in \mathsf{Sol}^{\mathsf{ms}}(y, U)$. Then if E is the diagonal matrix with entries $(\mathbb{1}_{\{1\notin\mathsf{supp}(x)\}}, \mathbb{1}_{\{2\notin\mathsf{supp}(x)\}}, \ldots, \mathbb{1}_{\{N\notin\mathsf{supp}(x)\}})$ where $\mathbb{1}_{\{i\notin\mathsf{supp}(x)\}}$ is 1 if $i \notin \mathsf{supp}(x)$ and 0 otherwise and if $\delta \in (0, 1)$ we have then $\mathsf{Sol}^{\mathsf{UL}}(y, U(\mathrm{I} - \delta E^1)) = \{x\}$.*

Proof of Lemma 8.8. This proof is similar to [7, Lemma 17.3]. We show that x is the unique vector in $\mathsf{Sol}^{\mathsf{UL}}(y, U - \delta UE)$.

Suppose that v is such that $\|(U - \delta UE)v - y\|_2^2 + \lambda\|v\|_1 \leq \|(U - \delta UE)x - y\|_2^2 + \lambda\|x\|_1$. We claim that $\mathsf{supp}(v) \subseteq \mathsf{supp}(x)$. Otherwise, if we let $\hat{v}$ be defined by $\hat{v}_i = v_i$ whenever $i \in \mathsf{supp}(x)$ and $\hat{v}_i = (1 - \delta)v_i$

whenever $i \notin \mathsf{supp}(x)$, we obtain $(U - UE^1)v = U\hat{v}$ and $\|\hat{v}\|_1 < \|v\|_1$ (the strict inequality follows from $\mathsf{supp}(v) \not\subseteq \mathsf{supp}(x)$) and hence (setting $f(w) := \|Uw - y\|_2^2 + \lambda\|w\|_1$)

$$f(\hat{v}) < \|(U - \delta UE)v\|_2^2 + \lambda\|v\|_1 \leq \|(U - \delta UE)x - y\|_2^2 + \lambda\|x\|_1 = f(x)$$

contradicting $x \in \mathsf{Sol}^{\mathsf{UL}}(y, U)$. Hence $\mathsf{supp}(v) \subseteq \mathsf{supp}(x^1)$. But then $Uv = (U - \delta UE)v$ and so

$$f(v) = \|(U - \delta UE)v\|_2^2 + \lambda\|v\|_1 \leq \|(U - \delta UE)x - y\|_2^2 + \lambda\|x^1\|_1 = f(x^1)$$

so $v \in \mathsf{Sol}^{\mathsf{UL}}(y, U)$ with $\mathsf{supp}(v) = \mathsf{supp}(x)$. It follows from $x \in \mathsf{Sol}^{\mathsf{ms}}(y, U)$ that $v = x$ and so x is the unique vector in $\mathsf{Sol}^{\mathsf{UL}}(y, U - UE^1)$. □

Proof of Theorem 2.1, part (3). We will consider two cases: firstly, the case where there exists $(b, U) \in \Omega$ with $|\Xi(b, U)| > 1$ and secondly the case where all $(b, U) \in \Omega$ have $|\Xi(b, U)| = 1$.

Case 1: There exists $(b, U) \in \Omega$ with $|\Xi(b, U)| > 1$.

By Lemma 8.2, there exists $x^1, x^2 \in \Xi^{\mathrm{ms}}(b, U)$ with $\mathsf{supp}(x^1) \neq \mathsf{supp}(x^2)$. We now use Lemma 8.8 to see that for each $n \in \mathbb{N}$ and for $i = 1, 2$, there exists $b^{i,n} \in \mathbb{R}^m, U^{i,n} \in \mathbb{R}^{m\times N}$ so that $\Xi(b^{i,n}, U^{i,n}) = \mathsf{supp}(x^i)$ and $\|b^{i,n} - b\|_\infty \leq 2^{-n}$, $\|U^{i,n} - U\|_{\max} \leq 4^{-n}$. Since Ω is open, we can pass to a subsequence and assume that $(b^{i,n}, U^{i,n}) \in \Omega$ for all $n \in \mathbb{N}$ and $i \in \{1, 2\}$. The conclusion now follows by an application of Proposition 9.7, where we choose $\iota_n^1 = (b^{1,n}, U^{1,n})$, $\iota_n^2 = (b^{2,n}, U^{2,n})$ and $\iota^0 = (b, U)$.

Case 2: All $(b, U) \in \Omega$ have $|\Xi(b, U)| = 1$.

Let $\Xi(y, A) = s^0$. By assumption, $\mathscr{C}_{\mathrm{UL}}(y, A) = \infty$. Thus there exists a sequence $(y^n, A^n) \in \mathbb{R}^m \times \mathbb{R}^{m\times N}$ with $\|y^n - y\|_\infty \leq 4^{-n}$, $\|A^n - A\|_{\max} \leq 4^{-n}$ and $\Xi(y^n, A^n) = s^{1,n}$ with $s^{1,n} \neq s^0$. Since Ω is open we can assume by potentially passing to a subsequence that $(y^n, A^n) \in \Omega$. Furthermore, since the number of possible supports for vectors in $\mathbb{R}^N$ is finite, $\{s^{1,n} \,|\, n \in \mathbb{N}\}$ is finite and we can again pass to a subsequence to assume that $s^{1,n} = s^1$. Once again, the conclusion now follows by an application of Proposition 9.7, where we choose $\iota_n^1 = (y^{1,n}, A^{1,n})$ and $\iota^0 = \iota_n^2 = (y, A)$. □

9. Tools from the SCI hierarchy and GHA

In this section we explain the SCI framework in the context of the LASSO problem. The purpose of introducing this framework is to ensure that Theorem 2.1, part (3)] is proven with as much generality as possible, taking into account the wide variety of computational models (e.g. Turing, BSS, Von-Neumann, etc.) that are prevalent in the optimisation literature. We start by defining the LASSO feature selection as a 'Computational problem' following the setup of [7].

Definition 9.1 (The LASSO computational problem)**.** For some set $\Omega \subset \mathbb{R}^m \times \mathbb{R}^{m\times N}$, which we call the *input* set, the *LASSO computational problem on* Ω is the collection $\{\Xi, \Omega, \mathbb{B}^N, \Lambda\}$ where $\Xi : \Omega \to 2^{\mathbb{B}^N}$ is defined as in (1.2) and

$$\Lambda = \{f^{\mathrm{vec}}, f^{\mathrm{mat}}\} \text{ with } f^{\mathrm{vec}} : \Omega \to \mathbb{R}^m, f^{\mathrm{mat}} : \Omega \to \mathbb{R}^{m\times N}$$

are defined by $f^{\mathrm{vec}}(y, A) = y$ and $f^{\mathrm{mat}}(y, A) = A$ for all $(y, A) \in \Omega$.

We want to generalise the LASSO computational problem so that we work with *inexact inputs*. To do so, we will consider the collection of all functions $f_n^{\mathrm{vec}} : \Omega \to \mathbb{R}^m$ and $f_n^{\mathrm{mat}} : \Omega \to \mathbb{R}^{m\times N}$ satisfying

$$\|f_n^{\mathrm{vec}}(y, A) - y\|_\infty \leq 2^{-n}, \quad \|f_n^{\mathrm{mat}}(y, A) - A\|_{\max} \leq 2^{-n} \tag{9.1}$$

for all $(y, A) \in \Omega$. To handle inexact input we follow [7] and replace the exact input set, Ω, by the inexact input set $\tilde{\Omega}$ to form the *inexact LASSO computational problem*.

Definition 9.2 (Inexact LASSO computational problem)**.** The *inexact LASSO computational problem on* Ω (ILCP) is the quadruple $\{\tilde{\Xi}, \tilde{\Omega}, \mathbb{B}^N, \tilde{\Lambda}\}$, where

$$\begin{aligned}\tilde{\Omega} = \big\{\tilde{\iota} = \{(f_n^{\mathrm{vec}}(\iota), f_n^{\mathrm{mat}}(\iota)\}_{n\in\mathbb{N}} \,|\, \iota = (y, A) \in \Omega \text{ and} \\ f_n^{\mathrm{vec}} : \Omega \to \mathbb{R}^m, f_n^{\mathrm{mat}} : \Omega \to \mathbb{R}^{m\times N} \text{ satisfy (9.1) respectively}\big\}\end{aligned} \tag{9.2}$$

It follows from (9.1) that there is a unique $\iota = (y, A) \in \Omega$ for which

$$\tilde{\iota} = \left\{(f_n^{\mathrm{vec}}(\iota), f_n^{\mathrm{mat}}(\iota))\right\}_{n\in\mathbb{N}}.$$

We say that this $\iota \in \Omega$ *corresponds* to $\tilde{\iota} \in \tilde{\Omega}$ and we set $\tilde{\Xi} : \tilde{\Omega} \rightrightarrows \mathbb{B}^N$ so that $\Xi(\tilde{\iota}) = \Xi(\iota)$, and $\tilde{\Lambda} = \{\tilde{f}_n^{\mathrm{vec}}, \tilde{f}_n^{\mathrm{mat}}\}_{n\in\mathbb{N}}$, with $\tilde{f}_n^{\mathrm{vec}}(\tilde{\iota}) = f_n(\iota)$, $\tilde{f}_n^{\mathrm{mat}}(\tilde{\iota}) = f_n^{\mathrm{mat}}(\iota)$ where ι corresponds to $\tilde{\iota}$.

Part (3) in Theorem 2.1 is a negative result (it states that a computational problem cannot be solved). To make it as general as possible we want it to apply to a broad class of algorithms. These are defined, roughly speaking, by a single requirement namely, that for a given $\tilde{\iota} \in \tilde{\Omega}$, the algorithm finds any member of $\tilde{\Xi}(\tilde{\iota})$ by accessing finitely many values of $\tilde{f}(\tilde{\iota})$ where $\tilde{f} \in \tilde{\Lambda}$. This captures the idea that any algorithm that solves the ILCP should return an answer by accessing arbitrarily many (but finitely many) approximations to the true input. Note, the algorithm should work for any choice of approximations and any input but it may return different results depending on which approximations it sees.

The type of algorithm described above are called *general algorithms*. As noted in [7], the purpose of a general algorithm is to have a definition that encompasses any model of computation. This ensures that Theorem 2.1(3) holds in all relevant computational models considered by the optimisation community (e.g. Turing, BSS).

Definition 9.3 (General Algorithms for the ILCP)**.** A *general algorithm* for $\{\tilde{\Xi}, \tilde{\Omega}, \mathbb{B}^N, \tilde{\Lambda}\}$, is a mapping $\Gamma : \tilde{\Omega} \to \mathbb{B}^N$ such that, for every $\tilde{\iota} \in \tilde{\Omega}$, the following conditions hold:

(i) there exists a nonempty subset of evaluations $\Lambda_\Gamma(\tilde{\iota}) \subset \tilde{\Lambda}$ with $|\Lambda_\Gamma(\tilde{\iota})| < \infty$,
(ii) the action of Γ on $\tilde{\iota}$ is uniquely determined by $\{f(\tilde{\iota})\}_{f\in\Lambda_\Gamma(\tilde{\iota})}$,
(iii) for every $\iota' \in \Omega$ such that $f(\iota') = f(\tilde{\iota})$ for all $f \in \Lambda_\Gamma(\tilde{\iota})$, it holds that $\Lambda_\Gamma(\iota') = \Lambda_\Gamma(\tilde{\iota})$.

This will prove useful when we come to define randomised general algorithms to prove the random setting statement in Theorem 2.1(3). We can thus proceed to define randomised general algorithms.

Definition 9.4 (Randomised General Algorithm for the ILCP)**.** A *randomised general algorithm* (RGA) for $\{\tilde{\Xi}, \tilde{\Omega}, \mathbb{B}^N, \tilde{\Lambda}\}$ is a collection X of general algorithms $\Gamma : \tilde{\Omega} \to \mathbb{B}^N$, a sigma-algebra $\mathcal{F}$ on X, and a family of probability measures $\{\mathbb{P}_\iota\}_{\iota\in\Omega}$ on $\mathcal{F}$ such that the following conditions hold:

(Pi) For each $\iota \in \tilde{\Omega}$, the mapping $\Gamma_\iota^{\mathrm{ran}} : (X, \mathcal{F}) \to (\mathbb{B}^N, \mathcal{B})$ defined by $\Gamma_\iota^{\mathrm{ran}}(\Gamma) = \Gamma(\iota)$ is a random variable, where $\mathcal{B}$ is the Borel sigma-algebra on $\mathbb{B}^N$.
(Pii) For each $n \in \mathbb{N}$ and $\iota \in \tilde{\Omega}$, we have $\{\Gamma \in X \,|\, T_\Gamma(\iota) \leq n\} \in \mathcal{F}$, where

$$T_\Gamma(\tilde{\iota}) := \sup\{m \in \mathbb{N} \,|\, \text{ either } f_m^{\mathrm{vec}} \in \Lambda_\Gamma(\tilde{\iota}) \text{ or } f_m^{\mathrm{mat}} \in \Lambda_\Gamma(\tilde{\iota})\}.$$

(Piii) For all $\iota_1, \iota_2 \in \tilde{\Omega}$ and $E \in \mathcal{F}$ so that, for every $\Gamma \in E$ and every $f \in \Lambda_\Gamma(\iota_1)$, we have $f(\iota_1) = f(\iota_2)$, it holds that $\mathbb{P}_{\iota_1}(E) = \mathbb{P}_{\iota_2}(E)$.

Remark 9.5. The quantity $T_\Gamma(\tilde{\iota})$ is known as the minimum amount of input information in [7].

Definition 9.6 (Halting randomised general algorithms)**.** A randomised general algorithm $\Gamma^{\mathrm{ran,h}}$ for a computational problem $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ is called a *halting randomised general algorithm* (hRGA) if $\mathbb{P}_\iota(\Gamma_\iota^{\mathrm{ran,h}} = \mathsf{NH}) = 0$, for all $\iota \in \Omega$.

We make use of the following propositions, taken from [7] and simplified for the specific problem under consideration in this paper.

Proposition 9.7. *(Simplified from [7], Proposition 9.5) Suppose that a subset Ω of $\cup_{N=1}^\infty \cup_{m=1}^N \mathbb{R}^m \times \mathbb{R}^{m\times N}$ is such that there exists two sequences $\{\iota_n^1\}_{n=1}^\infty, \{\iota_n^2\}_{n=1}^\infty \subset \Omega$ and an $\iota^0 \in \Omega$ satisfying the following conditions:*

(a) There are disjoint sets $S^1, S^2 \subset \mathbb{B}^N$ with $\Xi(\iota_n^i) \subset S^i$ for $i = 1, 2$ where Ξ is defined as in Definition 9.1.
(b) $\iota_n^i = (y^{i,n}, A^{i,n})$ and $\iota^0 = (y, A)$ with $\|y^{i,n} - y\|_\infty \leq 1/4^n$ and $\|A^{i,n} - A\|_{\max} \leq 1/4^n$ for all $n \in \mathbb{N}$ and $i = 1, 2$.

Then each of the following holds:

(1) For any general algorithm $\Gamma : \tilde{\Omega} \to \mathbb{B}^N$, there must exist a $\tilde{\iota} \in \tilde{\Omega}$ so that $\Gamma(\tilde{\iota}) \neq \tilde{\Xi}(\tilde{\iota})$.

(2) For any $p > 0$ and halting randomised general algorithm $\Gamma^{\mathrm{ran,h}}$, there must exist $\tilde{\iota}$ so that the probability that $\Gamma^{\mathrm{ran,h}}(\tilde{\iota}) \neq \tilde{\Xi}(\tilde{\iota})$ is at least $1/2 - p$.

REFERENCES

[1] B. Adcock and A. C. Hansen. *Compressive Imaging: Structure, Sampling, Learning*. Cambridge University Press, 2021.

[2] D. Amelunzen, M. Lotz, M. B. McCoy, and J. A. Tropp. Living on the edge : phase transitions in convex programs with random data. *Information and Inference*, 3(3):224–294, June 2014.

[3] V. Antun, M. J. Colbrook, and A. C. Hansen. Proving existence is not enough: Mathematical paradoxes unravel the limits of neural networks in artificial intelligence. *SIAM News*, 55(04):1–4, May 2022.

[4] V. Antun, F. Renna, C. Poon, B. Adcock, and A. C. Hansen. On instabilities of deep learning in image reconstruction and the potential costs of AI. *Proc. Natl. Acad. Sci. USA*, 117(48):30088–30095, 2020.

[5] S. Arora and B. Barak. *Computational Complexity - A Modern Approach*. Princeton University Press, 2009.

[6] A. Bastounis, F. Cucker, and A. C. Hansen. When can you trust feature selection? – ii: On the effects of random data on condition in statistics and optimisation. *Preprint*, 2023.

[7] A. Bastounis, A. C. Hansen, and V. Vlačić. The extended Smale's 9th problem – On computational barriers and paradoxes in estimation, regularisation, computer-assisted proofs and learning. *arXiv:2110.15734*, 2021.

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

[9] J. Ben-Artzi, M. J. Colbrook, A. C. Hansen, O. Nevanlinna, and M. Seidel. Computing spectra – On the solvability complexity index hierarchy and towers of algorithms. *arXiv:1508.03280v5*, 2020.

[10] J. Ben-Artzi, M. Marletta, and F. Rösler. Computing scattering resonances. *Journal of the European Mathematical Society*, 2023.

[11] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton Series in Applied Mathematics. Princeton University Press, October 2009.

[12] A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88(3):411–424, 2000.

[13] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation*. Springer-Verlag, 1998.

[14] L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *BAMS*, 21:1–46, 1989.

[15] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[16] P. Bürgisser and F. Cucker. *Condition: The Geometry of Numerical Algorithms*. Number 349 in Grundlehren der matematischen Wissenschaften. Springer Verlag, 2013.

[17] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *J. Math. Imaging Vis.*, 40(1):120–145, May 2011.

[18] A. Chambolle and T. Pock. An introduction to continuous optimization for imaging. *Acta Numerica*, 25:161–319, 2016.

[19] D. Cheung and F. Cucker. A new condition number for linear programming. *Mathematical Programming*, 91(1):163–174, 2001.

[20] D. Cheung and F. Cucker. Solving linear programs with finite precision: I. Condition numbers and random programs. *Math. Programming*, 99:175–196, 2004.

[21] D. Cheung and F. Cucker. A note on level-2 condition numbers. *Journal of Complexity*, 21(3):314–319, 2005.

[22] D. Cheung, F. Cucker, and J. Peña. On strata of degenerate polyhedral cones. I: Condition and distance to stratae. *European Journal of Operational Research*, 198:23–28, 2009.

[23] C. Choi. 7 revealing ways AIs fail. *IEEE Spectrum*, September, 2021.

[24] C. Choi. Some AI systems may be impossible to compute. *IEEE Spectrum*, March, 2022.

[25] M. Colbrook and A. C. Hansen. The foundations of spectral computations via the solvability complexity index hierarchy. *Journal of the European Mathematical Society*, 2022 (online).

[26] M. J. Colbrook, V. Antun, and A. C. Hansen. The difficulty of computing stable and accurate neural networks: On the barriers of deep learning and smale's 18th problem. *Proc. Natl. Acad. Sci. USA*, 119(12):e2107151119, 2022.

[27] F. Cucker and S. Smale. Complexity estimates depending on condition and round-off error. *Journal of the ACM*, 46(1):113–184, 1999.

[28] C. Fefferman, A. C. Hansen, and S. Jitomirskaya, editors. *Computational mathematics in computer assisted proofs*, American Institute of Mathematics Workshops. American Institute of Mathematics, 2022. Available online at `https://aimath.org/pastworkshops/compproofsvrep.pdf`.

[29] L. E. Gazdag and A. C. Hansen. Generalised hardness of approximation and the SCI hierarchy – On determining the boundaries of training algorithms in AI. *arXiv:2209.06715*, 2022.

[30] H. Goldstine and J. von Neumann. Numerical inverting matrices of high order, II. *Proc. Amer. Math. Soc.*, 2:188–202, 1951.

[31] N. M. Gottschling, V. Antun, A. C. Hansen, and B. Adcock. The troublesome kernel – on hallucinations, no free lunches and the accuracy-stability trade-off in inverse problems. 2023.

[32] T. C. Hales. A proof of the Kepler conjecture. *Ann. of Math. (2)*, 162(3):1065–1185, 2005.

[33] T. C. Hales and et al. A formal proof of the kepler conjecture. *Forum of Mathematics, Pi*, 5:e2, 2017.

[34] K. Hammernik, T. Klatzer, E. Kobler, M. P. Recht, D. K. Sodickson, T. Pock, and F. Knoll. Learning a variational network for reconstruction of accelerated MRI data. *Magnetic Resonance in Medicine*, 79(6):3055–3071, 2018.

[35] A. C. Hansen. On the solvability complexity index, the n-pseudospectrum and approximations of spectra of operators. *Journal of the American Mathematical Society*, 24(1):81–124, 2011.

[36] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[37] T. Hastie, R. Tibshirani, and M. Wainwright. *Statistical Learning with Sparsity: The Lasso and Generalizations (Chapman & Hall/CRC Monographs on Statistics & Applied Probability)*. Chapman and Hall/CRC, May 2015.

[38] D. Heaven et al. Why deep-learning AIs are so easy to fool. *Nature*, 574(7777):163–166, 2019.

[39] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2002.

[40] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser. Deep convolutional neural network for inverse problems in imaging. *IEEE Transactions on Image Processing*, 26(9):4509–4522, 2017.

[41] A. Juditsky, F. Kilinç-Karzan, A. Nemirovski, and B. Polyak. Accuracy guaranties for ℓ_1 recovery of block-sparse signals. *The Annals of Statistics*, 40(6):3077 – 3107, 2012.

[42] D. Knuth. Big omicrom and big omega and big theta. *SIGACT News*, pages 18–24, Apr.-Jun. 1976.

[43] M. Lotz, D. Amelunxen, and J. Walvin. Effective condition number bounds for convex regularization. *IEEE Transactions on Information Theory*, Jan. 2020.

[44] M. T. McCann, K. H. Jin, and M. Unser. Convolutional neural networks for inverse problems in imaging: A review. *IEEE Signal Process Magazine*, 34(6):85–95, 2017.

[45] R. D. C. Monteiro and I. Adler. Interior path following primal-dual algorithms. part ii: Convex quadratic programming. *Mathematical Programming*, 44(1):43–66, May 1989.

[46] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. In *IEEE Conference on computer vision and pattern recognition*, pages 86–94, July 2017.

[47] L. Narici and E. Beckenstein. *Topological Vector Spaces*. Chapman and Hall/CRC, July 2010.

[48] A. Nemirovski. Lectures on Robust Convex Optimization. Available online at `https://www2.isye.gatech.edu/~nemirovs/`, 2009.

[49] Y. E. Nesterov and A. Nemirovski. On first-order algorithms for l1/nuclear norm minimization. *Acta Numer.*, 22:509–575, 2013.

[50] J. Peña. Conditioning of convex programs from a primal-dual perspective. *Mathematics of Operations Research*, 26(2):206–220, 2001.

[51] J. Peña. Two properties of condition numbers for convex programs via implicitly defined barrier functions. *Mathematical Programming*, 93(1):55–75, 2002.

[52] H. R, J. H, and S. M. JI. Robustness and explainability of artificial intelligence. (KJ-NA-30040-EN-N (online)), 2020.

[53] J. Renegar. A polynomial-time algorithm, based on newton's method, for linear programming. *Mathematical Programming*, 40(1-3):59–93, 1988.

[54] J. Renegar. Is it possible to know a problem instance is ill-posed? *J.of Complexity*, 10:1–56, 1994.

[55] J. Renegar. Incorporating condition measures into the complexity theory of linear programming. *SIAM Journal on Optimization*, 5(3):506–524, 1995.

[56] J. Renegar. Linear programming, complexity theory and elementary functional analysis. *Mathematical Programming*, 70(1):279–351, 1995.

[57] J. Renegar. Condition numbers, the barrier method, and the conjugate-gradient method. *SIAM Journal on Optimization*, 6:879–912, 1996.

[58] J. Renegar. *A mathematical view of interior-point methods in convex optimization*, volume 3. Siam, 2001.

[59] S. Smale. Mathematical problems for the next century. In V. Arnold, M. Atiyah, P. Lax, and B. Mazur, editors, *Mathematics: Frontiers and Perspectives*. American Mathematical Society, 2000.

[60] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994.

[61] R. J. Tibshirani. The lasso problem and uniqueness. *Electron. J. Statist.*, 7:1456–1490, 2013.

[62] A. Turing. Rounding-off errors in matrix processes. *Quart. J. Mech. Appl. Math.*, 1:287–308, 1948.

[63] J. von Neumann and H. Goldstine. Numerical inverting matrices of high order. *Bull. Amer. Math. Soc.*, 53:1021–1099, 1947.

[64] J. Wilkinson. *Rounding Errors in Algebraic Processes*. Prentice Hall, 1963.

[65] S. Wright and B. Recht. *Optimization for Data Analysis*. Cambridge University Press, 2022.

[66] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009.

UNIVERSITY OF LEICESTER, UK
Email address: ajb177@leicester.ac.uk

CITY UNIVERSITY OF HONG KONG, HONG KONG
Email address: macucker@gmail.com

UNIVERSITY OF CAMBRIDGE, UK
Email address: ach70@cam.ac.uk