

Randomized algorithms and theory for rank estimation and least squares



Maike Meier

Balliol College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

December 2024

To my parents and brother, whose love and support make everywhere feel like home

Abstract

This thesis is concerned with randomized algorithms for two matrix problems: numerical rank estimation, with extensions to low-rank approximation, and numerical solutions to least squares (LS) problems. As the dimensions of matrix problems in modern computational settings grow, classical algorithms become increasingly insufficient in terms of computational, communication, and storage costs. Randomized algorithms have been shown to provide excellent alternatives in large-scale settings, and they are the main focus of this thesis. The work focuses on under-explored elements of randomized algorithms, such as numerical stability and parameter selection. The main overarching aim is to develop theory and algorithms that further aid in bringing randomized numerical linear algebra towards the mainstream of computational science.

Randomized algorithms have achieved significant speed-ups for the computation of low-rank approximations. However, these algorithms often require the target rank as input, which is difficult to determine a priori if a specified accuracy is desired. Within this context, a randomized rank estimation algorithm for approximately low-rank matrices is presented in Chapter 3. The algorithm is based on dimension reduction using random embedding matrices (sketching), and is theoretically and experimentally shown to preserve the order of magnitude of leading singular values. This allows users to inspect the approximate spectrum of the matrix as part of a low-rank approximation algorithm, and adjust the target rank as necessary – thus saving computational cost. The rank estimation algorithm is suitable for matrices with rapidly decaying singular values, specifically for which the tail is much smaller than the dominant singular values.

Next we consider linear LS problems and (randomized) iterative solvers to compute approximate solutions. Sketch-and-precondition is a state-of-the-art algorithmic framework for solving highly-overdetermined LS problems. Through sketching, an excellent randomized preconditioner is obtained at low computational cost. This preconditioner is then used in combination with an iterative (Krylov subspace) method to solve the LS problem. In Chapter 4, we first discuss different metrics for the quality of approximate solutions of LS problems and show how available (classical) algorithms perform on these metrics. It is then shown that a particular class of implementations of the sketch-and-precondition framework does not perform well on various performance metrics for a range of problems.

This is caused by rounding error accumulating throughout the run of the algorithm. In particular, we conclude that the sketch-and-precondition algorithm without initialization is not numerically stable. Numerical evidence suggests that these instabilities are not necessarily caused by the randomization, but rather inherent to preconditioned Krylov subspace solvers for LS problems and that rounding errors are incurred specifically by repeated application of a preconditioner.

In Chapter 5, we aim to resolve the numerical instabilities present in the sketch-and-precondition algorithm without initialization. We first note the importance of a good initialization for improved stability properties. This is in agreement with Rokhlin and Tygert (2008) and Cartis, Fiala, and Shao (2021) who include it in their sketch-and-precondition implementations. However, we find that sketch-and-solve initialization it is inadequate for guaranteeing stability. We propose a variation, coined sketch-and-apply, and show that it is numerically stable using classical stability analysis. The algorithm is based on explicitly computing the preconditioned operator and solving the resulting well-conditioned system iteratively. This is one of the first randomized algorithms shown to be backward stable in a classical sense. However, the asymptotic complexity $\mathcal{O}(mn^2)$ of sketch-and-apply is on par with classical algorithms and slower than existing sketch-and-precondition variants.

Finally, in Chapter 6, the sketch-and-precondition framework is extended to regularized least squares problems. Regularization of LS problems is necessary when problems are ill-conditioned, noisy, and the main objective is to minimize the (forward) error, as is often the case in inverse problems. However, selecting a good parameter that controls the level of regularization is highly nontrivial. In Chapter 6 we present a variety of algorithms, all of which efficiently compute accurate solutions for a range of admissible regularization parameters. They are all based on inexpensively updating a randomized preconditioner and initial guess to solve a sequence of problems. One of the algorithms exploits low-rank structure in the problems with improved numerical stability as compared to previous work, and is suitable for even square problems provided they are sufficiently low-rank.

Through our investigation of parameter selection for low-rank approximation and regularized LS, we improve the cost-effectiveness and user-friendliness of randomized approaches to these problems. Our research on numerical stability of sketch-and-precondition algorithms and stabilizing suggestions aids in understanding the finite-precision behaviour of the algorithms. Guarantees on the performance of these algorithms in finite-precision arithmetic are crucial for widespread adoption of the methods. Investigation of these themes improves the understanding and performance of randomized algorithms for these

problems, and is important for the implementation of randomized algorithms in wider computational science.

Acknowledgements

First and foremost, I would like to express my deepest appreciation and gratitude to my supervisor Yuji Nakatsukasa. You have been the most supportive, enthusiastic, and kind supervisor any student could ask for. Your never-ending stream of ideas, together with an unstoppable belief that all of them are going to change the world, is what kept me loving maths. Thank you for all your support.

I express my deep gratitude to my examiners Coralia Cartis and Per-Gunnar Martinsson for examining this thesis. Their comments and our discussions during the viva have significantly improved the work.

I am grateful to Dr. Zhaodong Wang for funding my DPhil through the Oxford Wang Scholarship.

I am thankful to Alex Townsend and Marcus Webb, in collaboration with whom some of the work in this thesis was completed. Their advice greatly improved Chapter 5. I am especially indebted to Alex, who has supported me and from whom I have learned so much.

I am indebted to many of the faculty at the Mathematical Institute and beyond. In Oxford, I am incredibly thankful for Coralia Cartis, Nick Trefethen, Kathryn Gillow, Andy Wathen, Jared Tanner, Patrick Farrell, Endre Süli and Harald Oberhauser for their support and inspiring conversations. Beyond Oxford, I thank Hussam Al Daas and Ethan Epperly for the wonderful collaborations.

The numerical analysis group has been a wonderful home for the last 4 years. I was lucky to be around so many amazing people and thank all my friends for the memories and support. Thank you, Nicolas, Gonzalo, Pablo, Francis, and Alice for all those lunches in an empty common room. Because of you I found footing in the department. Many thanks to John, Diana, and Astrid for making maths conferences (extra) exciting. Boris, India, Charlie, Kars, Karl, Aaron, Umberto, and Mingdong; thank you so much for always providing good banter in the furthest hallway of the MI. Above all, Taejun and Lorenzo, I am so incredibly grateful to have both of you as my academic brothers, thank you for everything.

My time in Oxford would not have been the same without all the amazing people I had the pleasure of meeting at Oxford University Basketball Club. I am grateful to the incredibly supportive community I have found there.

I would not have gotten here without the endless support of my friends. Thank you to Charlotte, Hanne, Frigg for the very many years of love and good times. Thank you to all my friends who have made Oxford home, Kumeren, Conan, Anna, Adam, Alex, Alex, Chloe, Felicia, Verena, and Arthur. I am especially grateful to Momo, Boris, Casey, and Johnny; I would not have been able to make it to this point without them. Momo and Boris, thank you for all the wonderful moments together; Casey, thank you for your friendship always, and Johnny, thank you so very much for everything you've done for me.

Finally, my biggest thanks goes to my family. Thank you for your unwavering support, not just in the last 4 years but always. I would have never dreamed of all of this if it weren't for your inspiration, encouragement, and love. I can't wait to come home.

Contents

1	Introduction	1
1.1	Two fundamental matrix problems	2
1.1.1	Low-rank approximation and numerical rank	2
1.1.2	Linear least squares	4
1.1.3	Regularized least squares problems	6
1.2	The origins of randomized numerical linear algebra	7
1.2.1	Subsampling era	7
1.2.2	Random embeddings: fast relative error approximations	10
1.2.3	Sketch-and-precondition for highly accurate results	12
1.3	Randomized numerical linear algebra matures	12
1.3.1	Low-rank approximations	13
1.3.2	Least squares problems	14
1.3.3	Changing computational context	14
1.4	Modern randomized NLA and this thesis' contributions	15
1.4.1	Challenges in randomized NLA	16
1.4.1.1	Parameter selection	16
1.4.1.2	Numerical stability	17
1.4.2	Contributions of this thesis	17
1.4.2.1	Numerical rank estimation	17
1.4.2.2	The stability of sketch-and-precondition for LS	18
1.4.2.3	Randomized algorithms for regularized least squares	20
1.4.3	Relation to published work	20
1.5	Notation	21
1.6	Outline	21

2	Preliminaries	22
2.1	Sketching	22
2.1.1	Compression: sketching dominant spaces	22
2.1.2	Embedding: sketching full spaces and the spectrum	23
2.1.3	Sketching matrices	24
2.1.3.1	Gaussian embedding matrices	24
2.1.3.2	Subsampling matrices	24
2.1.3.3	Sparse random embeddings	25
2.1.3.4	Subsampled randomized trigonometric transforms	26
2.1.3.5	Hashed randomized trigonometric transforms	26
2.2	Randomized low-rank approximation	27
2.3	Randomized algorithms for LS problems	28
2.3.1	Sketch-and-solve	29
2.3.2	Sketch-and-precondition	30
2.4	Looking ahead	32
3	Fast randomized rank estimation for numerically low-rank matrices	33
3.1	Introduction	33
3.1.1	Numerical rank and goal of a rank estimator	35
3.2	Related work	37
3.2.1	Existing methods for numerical rank estimation	37
3.2.2	Related work in statistics	38
3.2.3	Related algorithms in (randomized) numerical linear algebra	39
3.3	Sketching roughly preserves singular values: $\sigma_i(AS)/\sigma_i(A) = \mathcal{O}(1)$ for leading i . . .	40
3.3.1	Bounds for general subspace embeddings	44
3.3.2	The effect of tail singular values and oversampling	45
3.3.2.1	The effect of a heavy tail	47
3.3.3	Choice of sketch matrix	49
3.3.4	A numerical experiment	50
3.4	Preserving all singular values with embedding: $\sigma_i(S_2AS_1)/\sigma_i(AS_1) = \mathcal{O}(1)$	51
3.4.1	A numerical experiment	54
3.5	A numerical rank estimation scheme	55
3.5.1	Algorithm and parameter choices	56
3.5.2	Rank estimation as part of randomized low-rank approximation	58
3.5.2.1	Fixed-precision low-rank approximation problem	58
3.6	Numerical experiments	60

3.6.1	Detecting gaps in the spectrum	61
3.6.2	Robustness of the rank estimator	61
3.6.3	Comparison with other numerical rank estimation algorithms	62
3.6.4	The fixed-precision problem	64
3.7	Summary and outlook	65
4	Iterative solvers for least squares problems and the numerical stability of sketch-and-precondition	68
4.1	Introduction	68
4.1.1	Accuracy metrics for approximate solutions to LS problems	70
4.2	Exact and approximate solutions to LS problems	71
4.2.1	The exact solution to a LS problem	71
4.2.2	Approximate solutions to a LS problem	73
4.2.3	Perturbation theory	73
4.2.4	Backward error and backward stability	75
4.2.5	Direct methods and their stability	76
4.3	Iterative Krylov subspace solvers: LSQR and LSMR	79
4.3.1	Ingredients of LSQR and LSMR	80
4.3.1.1	Golub-Kahan bidiagonalization	80
4.3.1.2	Projection onto the Krylov subspace and minimization	81
4.3.1.3	Updating the QR factorization	82
4.3.1.4	Updating the iterates	83
4.3.2	Stopping criteria	84
4.4	Convergence behaviour of LSQR and LSMR	86
4.4.1	Convergence in exact arithmetic	86
4.4.2	Finite precision behaviour	87
4.4.2.1	Empirical investigation of the finite precision behaviour	88
4.5	Preconditioned LSQR	90
4.5.1	The sketch-and-precondition framework	92
4.6	The numerical stability of sketch-and-precondition without initialization	94
4.6.1	Sensitivity to the condition number and consistency	95
4.6.2	Variants of sketch-and-precondition	95
4.7	The numerical stability of preconditioned LSQR	96
4.8	Summary and outlook	97

5	Stabilizing the sketch-and-precondition framework	99
5.1	Introduction	99
5.2	Stabilizing approaches to the sketch-and precondition framework	100
5.2.1	Sketch-and-precondition with sketch-and-solve initialization	101
5.2.2	Sketch-and-apply	102
5.2.3	Smoothed sketch-and-apply	104
5.3	Finite-precision preliminaries	105
5.3.1	Numerical stability analysis	105
5.3.1.1	Matrix-matrix multiplication	105
5.3.1.2	Householder QR factorization	106
5.3.1.3	Triangular system solving with substitution	106
5.3.2	Smoothed analysis of condition numbers	107
5.4	On the numerical stability of sketch-and-apply without smoothing	107
5.4.1	Bounding $\kappa_2(\hat{Y})$ in terms of numerical errors	108
5.4.2	Bounding the numerical errors	110
5.4.3	Bounding $\kappa_2(\hat{Y})$	111
5.4.4	The backward error of sketch-and-apply	112
5.4.5	Ill-conditioning caused by poor column scaling	114
5.5	On the numerical stability of smoothed sketch-and-apply	114
5.5.1	Bounding $\kappa_2(\hat{Y})$	115
5.5.2	The backward error of smoothed sketch-and-apply	116
5.6	Master sketch-and-apply algorithm and numerical experiments	117
5.6.1	Numerical experiments with sketch-and-apply	119
5.6.2	Numerical experiments with smoothed sketch-and-apply	120
5.6.3	Sparse matrices	121
5.7	Summary and outlook	122
6	Randomized algorithms for Tikhonov regularization in linear least squares	125
6.1	Introduction	125
6.1.1	An example inverse problem: computed tomography	126
6.1.2	Tikhonov regularization	128
6.1.2.1	Selecting a regularization parameter	128
6.1.2.2	Other regularization methods	130
6.1.3	Scope and focus of the algorithms	131
6.1.4	Sketch-and-precondition for Tikhonov regularization	132
6.1.5	Our contributions	133
6.1.5.1	General context	134

6.1.5.2	A context with low-statistical dimension	134
6.1.5.3	Convergence results	135
6.1.6	Related work	136
6.2	Randomized sketch-and-precondition algorithms for Tikhonov regularization in general settings	136
6.2.1	An SVD-based approach	137
6.2.1.1	A sketch-and-solve initial guess	138
6.2.1.2	Algorithm	138
6.2.2	An eigendecomposition-based approach	138
6.2.3	A Cholesky-based approach	139
6.2.4	Convergence analysis	140
6.2.5	Adaptation to underdetermined problems	141
6.3	Randomized Tikhonov regularization for problems with small statistical dimension	142
6.3.1	A preconditioner with low-rank structure	143
6.3.2	A sketch-and-solve initial guess	144
6.3.3	An algorithm exploiting small statistical dimension	144
6.3.3.1	Estimating the statistical dimension	145
6.3.4	Convergence analysis	146
6.3.5	Sketching matrices	148
6.4	A weighted regularization function	148
6.4.1	GSVD-based approach to the overdetermined problem	149
6.4.2	Convergence analysis	151
6.5	Numerical experiments	152
6.5.1	A square ill-posed inverse problem	152
6.5.2	A highly ill-conditioned sparse matrix	153
6.5.3	A dense and overdetermined problem from CT	154
6.6	Summary and outlook	155
7	Concluding remarks	157
	Bibliography	160

Chapter 1

Introduction

Today's world is data-driven. Modern applications generate enormous amounts of data, for instance, in weather forecasting [191], machine learning [287], tomographic imaging [10], medical imaging [280], astronomical surveys [286], and particle simulations [210]. Computations with these large datasets, arising in a wide range of sciences, are reliant on fundamental linear algebraic tasks. Recognising patterns in the data, filtering out noise, compressing data sets, and fitting data to a model are all important tools that can be reduced to linear algebraic problems. If one hopes to gain information from (big) data, it is inevitable to, e.g., compute low-rank decompositions, solve linear systems and optimization problems, and compute eigenvalues.

The scale of the data, however, is steadily moving beyond the reach of traditional computational routines [88]. Traditional numerical linear algebra (NLA) routines are highly accurate and optimized for moderately-sized matrices, but often come with significant computational, communication, and memory costs. Although the speed at which computers perform elementary operations has increased, traditional computing often still demands a computational cost too large for enormously large datasets [181]. Developments in hardware mean that communication costs – accessing and moving of data – are increasingly becoming the limiting factor [192]. New computational contexts arise when storing the data at all might be impossible. At the same time, it is appropriate in many applications to accept a trade-off between accuracy and computational speed, especially when computational errors are outweighed by inaccuracies in the data [88].

Computational scientists are therefore tasked with designing new approaches to computing in order to meet modern demands; there is significant need for algorithms that allow much larger dimensions, require less communication, require less memory, and can efficiently permit lower-accuracy results [89]. An exciting and fast-emerging framework for constructing such algorithms is to introduce probabilistic elements. This has given rise to the field of randomized numerical linear algebra, which has enabled computation on a scale beyond the reach of traditional NLA [181].

Avron, Maymounkov, and Toledo (2010) claim (quite reasonably) in the widely influential ‘Blendenpik’ paper [14] that “randomization is arguably the most exciting and innovative idea to have hit linear algebra in a long time.” It offers a tool to tackle fundamental NLA problems in modern computational settings. Throughout this thesis we will demonstrate the relevance of randomized algorithms in this era of computing with extreme data. We specifically focus on themes in randomized NLA related to the implementation, accuracy, and reliability of randomized algorithms, such as parameter selection and numerical stability. We argue that these are important, and sometimes understated, elements that are crucial to the further development of the field.

This thesis is specifically concerned with designing and analysing randomized algorithms for two fundamental matrix problems: numerical rank estimation and linear least squares. In this introduction, we first discuss these fundamental problems and then introduce randomized NLA through a discussion on the emergence of the field, paying special attention to algorithms for these problems.

1.1 Two fundamental matrix problems

Randomized algorithms have already demonstrated the ability to accelerate classical algorithms considerably [181]. Two of the biggest successes are the development of algorithms for low-rank approximation and linear least squares problems, and these methods have greatly influenced the development of the field [76, 174]. In this section, we introduce these problems, classical approaches to solving them, and the limitations of classical algorithms. In the following sections, we then discuss how randomization can improve upon the classical approaches.

1.1.1 Low-rank approximation and numerical rank

We first consider low-rank matrices, low-rank approximation, and numerical rank estimation. It is within the context of low-rank approximation that numerical rank estimation is especially important. A wide variety of problem settings in data analysis [8, 231], machine learning [110, 228], scientific computing [116, 253], and signal processing [71, 73] — to name a few domains — produce matrices with low numerical rank. These matrices have intrinsically low-dimensional representations; the dimensionality of the information they contain is smaller than the dimension of the data. One may exploit this low-rank structure for further computations, or the low-rank factors may contain relevant information about the data itself, as in signal processing and principal component analysis (PCA).

The numerical rank of a matrix is determined by its singular values. Let $A \in \mathbb{C}^{m \times n}$ ($m \geq n$) have a singular value decomposition (SVD) $A = U\Sigma V^*$, where $U \in \mathbb{C}^{m \times n}$ and $V \in \mathbb{C}^{n \times n}$ have orthonormal columns and $\Sigma \in \mathbb{R}^{n \times n}$ is a square diagonal matrix with singular values $\sigma_1 \geq \dots \geq$

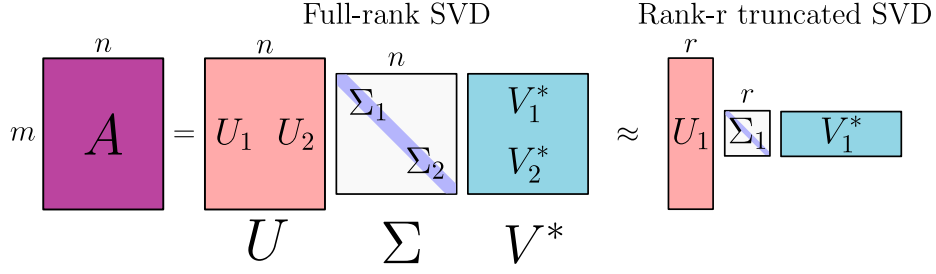


Figure 1.1: The full-rank and rank- r truncated SVDs. The singular values $\sigma_1 \geq \dots \sigma_n \geq 0$ are ordered in magnitude, and Σ_1 contains the r largest singular values.

$\sigma_n \geq 0$. For a given tolerance $\varepsilon > 0$, we define the (relative) numerical rank to be the integer i such that $\sigma_i(A) > \varepsilon \|A\|_2 \geq \sigma_{i+1}(A)$ (as in [29]).

The problem of low-rank approximation is to find a matrix $\hat{A}$ of size $m \times n$ with a specified (maximal) rank $r < \min\{m, n\}$ such that the quantity $\|A - \hat{A}\|$ is minimal in some norm (often the spectral or the Frobenius norm). The approximation will generally be the product of matrices of which at least one has a dimension equal to r . Such a low-rank decomposition will result in lower storage cost, and when computing with the approximation instead of the full-rank matrix, smaller communication and arithmetic cost.

Many full-rank matrix decompositions are well-known, such as the QR factorization, the eigenvalue decomposition, or the SVD. Some of these decompositions naturally lead to low-rank approximations, for instance by truncating an SVD [114]. Split the factors of the full-rank SVD up in a dominant and a trailing part as in Figure 1.1. The rank- r truncated SVD A_r is given as a product of the dominant parts $A_r = U_1 \Sigma_1 V_1^*$, where $U_1 \in \mathbb{C}^{m \times r}$, $\Sigma_1 \in \mathbb{R}^{r \times r}$, and $V_1 \in \mathbb{C}^{n \times r}$.

The truncated SVD is the optimal solution to the low-rank approximation problem in the spectral and Frobenius norm. By the Eckart-Young Theorem we have in any unitarily invariant norm [114]

$$\min_{\text{rank}(\hat{A}) \leq r} \|A - \hat{A}\| = \|A - A_r\| = \begin{cases} \sigma_{r+1} & \text{if } \|\cdot\| = \|\cdot\|_2, \\ \sqrt{\sum_{k=r+1}^n \sigma_k^2} & \text{if } \|\cdot\| = \|\cdot\|_F. \end{cases} \quad (1.1)$$

The quality of the approximation depends on the next singular value σ_{r+1} (for the spectral norm) or the tail of the singular values (for the Frobenius norm). The optimality guarantee, interpretability of the decomposition, and availability of a very efficient algorithm make the SVD a great choice for decomposing moderately-sized data. Computing a full SVD costs $\mathcal{O}(mn^2)$ flops for an $m \times n$ matrix with $m \geq n$. One can compute a rank- r truncated SVD in $\mathcal{O}(mnr)$ time [174]. Any algorithm that aims to improve upon the (asymptotic) computational cost would have to run in sub-cubic complexity.

We note that one may write the truncated SVD as

$$A_r = U_1 U_1^* A = A V_1 V_1^*.$$

This is a fundamental fact that many of the randomized algorithms for low-rank approximation we discuss here exploit. If one has an approximation $\hat{U}_1$ to the dominant column space U_1 or an approximation $\hat{V}_1$ to the dominant row space, one naturally has a low-rank approximation to the truncated SVD $\hat{A}_r = \hat{U}_1(\hat{U}_1^* A)$ or $\hat{A}_r = (A\hat{V}_1)\hat{V}_1^*$, i.e. by projecting A onto the approximate dominant spaces.

We distinguish between two types of low-rank approximation problems: fixed-rank and fixed-precision. The fixed-rank problem is to find a matrix $\hat{A}_r$ of specified rank r so that $\|A - \hat{A}_r\|$ is small. The fixed-precision problem is to find a low-rank matrix $\hat{A}$ so that $\|A - \hat{A}\| \leq \varepsilon$ for some specified tolerance ε . Considering (1.1), if we know the numerical rank of a matrix for a specified tolerance, we know (approximately) what rank our approximation should be to attain that same tolerance. Similarly, with the numerical rank in hand one can decide whether a fixed-rank approximation leads to a reasonable error given its rank. The widespread nature of low-rank matrices in a variety of computational sciences make (numerical) rank estimation an important problem.

1.1.2 Linear least squares

We turn to a fundamental problem in numerical linear algebra; solving linear systems. We focus our attention on the overdetermined linear least squares problem, which is defined as

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2, \quad A \in \mathbb{R}^{m \times n}, \quad b \in \mathbb{R}^m, \quad (1.2)$$

where $m > n$. Problems of this form are ubiquitous in applications, such as medical imaging, image deblurring, or linear regression [76, 130]. The LS problem (1.2) has a unique solution provided that A is of full column rank. This unique minimizer is denoted by x^* and equal to the pseudo-inverse of A applied to b , that is $x^* = A^\dagger b$.¹ If b happens to lay in the range of A , the minimizer x^* results in a zero residual. More often, we cannot solve $Ax = b$ exactly and compute minimizers that have some residual $\|r\|_2 = \|Ax - b\|_2 > 0$. If A does not have full column rank, the minimizer to (1.2) is not unique. The vector $x^* = A^\dagger b$ is still a solution, and it is the solution with the smallest spectral norm [140].

We will also consider the underdetermined least squares problem in this thesis. For $A \in \mathbb{R}^{m \times n}$ with $m < n$, the problem $Ax = b$ can be solved with zero residual provided that A has full row

¹The pseudo-inverse of a matrix A can be found using the (economy-sized) SVD of A . If $A = U\Sigma V^T$ is an SVD of A with Σ a square matrix of dimension $\text{rank}(A)$, then $A^\dagger = V\Sigma^{-1}U^T$ is the pseudo-inverse of A .

rank. Following [140], we let the unique least squares solution to the overdetermined problem be the solution that minimizes the spectral norm. Once again, this is given by $x^* = A^\dagger b$. The underdetermined problem will be revisited in Chapter 6. Throughout the thesis, we focus primarily on overdetermined problems.

There is a multitude of algorithms available to solve (1.2). For direct methods, all of these cost $\mathcal{O}(mn \min(m, n))$ operations. Among the most prominent is using the QR decomposition of $A = QR$ [250]. The minimizer to (1.2) is then computed as $x^* = R^{-1}Q^T b$. This method is numerically stable [140], which roughly means that it computes a solution as good as one can hope to compute in finite precision.² An alternative direct method for the overdetermined problem is to solve the $(n \times n)$ -dimensional normal equations $A^T A x = A^T b$ using, for instance, a Cholesky decomposition. Despite the equivalence in exact arithmetic, introducing the normal equations and specifically the Gram matrix $A^T A$, deteriorates the conditioning of the problem [140]. As a result, it may not be possible to compute accurate solutions.

To exploit sparsity or otherwise cheap matrix-vector products, reduce storage cost, or in large-scale settings, we may prefer iterative methods to direct methods [250]. Such methods compute iterates that (hopefully) get increasingly close to a sufficiently good solution. Throughout this thesis, we focus primarily on Krylov subspace methods. At each iteration k we search for an approximate solution in a growing subspace of $\mathbb{R}^n$. This subspace is the k -dimensional Krylov subspace associated to the normal equations

$$\mathcal{K}_k(A^T A, A^T b) = \text{span} \{A^T b, (A^T A)A^T b, \dots, (A^T A)^{k-1}A^T b\}.$$

The iterative Krylov subspace solver LSQR [205], for instance, computes iterate x_k at iteration k

$$x_k = \underset{x \in \mathcal{K}(A^T A, A^T b)}{\text{argmin}} \|Ax - b\|_2. \quad (1.3)$$

The dominant computational cost of such algorithms is matrix-vector products with A and A^* in each iteration, which is $\mathcal{O}(mn)$ for dense matrices and $\mathcal{O}(\text{nnz}(A))$ for sparse matrices, where $\text{nnz}(A)$ is the number of nonzeros of A . The number of iterations necessary depends on the desired accuracy and the conditioning of the problem. The conditioning of the problem refers to the width of the singular value spectrum of A . We say the ratio of the largest and smallest singular value is the condition number $\kappa_2(A) = \sigma_1(A)/\sigma_n(A)$. If the condition number is close to 1, Krylov subspace methods converge very quickly [224]. For this reason, these methods are usually competitive when A is large, sparse, and well-conditioned.

²Throughout this thesis, we use the term numerical stability to refer to the backward stability of an algorithm. This is precisely defined in Section 4.2.4.

If A is ill-conditioned, we can try to find a change of variables $P \in \mathbb{R}^{n \times n}$ that clusters the spectrum of A in the sense that $\kappa_2(AP)$ is smaller than $\kappa_2(A)$. We then iteratively solve the *preconditioned* problem

$$\min_{y \in \mathbb{R}^n} \|APy - b\|_2, \quad x = Py, \quad (1.4)$$

where P is referred to as a preconditioner. If A is well-conditioned or if we have access to a good preconditioner, iterative solvers can converge in less than $\mathcal{O}(mn^2)$ time.

It is not immediately obvious what makes a good numerical solution to (1.2). Taking into account the approximate nature of computing, a computed solution $\hat{x}$ will always be an approximation to x^* . We might desire the computed solution to attain optimal residual $\|A\hat{x} - b\| = \|Ax^* - b\|$, minimize the (forward) error $\|\hat{x} - x^*\|$, or satisfy $A^T A\hat{x} = A^T b$. These are sometimes competing wishes, especially when A is ill-conditioned (i.e. has large condition number). Ill-conditioning combined with very small perturbations in A or b , such as rounding errors, can result in a very large error (see Figure 1.2). As numerical analysts, we might aim for backward stable solutions, which means the computed solution solves a nearby problem. Practitioners, especially in an inverse problem context, might be more interested in minimizing the error [130]. They must turn to regularization.

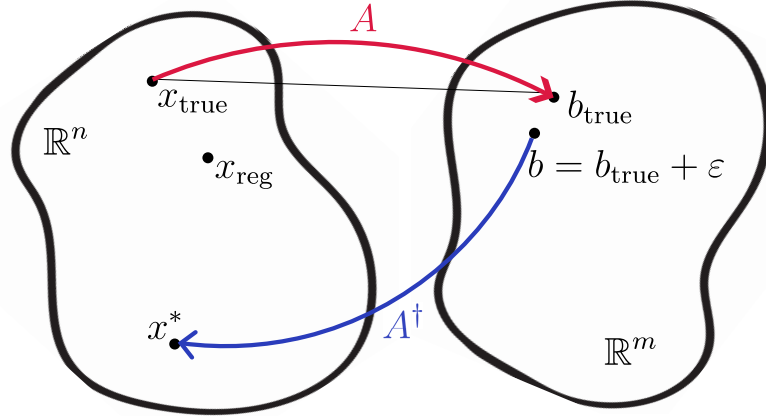


Figure 1.2: If A is very ill-conditioned, the (numerically stable) computed solution $\hat{x} = A^\dagger b$ might be very different from x_{true} . Here, we have a ground truth linear system $Ax_{\text{true}} = b_{\text{true}}$, but we observe slightly noisy data $b = b_{\text{true}} + \varepsilon$. Solving the LS problem $x^* = \operatorname{argmin} \|Ax - b\|_2$ results in a large error $\|x^* - x_{\text{true}}\|_2$. If we wish to minimise the error, it is necessary to regularize. This figure is inspired by [130, Fig. 4.1].

1.1.3 Regularized least squares problems

There are various contexts in which regularization is necessary to (numerically) obtain a reasonable solution to the least squares problem. In particular, if $\min \|Ax - b\|$ does not have a unique solution (either because A is tall and rank-deficient or because the problem is underdetermined), regularization is a necessary tool to transform the problem into a well-posed one. Even if the problem

is full-rank but A is very ill-conditioned, regularization can aid in minimizing the error in $\hat{x}$. We consider the Tikhonov regularized least squares problem, which has the form

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2 + \lambda \|x\|_2^2, \quad (1.5)$$

for some regularization parameter $\lambda > 0$. The regularization term $\lambda \|x\|_2^2$ serves to ‘smoothe’ the computed solution by improving the overall conditioning of the problem. Figure 1.2 shows that a regularized solution x_{reg} is likely much closer to x_{true} than $A^\dagger b$. Fundamentally, we are balancing goodness-of-fit (small $\|Ax - b\|_2$) with regularity (small $\|x\|_2$). The parameter λ controls this balance, and selecting an appropriate value is a non-trivial task [122, 130, 132]. From a linear algebraic point of view, including the regularization term makes (1.5) a full-rank overdetermined LS problem with a unique solution.

This can be seen by rephrasing (1.5) into standard LS formulation. The problem (1.5) is equivalent to the minimization problem

$$\min_{x \in \mathbb{R}^n} \left\| \begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix} x - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|_2^2. \quad (1.6)$$

We can apply standard LS solvers on this reformulated problem, but ideally one exploits the structure present in the problem. An SVD of A is sufficient to solve (1.6) for any λ , yet incurs $\mathcal{O}(mn \min(m, n))$ cost. Iterative methods for the regularized LS problem are discussed in detail in Chapter 6.

1.2 The origins of randomized numerical linear algebra

We will discuss randomized NLA through a discussion on randomized algorithms for the before-mentioned problems. More technical detail on the algorithms and theory stated informally here is presented in Chapter 2. We start with low-rank matrix approximation, which kick-started two decades of research on randomized algorithms and is still a very active area of research.

1.2.1 Subsampling era

The first algorithms that resemble modern randomized NLA were designed for approximate low-rank decompositions. As mentioned in Section 1.1.1, an approximation to the SVD of a matrix A can be obtained if one has an approximation to the dominant column or row space of A . Frieze, Kannan, and Vempala (1998) were the first to suggest randomly sampling rows from A to create such an approximate row space [104]. This process is graphically displayed (in simplified form) in Figure 1.3. The authors show that if the subsampling probabilities are proportional to the norms of the rows

of A , then there exists a good approximate basis for V_1 within the sampled submatrix (when more rows are sampled than the rank of V_1). This submatrix is now referred to as a *sketch* of A , because, although it is much smaller, we hope it ‘sketches’ a dominant space of A . The authors name it a Monte Carlo algorithm for low-rank approximation.

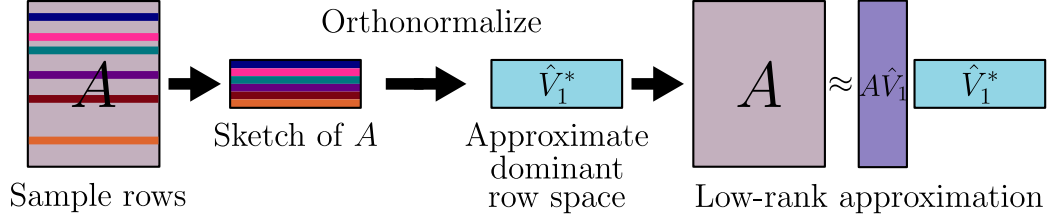


Figure 1.3: A simplified version of the ‘Monte Carlo’ algorithm for low-rank approximation from [104].

Drineas, Kannan, and Mahoney (2006) built on this work and suggested a sophisticated algorithm for sampling important columns to obtain a good ‘sketch’ [85]. The column-based low-rank approximation is similar to the process in Figure 1.3, yet we select columns that should contain a basis for U_1 . The approximation is then given by $\hat{A}_r = U_1(U_1^* A)$. A variety of algorithms where subsampling rows or columns is the basis of SVD-based low-rank matrix approximation algorithms followed [79, 80, 85, 91]. Error bounds for these algorithms are of the form

$$\|A - \hat{A}_r\|^2 \leq \|A - A_r\|^2 + \text{poly}(r, 1/s) \|A\|_F^2, \quad (1.7)$$

where $\hat{A}_r$ is the resulting rank- r approximation, A_r is the optimal rank- r approximation, and s is the number of sampled columns or rows. Note the final term, the Frobenius norm $\|A\|_F^2$, might be very large despite the low-rank structure of A .

The low-rank approximation algorithms highlight a fundamental idea in almost all randomized algorithms that followed: reducing the size of the matrix yet retaining most of the relevant information using a random procedure. In the previous examples the authors used random sampling. An early example of a different kind of dimension reduction is developed in Papadimitriou, Raghavan, Tamaki, and Vempala (1998). They are also concerned with low-rank approximation and obtaining an approximate dominant row-space for that purpose. Instead of sampling rows, however, they propose premultiplying A with a small random matrix $S \in \mathbb{R}^{s \times m}$, $s \ll m, n$, with orthonormal rows to obtain a different kind of sketch SA [207].

The procedure of randomized dimension reduction is now known as ‘sketching’. Despite the work in [207], the focus of sketching procedures remained on sampling techniques in the early 2000s with algorithms for SVD-like low-rank approximations [79, 209], other low-rank decompositions such as the CUR decomposition [86, 92], and different matrix problems [84].

The early consensus on sampling was that uniform sampling of rows or columns is inadequate to obtain approximations with small errors [23]. Instead, sampling probabilities were sometimes

based on Euclidean row or column norms. That is, if one were to sample rows from a large matrix, the probability of sampling a particular row would be proportional to its row norm squared. The underlying idea is that such rows are, in some sense, ‘more important’ to the geometry of the matrix. Regardless of this slightly more sophisticated sampling technique, randomized procedures produced error bounds of the form (1.7), with a bothersome $\|A\|_F$ term.

Drineas, Mahoney, and Muthukrishnan (2006) were the first to extend the idea of random sampling to the overdetermined least squares (LS) problem

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|, \quad A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m, m \gg n,$$

and simultaneously further developed sampling algorithms [90]. They proposed sampling based on the row-norms of an orthogonal basis for A , instead of A itself. That is, say $A = U\Sigma V^T$ is an (economy-sized) SVD of A , then the probability of sampling row i of A is proportional to the squared row-norm of the i th row of U . The squared row-norms of U are referred to as leverage scores (see Definition 2.2 for a formal definition), and provide a much better representation of the ‘importance’ of a row than the row-norms of A directly. In particular, the i th leverage score is a measure for the leverage or influence that the i th row of A has on the solution x^* to the LS problem (see [87] and [234] for further details).

After having sampled rows from A and b according to leverage scores, one can solve the (much smaller) subsampled problem exactly using a classical approach. Let $S \in \mathbb{R}^{s \times m}$ denote a sampling operator: a short matrix consisting of rows of the identity matrix corresponding to rows of A that are to be sampled. The subsampled (or sketched) least squares problem is then

$$\hat{x} = \operatorname{argmin}_{x \in \mathbb{R}^n} \|S(Ax - b)\|_2. \quad (1.8)$$

Since $m \gg s \geq n$, solving (1.8) is much cheaper than solving the full-dimensional problem. This is the first work where it was shown that this randomized approach leads to good relative error approximations in the sense that, with high probability

$$\|A\hat{x} - b\|_2 \leq (1 + \varepsilon) \min \|Ax - b\|_2, \quad (1.9)$$

where ε is typically a number not smaller than, say, 0.1 [76]. The issue, however, is that sampling probabilities based on an orthogonal decomposition of A require such a decomposition, incurring $\mathcal{O}(mn^2)$ cost. Does there exist a randomized dimension reduction technique that is cheaper than $\mathcal{O}(mn^2)$ but still about as effective in attaining relative error approximations?

1.2.2 Random embeddings: fast relative error approximations

In a seminal work discussing a breadth of matrix problems, Sarlós (2006) answered this affirmatively and laid the foundation for randomized algorithms to move past sampling [229]. He proposed randomized algorithms for matrix multiplication, least squares problems, and approximate singular value decompositions; obtaining accurate approximations that improve on the asymptotic computational cost of classical algorithms. The foundation of the work was a different kind of randomized dimension reduction matrix, inspired by the fast Johnson–Lindenstrauss transform embedding matrix introduced by Ailon and Chazelle (2006).

The Johnson-Lindenstrauss Lemma [152] that n vectors in $\mathbb{R}^m$ can be embedded into $\mathbb{R}^s$ space such that the distance between the vectors are preserved up to a $1 + \varepsilon$ factor, provided that $s = \mathcal{O}(\log(n)/\varepsilon^2)$ [229]. In particular, there exist random, linear maps that will achieve this with high probability. Consider for instance a Gaussian matrix $S \in \mathbb{R}^{s \times m}$ with $\mathcal{N}(0, s^{-1})$ identically and independently distributed (i.i.d.) entries, where $s = \mathcal{O}(\log(n) \log(1/\delta)/\varepsilon^2)$. Then, for a set $\{a_1, \dots, a_n\} \subset \mathbb{R}^m$, with probability at least $1 - \delta$

$$(1 - \varepsilon)\|a_i - a_j\| \leq \|S(a_i - a_j)\| \leq (1 + \varepsilon)\|a_i - a_j\|, \quad \text{for all } 1 \leq i, j \leq n.$$

Sarlós’ seminal contribution was extending this idea to matrix problems. Random matrices preserve the geometry of a set, so can we also use them to preserve the geometry of a matrix? We call preserving matrix geometry in lower dimensions *subspace embedding*:

$$(1 - \varepsilon)\|Ax\|_2 \leq \|SAx\|_2 \leq (1 + \varepsilon)\|Ax\|_2, \quad \text{for all } x \in \mathbb{R}^n. \quad (1.10)$$

Intuitively, the dimensionality of a whole subspace is ‘larger’ than of a finite set, so we need a larger random matrix to achieve (1.10). For a Gaussian matrix, this embedding dimension is $s = \mathcal{O}(n \log(1/\delta)/\varepsilon^2)$ for a $m \times n$ matrix of full column rank, and more generally $s = \mathcal{O}(\text{rank}(A) \log(1/\delta)/\varepsilon^2)$ (see [275] for a proof). The subspace embedding property (1.10) is very useful for matrix algorithms. Similarly to subsampling, it achieves a randomized dimension reduction but does so more efficiently.

A Gaussian matrix, however, costs $\mathcal{O}(mns)$ operations just to apply to A . If we set out to improve the asymptotic cost of low-rank approximation or LS algorithms, this is the cubic cost we wish avoid. The fast Johnson-Lindenstrauss embeddings in [4] were revolutionary because they can be applied in $\mathcal{O}(mn \log(s))$ time. This is possible due to the usage of an FFT-type procedure. The embedding proposed in [229] has the form $S = GHD$, where D is a diagonal matrix of random signs, H is a orthogonal Hadamard matrix, and $G \in \mathbb{R}^{s \times m}$ is a sparse matrix with non-zeros distributed as normal random variables. Such a matrix satisfies the subspace embedding property (1.10) with high probability when $s = \mathcal{O}(\text{rank}(A) \log(\text{rank}(A))/\varepsilon^2)$ (we suppress the $\mathcal{O}(\log(1/\delta))$ dependency

on the failure probability δ throughout). The fast random embedding comes with a slightly larger dimension than in the Gaussian case, but with improved computational cost.

Many different types of random embedding matrices now exist and have been studied in-depth, both theoretically and practically. We discuss this in Section 2.1.3. Fundamentally, a random embedding for $A \in \mathbb{R}^{m \times n}$ is a matrix $S \in \mathbb{R}^{s \times m}$ for which (1.10) holds, and it is fast if it can be applied quicker than in $\mathcal{O}(mns)$ time. We distinguish between oblivious and non-oblivious embedding matrices; oblivious embeddings are constructed without prior knowledge about the matrix A , whereas non-oblivious embeddings use properties of A (e.g. leverage score sampling) [181]. Applying a random embedding is often referred to as sketching, and the resulting matrix SA as a sketch of A .

Sarlós (2006) showed that by applying such a fast random embedding one can obtain relative error bounds on the approximate SVD that do not include the $\|A\|_F$ -term as in (1.7). Draw a random matrix $S \in \mathbb{R}^{s \times m}$ from the distribution of fast random embeddings discussed before, with $s = \mathcal{O}(r/\varepsilon + r \log(r))$, and sketch the column space of A by computing AS^T . Compute an orthogonal matrix $\hat{U}_1$ whose columns span the column space of AS^T . If we exactly compute the rank- r truncated SVD of $\hat{U}_1(\hat{U}_1^T A)$ denoted $\hat{A}_r = (\hat{U}_1 \hat{U}_1^T A)_r$, then, with probability at least $1/2$

$$\|A - \hat{A}_r\|_F \leq (1 + \varepsilon)\|A - A_r\|_F. \quad (1.11)$$

For dense matrices, the asymptotic cost is still $\mathcal{O}(mnr)$ due to the matrix-matrix multiplication $\hat{U}_1 A$. However, this procedure and error bound much resembles modern state-of-the-art randomized SVD algorithms.

Sarlós [229] obtained a similar result for least squares problems. If one embeds a (full rank) $m \times n$ least squares problem using a random embedding S of dimension $s = \mathcal{O}(n \log(n)/\varepsilon)$, then, with probability at least $1/3$, $\hat{x} = \operatorname{argmin} \|S(Ax - b)\|_2$ satisfies $\|A\hat{x} - b\|_2 \leq (1 + \varepsilon) \min \|Ax - b\|_2$. This is a very fast way to obtain an approximate solution that attains a residual within a small multiple of the optimal residual! We now refer to this procedure as a sketch-and-solve algorithm (see Section 2.3.1).

A similar result for a general random embedding is presented in Lemma 2.4. For an embedding matrix $S \in \mathbb{R}^{s \times m}$ that is Gaussian with $s = \mathcal{O}(\operatorname{rank}(A)\varepsilon^{-2})$ or a fast embedding with $s = \mathcal{O}(\operatorname{rank}(A) \log(\operatorname{rank}(A))\varepsilon^{-2})$, we have

$$\hat{x} = \operatorname{argmin}_{x \in \mathbb{R}^n} \|S(Ax - b)\|_2 : \quad \|A\hat{x} - b\| \leq \frac{1 + \varepsilon}{1 - \varepsilon} \|Ax^* - b\|_2,$$

with high probability. As $s \ll m$ is needed to ensure solving the $s \times n$ LS problem deterministically is computationally beneficial, we cannot make ε very small. Indeed, usual relative errors for this ‘sketch-and-solve’ approach would have $\varepsilon \geq 0.1$.

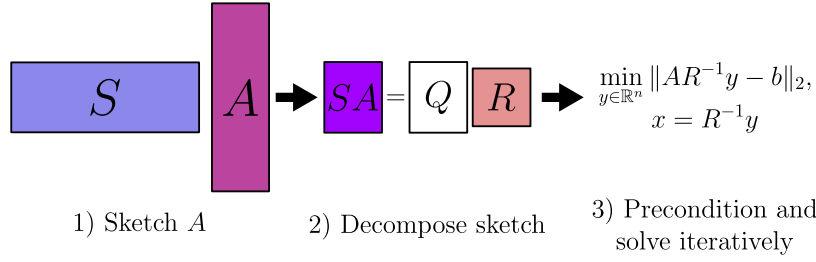


Figure 1.4: A sketch-and-precondition-type algorithm. Rokhlin and Tygert [222] specifically performed final step with the preconditioned conjugate gradient algorithm.

1.2.3 Sketch-and-precondition for highly accurate results

Another revolution in randomized numerical linear algebra occurred when Rokhlin and Tygert (2008) was published. This work showed that random embedding approaches in NLA do not just lead to rapid approximations, but can also result in faster high-quality computations. They introduced what would later be known as the sketch-and-precondition framework for solving LS problems. Instead of reducing the dimension of the entire problem to solve it in a lower-dimensional space, as in [90, 229], the authors suggest to use the random embedding to find a good preconditioner.

In particular, the tall matrix $A \in \mathbb{R}^{m \times n}$ with $m \gg n$ is sketched to a smaller matrix $SA \in \mathbb{R}^{s \times n}$ with a random embedding S . One then computes a QR decomposition of $SA = QR$ and uses $P = R^{-1}$ as a preconditioner (see (1.4)) in an iterative solver for the original LS system. This process is displayed in Figure 1.4.

The sketch-and-precondition algorithm requires $\mathcal{O}(mn \log(n) + n^3 \log(n))$ cost for embedding A and computing the preconditioner. The condition number of AR^{-1} will be close to 1, which means an iterative Krylov subspace solver converges in $\mathcal{O}(\log(1/\varepsilon))$ iterations for ε accuracy [222]. The change from $\mathcal{O}(n \log(n) \varepsilon^{-2})$ dependency on the accuracy ε for sketch-and-solve to $\mathcal{O}(mn \log(\varepsilon^{-1}))$ dependency for sketch-and-precondition meant the LS problem could be solved to ε relative precision for values of ε close to machine precision, as opposed to in the sketch-and-solve framework (1.9). Using randomization, one could now obtain solutions with similar accuracy guarantees as direct solvers, but while reducing the computational and communication cost.

The ideas presented here on using randomized algorithms for approximate truncated SVDs, approximate solutions to LS problems, and preconditioning LS problems have been highly influential to the future of the field. Much of the current work is still connected to these fundamental ideas.

1.3 Randomized numerical linear algebra matures

We next describe how these new and exciting ideas — originally mostly written in a theoretical computer science (TCS) context — made their way to numerical linear algebra and broader scientific computing communities, where algorithms were seeing practical applications.

1.3.1 Low-rank approximations

In a sequence of papers, approximate low-rank SVD's and other low-rank decompositions were further developed [126,167,179,221,276]. Liberty, Woolfe, Martinsson, Rokhlin, and Tygert (2007) were the first to use the random embedding to improve the computational complexity of an approximate SVD algorithm for dense matrices [167]. One of these variants uses an interpolative decomposition (ID). If $A \in \mathbb{R}^{m \times n}$ has exact rank $r < \min(m, n)$, then it can be decomposed as $A = A_{(:,J)}Z$, where $A_{(:,J)} \in \mathbb{R}^{m \times r}$ is a subset of columns of A indexed by J and $Z \in \mathbb{R}^{r \times n}$ has some columns from the identity matrix and no entry greater than 2 in absolute value. An approximate ID can be defined in the same way, but with $A \approx A_{(:,J)}Z$, and can be computed in $\mathcal{O}(rmn \log(n))$ operations [276]. The low-rank approximation algorithm suggested in [167] is based on computing the approximate rank- r ID of a sketch $SA = (SA)_{(:,J)}Z$, where $S \in \mathbb{R}^{s \times n}$ is a fast random embedding with $s > r$ (but not much larger). A low-rank approximation can then be defined as $\hat{A}_r = A_{(:,J)}Z$, which has cost just $\mathcal{O}(mn \log(s) + rsn \log(n))$ operations. In this case, the sketch serves to identify important columns of A . An approximate SVD of A can then be obtained at an additional $\mathcal{O}((m+n)r^2)$ cost by computing the QR of $Z^* = QR$ and the SVD of $A_{(:,J)}R^*$. Woolfe, Liberty, Rokhlin, and Tygert (2008) suggest similar algorithms shortly after.

Martinsson, Rokhlin, and Tygert (2006) used Gaussian matrices for the choice of embedding [178]. Although this approach does not improve computational complexity relative to deterministic approaches, it was influential because the authors showed — with theory and experiments — how well the algorithms performed in terms of errors and failure probability. Rokhlin, Szlam, and Tygert (2010) build on this work, showing that a much better approximation to the dominant column space can be obtained by using a power iteration with A [221]. Specifically, instead of sketching A with a random embedding $S \in \mathbb{R}^{d \times n}$, we form the sketch $(AA^T)^q AS^T$ for a small integer q . Martinsson, Szlam, and Tygert (2011) improve on the stability of this scheme [180].

The survey paper by Halko, Martinsson, and Tropp (2011) brought randomized algorithms for approximate matrix decompositions to the attention of a mainstream NLA audience. They showed with extensive theory and numerical experiments that the approximate truncated SVD approach with random embeddings, which is now coined the randomized SVD, provides significant speed-ups relative to classical algorithms. They present an algorithm that computes a high-quality approximation to A_r in $\mathcal{O}(mn \log(r))$ flops using the ideas before combined with an interpolative decomposition. The low-rank matrix approximation algorithms presented there came with implementation suggestions, posterior error estimations, and original error analysis. This was somewhat different from the majority of the prior work, where the focus was theoretical rather than practical. Randomized algorithms were thus shown to have the potential to be valuable alternatives to classical algorithms.

1.3.2 Least squares problems

Similar in significance to the NLA community, Avron, Maymounkov, and Toledo (2010) presented a high-performance implementation of the sketch-and-precondition framework suggested in [222]. They discuss the algorithm in detail, provide implementation details and a software package, and most importantly, show that their randomized algorithm outperforms LAPACK’s highly-optimized classical LS solver [14].

Similarly, Meng, Saunders, and Mahoney (2014) developed an algorithm they called LSRN (least squares random normal) based on embedding with a Gaussian matrix [187]. The work had much more of a linear algebra flavour than [222], suggesting adaptations for underdetermined, rank-deficient, and regularized problems as (1.5). In particular, LSRN uses the SVD of the sketch SA and therefore can account for rank-deficiency by truncating (see also Section 4.5.1). Regardless of the rank of A , it is shown that LSRN converges to the minimum-norm solution. Although using a Gaussian embedding comes with $\mathcal{O}(mn^2)$ cost for a dense matrix-matrix multiplication, its implementation is considerably more straightforward than for fast embeddings. Additionally, Gaussian random matrices can be fast embeddings when A is sparse and, even for dense matrices, the process of applying them is “embarrassingly parallel” [187, p. 2]. LSRN can be implemented in just a few lines of code, bringing the sketch-and-precondition framework further into the mainstream.

These contributions further embedded the idea that randomized algorithms have great practical value. The sketch-and-precondition framework was extended to different contexts, such as kernel ridge regression [12], constrained least squares [265, 266], and various data-specific contexts [78, 241]. It is also one of the main inspirations for this thesis.

1.3.3 Changing computational context

The usefulness of randomized algorithms was underlined due to their applicability in difficult computational settings [174]. At the same time, such contexts also strongly influenced the direction of the development of the field. The algorithms discussed so far improve computational cost both theoretically and practically, and are very suitable for large-scale problems. Note as well that, especially for the low-rank approximation algorithms, we need few passes over the matrix A , also improving communication cost. Randomized algorithms have been developed for challenging data settings, such as data only being available through matrix-vector multiplications, data too large to be stored in fast memory, or data that may not be stored at all.

Clarkson and Woodruff (2009) suggest a one-pass algorithm for low-rank approximation, which means the matrix A only has to be accessed once. This is particularly useful for a streaming (turnstile) context: the data is presented as a stream of linear updates that cannot be stored. One

of the approximations they suggest is of the form

$$\hat{A} = AS_1^T(S_2AS_1^T)^\dagger S_2A, \quad (1.12)$$

where S_1 and S_2 are random embeddings of conforming size. We now refer to this as generalized Nyström. This approximation is further analysed in the extensive randomized NLA review paper by Woodruff (2014). However, these works on generalized Nyström do not include any computational experiments or specify implementation suggestions.

On the other hand, Tropp, Yurtsever, Udell, and Cevher (2017, 2019) focus on developing user-friendly algorithms and provide extensive numerical experiments [252, 253]. They consider further computational contexts, such as when the columns of data are revealed one-by-one and a new approximation has to be computed on the reveal of a new column, or when data is presented through a sum of ordered updates. The approximation they consider is in the same form as (1.12), but with orthogonalization steps that improve the numerical stability. This is one of the first algorithms explicitly considering finite-precision behaviour in algorithm design, and was later improved on in [196]. Similarly, Halko et al. (2011) discuss a randomized low-rank algorithm for data stored out-of-core [126].

Works like these mark the development of randomized NLA away from TCS towards a NLA focus.

1.4 Modern randomized NLA and this thesis' contributions

Dereziński and Mahoney (2024) write that “*the area of RandNLA (randomized NLA) has achieved a certain level of maturity*” [76]. We now have a multitude of randomized algorithms available for fundamental matrix tasks. Computational experiments in a range of contexts confirm competitiveness and speed-ups relative to classical deterministic algorithms. Sketching is particularly useful in a streaming context and when communication cost is high, as it usually means performing classical linear algebraic tasks in a lower dimension. Randomized algorithms have developed for a variety of fundamental linear algebra tasks such as approximating matrix functions [69, 125, 213], trace estimation [15, 68, 189, 225], CUR matrix decompositions [38, 83, 269, 279], kernel matrix approximation [6, 55, 165], operator learning [35, 212], square linear systems [20, 21, 77, 78, 197], optimization [43, 48–50, 160, 215], model order reduction [7, 16, 40, 218], and dynamic mode decomposition [3, 75, 102]. We refer to the surveys Mahoney (2011), Halko, Martinsson, and Tropp (2011), Woodruff (2014), Martinsson and Tropp (2020), Murray et al. (2023), and the references therein [127, 174, 181, 192, 275].

There are increasingly many efforts to bring randomized algorithms into a high-performance computing world. Various algorithms have been implemented in, for instance, the NAG library,

MATLAB, and NVIDIA’s cuSOLVER [192]. Increasingly many papers are accompanied by software libraries, such as Blendenpik [14], LSRN [187], Ski-LLS [47], pylspack [235], and LibSkylark [156]. Even standard software libraries RandBLAS and RandLAPACK optimized for randomized NLA are under development [192]. Various recent papers discuss online or streaming contexts [39, 151, 164, 236, 242, 253], or design algorithms in a distributed or low-memory environment [19, 105, 109, 151, 194, 282].

These efforts all show a development in the direction of mainstream computational science. However, the young field still faces many challenges. We discuss some of these challenges and then introduce the aims of this thesis.

1.4.1 Challenges in randomized NLA

Already in 2017, Tropp, Yurtsever, Udell, and Cevher [252] noted on low-rank approximation using random embeddings in complicated computational contexts “(...) *how do we implement these methods? Are they numerically stable? How should algorithm parameters depend on the input matrix? What is the right way to preserve structural properties? (...) These questions have often been neglected in the literature*” [252]. We agree with this point of view, and believe questions surrounding reliability, numerical stability, and user-friendliness need to be taken seriously for randomized algorithms to have a bigger impact on the wider scientific community. In this thesis, we specifically focus on questions on parameter selection in low-rank approximation algorithms, on ways in which randomized NLA can aid in parameter selection for regularized problems, and the reliability of randomized algorithms for LS problems in finite precision.

1.4.1.1 Parameter selection

An efficient way to set parameters is important to the use of algorithms in real-world applications. Many algorithms for low-rank approximation require the target-rank of the approximation as input [127, 196]. However, although there might be an upper bound on the rank available in terms of memory, practitioners might be more interested in an approximation being of particular quality. For instance, there may be a user-specified tolerance ε so that the approximation $\hat{A}$ must satisfy $\|A - \hat{A}\| \leq \varepsilon$ (the fixed-precision problem). Adaptive algorithms for the fixed precision problem exist (and are discussed in Section 3.6.4), but usually either require much more passes over the data increasing communication cost (e.g. `randQB_EI` in [284]) or decreased accuracy (e.g. `randQB_FP` in [284]).

Similarly (and related), an important problem in regularized least squares problems is the selection of an appropriate regularization parameter λ . This parameter determines the balancing between fitting the data (small residual) and regularity in the solution (small solution norm). Various existing methods with rigorous justification require large computational cost [122, 130, 132], such as

solving (1.5) for many different values of λ or computing the SVD of A . This is currently inaccessible for large-scale problems with classical approaches.

1.4.1.2 Numerical stability

The numerical stability of randomized algorithms has not been studied in much detail. Numerical stability refers to the performance of algorithms when performed in finite precision, as opposed to exact arithmetic. Small rounding errors are made whenever algorithms are performed on a computer, and when they accumulate it can lead to large errors in a computed solution. As many randomized algorithms exist to compute approximations with a few digits of accuracy, rounding errors have not been much of a concern for the community. However, optimising computational routines to avoid the accumulation of rounding errors is important to attain accurate results.

One of the first works explicitly considering the numerical stability of a random embedding-based algorithm through stability analysis was Nakatsukasa (2020) [196]. Nakatsukasa analysed a generalized Nyström algorithm that computes a low-rank approximation of the form

$$AS_1^T (S_2 AS_1^T)_\epsilon^\dagger S_2 A,$$

where $(S_2 AS_1^T)_\epsilon^\dagger$ denotes the pseudo-inverse of a truncated SVD of $S_2 AS_1^T$. This is a stabilized version of (1.12) that avoids the orthogonalization cost associated to Tropp et al. (2017). Connolly, Higham, and Pranesh (2022) analyse the randomized SVD algorithm in finite precision, and also suggest a mixed precision implementation of the adaptive randomized low-rank approximation algorithm [66]. As far as the author is aware, there are no other works explicitly considering the numerical stability of randomized algorithms in a classical sense.

1.4.2 Contributions of this thesis

1.4.2.1 Numerical rank estimation

In the context of low-rank matrices, their numerical rank, and low-rank approximations, Chapter 3 discusses how sketching a dominant matrix subspace also roughly preserves singular values. As discussed in detail throughout the introduction, one can draw a small random matrix $S \in \mathbb{R}^{s \times n}$ to approximate the dominant column space of a matrix $A \in \mathbb{R}^{m \times n}$ by computing AS^T . In low-rank approximation algorithms, the sketch is then often orthogonalized (see Figure 1.3). In performing this orthogonalization, all singular values of the sketch are lost. We argue that the singular values of AS^T contain useful information about A that can aid in selecting an appropriate rank for the approximation. More generally, cheap and coarse estimates of the singular value spectrum of A can inform further algorithmic considerations in low-rank approximation. Examples include whether it

is appropriate to use a randomized SVD-type algorithm, whether the cheaper generalized Nyström algorithm would be sufficient, or whether power iterations are a useful computational step.

The rank estimation algorithm is based on two sketching procedures: AS_1^T to estimate the leading singular values of A roughly, $S_2AS_1^T$ to estimate all singular values of AS_1^T through embedding. Using theoretical and experimental evidence, we show that the numerical rank of $S_2AS_1^T$ is close to the numerical rank of A . The main theoretical contribution is on how well the first sketching procedure AS_1^T preserves singular values. We show order-of-magnitude estimates of the leading singular values of A provided that the tail of A is small compared to the singular values one tries to estimate, making the algorithm suitable for matrices with rapid singular value decay.

Many randomized algorithms exploit the fact that this kind of sketching preserves the dominant range (see left panel in Figure 1.5). The fact that the spectrum is also roughly preserved by such a sketch has often been exploited implicitly in low-rank approximation and in error estimates. For instance, for an approximation $\hat{A} = QQ^T A$ of A , the error estimate $\|(I - QQ^T)A\|_F \approx \|(I - QQ^T)AS\|_F$ can be used, where S is a small Gaussian embedding of sketching dimension, say, 10 [127, 181]. The singular values of A are also naturally used in the generalized Nyström approximation [196, 252], where the spectrum of the approximation is entirely determined by spectra of the sketches. The novelty of our algorithm is in the coarse (leading) spectrum approximation that is provided at very limited cost, and can inform further algorithmic choices for low-rank approximation. We explicitly look at singular values rather than matrix norms, and show in Chapter 3 that the order of magnitude of the singular values in the compression regime is preserved provided A is sufficiently low-rank.

Most straightforwardly, the numerical rank estimate is useful for determining a suitable rank in the fixed precision low-approximation problem. Using the knowledge on the spectrum of AS^T and error bounds available for randomized low-rank approximation algorithms, we can estimate whether the sketch size is of appropriate dimension. If we find the sketch is either too small or too large, it is trivial to reduce the size. This saves computational cost because the expensive steps in the low-rank approximation algorithms are orthogonalization and matrix-matrix multiplication. Although we still require an initial guess, combining the low-rank approximation algorithms with rank estimation scales much better with respect to this initial guess. In addition, information on the spectrum of A prior to any computationally expensive steps can inform further algorithmic choices, such as which low-rank approximation algorithm to use. This work aims to improve the cost and user-friendliness of randomized algorithms.

1.4.2.2 The stability of sketch-and-precondition for LS

The remainder of the thesis is concerned with least squares problems. The content in Chapters 5 and 6 is based upon the sketch-and-precondition framework. We investigate the numerical stability

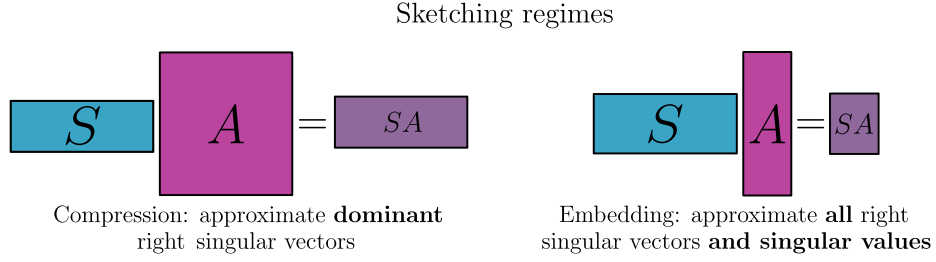


Figure 1.5: We distinguish between two sketching regimes. On the left, the sketching matrix is significantly smaller than the matrix A . As a result we preserve only an approximation to the dominant right singular vectors. We call this a compression regime. On the right, the matrix S serves as a subspace embedding for the column space of A , approximately preserving all right singular vectors and singular values. This can only happen when the smallest dimension of the embedding S is larger than the rank of A .

of these algorithms for LS problems and find, surprisingly, that they are not numerically stable in their forms presented in popular implementations (i.e. without initialization). In terms of various measures for the quality of a computed solution of a LS problem, solutions computed by sketch-and-precondition algorithms without initialization perform poorly for a range of complicated problems. These issues are caused by performing the algorithms in finite precision. In Chapter 4, we discuss two types of iterative solvers (and their preconditioned variants) in depth. Although we do not uncover the exact reason of the accumulation of rounding errors in sketch-and-precondition, this discussion serves to understand the behaviour of these algorithms better, so hopefully future work can pinpoint the stability issues.

In Chapter 5, we discuss stabilizing approaches to the sketch-and-precondition framework. Although still not always entirely numerically stable, we show empirically that using a good initial guess considerably improves the computational properties of the framework. The initialization of choice is the sketch-and-solve solution (1.8), as suggested in Rokhlin and Tygert’s paper [222] and included in general sketch-and-precondition framework [47]. The majority of the chapter is dedicated to the development and analysis of a new algorithm, which we call sketch-and-apply. Sketch-and-apply works by first explicitly computing a new operator $Y = AP$, where P is a randomized preconditioner. Then, solving $\min_z \|Yz - b\|$ using unpreconditioned LSQR on the very well-conditioned matrix Y , which we conjecture is backward stable.

As a result of the classical stability analysis of sketch-and-apply, we deduced that the approach has no stability guarantees for extremely ill-conditioned matrices, say a condition number greater than 10^{16} in double precision. To resolve this we propose ‘smoothing’ the data with a tiny amount of Gaussian noise. This not only stabilizes the method in this extremely ill-conditioned regime, but can even improve convergence behaviour of iterative solvers.

1.4.2.3 Randomized algorithms for regularized least squares

In Chapter 6 we extend the sketch-and-precondition algorithm to regularized linear LS of the form (1.5). Heuristic, as well as more formal methods, exist for selecting the appropriate regularization parameter [130]. The problem with many of these methods is that they require expensive preprocessing steps, such as computing the SVD of A , or require computing many solutions for different values of the regularization parameter.

We focus on a context where the optimal regularization parameter is unknown and develop a variety of algorithms to efficiently compute solutions corresponding to a range of admissible parameters. These algorithms are based on the sketch-and-precondition framework. We show that one can exploit the structure of (1.6) to efficiently sweep through the parameter space and update computed regularized solutions. The class of algorithms for this can handle underdetermined and overdetermined problems, can exploit low-rank structure, and is trivially parallelizable.

Theoretically, one of the main contributions of the chapter is the exploitation of rapidly decaying singular values in the data matrix for randomized preconditioners with low-rank structure. This is the first work that does so without invoking the normal equations, resulting in improved stability properties.

1.4.3 Relation to published work

Some of the work in this thesis has been published, either in a peer-reviewed journal, or as an preprint on arXiv.

- The content in Chapter 3 is based on the paper Meier, M., and Nakatsukasa, Y. (2024). *Fast randomized numerical rank estimation for numerically low-rank matrices*. Linear Algebra and its Applications, 686, 1-32.
- Some of the content in Chapter 4 and a large part of the content in Chapter 5 is based on the paper Meier, M., Nakatsukasa, Y., Townsend, A., and Webb, M. (2024). *Are sketch-and-precondition least squares solvers numerically stable?*. SIAM Journal on Matrix Analysis and Applications, 45(2), 905-929.
- Part of the content in Chapter 6 is based on the preprint Meier, M., and Nakatsukasa, Y. (2022). *Randomized algorithms for Tikhonov regularization in linear least squares*. arXiv:2203.07329.

The following work has been submitted to a journal, but its contents are not included in this thesis.

- Epperly, E. N., Meier, M., and Nakatsukasa, Y. (2024). *Fast randomized least-squares solvers can be just as accurate and stable as classical direct solvers*. arXiv:2406.03468.

1.5 Notation

In this thesis, A denotes a matrix, usually of dimension $m \times n$, over some field $\mathbb{F}$. In our case, $\mathbb{F} = \mathbb{R}$ or $\mathbb{F} = \mathbb{C}$. The i th largest singular value of A is denoted by $\sigma_i(A)$, and we use $\sigma_{\min}(A)$ and $\sigma_{\max}(A)$ for the smallest and largest singular values respectively. The condition number of A is $\kappa_2(A) = \kappa(A) = \sigma_{\max}(A)/\sigma_{\min}(A)$. The superscript $A^\dagger$ denotes the pseudoinverse of A . We use $\|\cdot\|$ or $\|\cdot\|_2$ to denote the spectral norm, and $\|\cdot\|_F$ for the Frobenius norm.

1.6 Outline

In Chapter 2, we discuss preliminaries on sketching, different types of randomized embedding matrices, low-rank approximation algorithms, and randomized algorithms for least squares problems. We give algorithms and discuss error bounds. Chapter 3 is on fast randomized rank estimation for numerically low-rank matrices, with extensions to low-rank approximation. We then switch to least squares problems. In Chapter 4, we discuss iterative solvers for least squares problems and compare their numerical stability to direct methods. We then delve into the numerical stability of the sketch-and-precondition framework and display its instabilities. The next chapter, Chapter 5, discusses stabilizing approaches and introduces a new sketch-and-apply algorithm. This algorithm is shown to be numerically stable using classical stability analysis. We consider randomized approaches to regularized least squares problems in Chapter 6. In the final chapter we conclude the thesis.

Chapter 2

Preliminaries

We discuss some of the basic preliminaries from randomized numerical linear algebra literature. Firstly, we discuss sketching, provide intuition and theory as to what it achieves, and present different types of random embedding matrices. Next, we formalize some of the randomized low-rank approximation algorithms introduced in Chapter 1 and present corresponding error bounds in Section 2.2. We finally introduce randomized algorithms for least squares (LS) problems in Section 2.3.

2.1 Sketching

In Chapter 1 we introduced two different applications of random embeddings; using a sketch smaller than the original matrix one can perform low-rank decompositions, and using a sketch slightly larger than the original matrix one can solve least squares problems. These different regimes are shown graphically in Figure 1.5 and we call them compression and embedding regimes, respectively. We discuss the intuition and theory behind these two regimes, and finally discuss different types of sketching matrices and their performance.

2.1.1 Compression: sketching dominant spaces

Consider first a matrix $A \in \mathbb{R}^{m \times n}$, where both m and n are large. What happens when this matrix is multiplied with a tall and skinny Gaussian matrix $G \in \mathbb{R}^{n \times r}$, $m, n \gg r$, from the right?¹

Following [173], suppose that A is of exact rank $r < \min(m, n)$, so it has singular values $\sigma_1 \geq \dots \geq \sigma_r > \sigma_{r+1} = \dots = \sigma_{\min(m, n)} = 0$ and SVD $A = U_1 \Sigma_1 V_1^T$ where $U_1 \in \mathbb{R}^{m \times r}$. Let $S \in \mathbb{R}^{n \times r}$ be a random matrix with independent and identically distributed (i.i.d.) $\mathcal{N}(0, 1)$ entries. Since the columns of S are linearly independent with probability 1, the columns of AS span the column space

¹Note that this is the transpose operation of compression as displayed in Figure 1.5. Most low-rank approximation algorithms sketch the column space of a matrix instead of the row space, which is why we focus our discussion there.

of A with probability 1. That is, $\text{Range}(AS) = \text{Range}(U_1)$. Just by multiplying with a smaller random matrix, we were able to reduce the dimension of the matrix at hand while recovering its entire range. In fact, this can be seen as both compression and embedding (see Figure 1.5).

If the trailing singular values are not exactly zero, but still small ($\sigma_1 \geq \dots \sigma_r \gg \sigma_{r+1} \geq \dots \geq \sigma_{\min(m,n)} \geq 0$), we have a similar effect. The rank- r truncated SVD is now an approximation to $A \approx U_1 \Sigma_1 V_1^T$, and the range of A is dominated by the dominant left singular vectors corresponding to the r largest singular values. If we now apply the same Gaussian matrix, we will still retain most of $\text{Range}(U_1)$ with $\text{Range}(AS)$. However, the space will be slightly ‘polluted’ by the trailing singular values and corresponding singular vectors.

This procedure is commonly referred to as *randomized rangefinder* [127]: by using a random matrix we can construct a small matrix whose columns roughly span the dominant range of A . As described in the introduction, we prefer random matrices that can be applied to a matrix quicker than the dense Gaussian embedding matrix. The intuition carries over to different random matrices. We discuss some variants in Section 2.1.3.

2.1.2 Embedding: sketching full spaces and the spectrum

In the embedding context, where the random matrix is larger than the matrix A , we embed the entire fundamental space of a matrix. We formally define the general subspace embedding.

Definition 2.1 (Subspace embedding [181]). *A matrix $S \in \mathbb{F}^{s \times m}$ is said to be a subspace embedding with distortion $0 < \varepsilon < 1$ for an n -dimensional subspace $\mathcal{V} \subset \mathbb{F}^m$ if*

$$(1 - \varepsilon)\|v\|_2 \leq \|Sv\|_2 \leq (1 + \varepsilon)\|v\|_2 \quad \text{for all } v \in \mathcal{V}. \quad (2.1)$$

A subspace embedding is said to be oblivious if it is constructed without prior knowledge of $\mathcal{V}$ except its dimension. If the columns of $U \in \mathbb{F}^{m \times n}$ form an orthonormal basis for $\mathcal{V}$, then (2.1) is equivalent to the condition

$$1 - \varepsilon \leq \sigma_{\min}(SU) \leq \sigma_{\max}(SU) \leq 1 + \varepsilon. \quad (2.2)$$

The matrix S is an embedding for the column space of $A \in \mathbb{R}^{m \times n}$ if

$$(1 - \varepsilon)\|Ax\|_2 \leq \|SAx\|_2 \leq (1 + \varepsilon)\|Ax\|_2 \quad \text{for all } x \in \mathbb{R}^n. \quad (2.3)$$

Subspace embedding matrices are useful when the dimension n of the subspace is considerably smaller than the dimension m of the space it lives in. In this case, we are able to map the entire subspace to a smaller dimension while preserving its geometry. We have seen how this is especially

useful for highly-overdetermined least squares problems, where our data is $m \times n$ with $m \gg n$. In this context, the subspace $\mathcal{V}$ is the column space of A or of the augmented matrix $[A, b]$.

Next we discuss different embedding matrices that achieve the subspace embedding property.

2.1.3 Sketching matrices

There are various ways to construct sketching matrices; almost all are randomly generated. We consider a few important types here: (1) Gaussian matrices [187, 259, 275], (2) subsampling matrices [146], (3) sparse random embeddings [36, 47, 57, 61, 63], (4) subsampled randomized trigonometric transforms (SRTTs) [14, 181, 222, 251], and (5) hashed randomized trigonometric transforms (HRTTs) [47].

We discuss what dimension these sketching matrices must have to ensure they satisfy the subspace embedding property in (2.3) with high probability. On the one hand, the embedding dimension s is ideally as small as possible so further computational routines with the sketch SA are cheaper. On the other hand, the cost of applying the random embedding to A should not be too large. There exists a trade-off between theoretical guarantees that allow us to use small sketches and the cost of sketching, as the following theory on sketching matrices shows. Throughout this section, we assume A is an $m \times n$ matrix of rank r .

2.1.3.1 Gaussian embedding matrices

A Gaussian embedding matrix S of size $s \times m$ is a matrix with independent and identically distributed (i.i.d.) Gaussian entries of mean 0 and variance $1/s$. If one selects $s = \mathcal{O}(r\epsilon^{-2})$, then G is a subspace embedding for $A \in \mathbb{R}^{m \times n}$ with high probability [275]. This is the optimal dependency of the embedding dimension s on the dimension r of the subspace. It has been shown that the entries of S can be any i.i.d. sub-Gaussian random variable, for instance a matrix of random signs [2]. The drawback of (sub-)Gaussian matrices is the cost of computing SA . In particular, the dense matrix-matrix multiplication costs $\mathcal{O}(mns)$ operations. This cubic cost is often exactly what we aim to avoid, making them not practical unless the embedding dimension $s = \mathcal{O}(1)$. However, Gaussian sketching matrices are often used in theoretical analysis due to the availability of excellent probability theory and concentration of measure arguments [127, 181].

2.1.3.2 Subsampling matrices

Subsampling matrices (which we denote with $\mathcal{S}$) have just one non-zero entry per row. The position of these non-zeros correspond to sampling the associated row and are chosen at random according to some distribution. One such distribution is uniform sampling, where each row has the same probability of being sampled. Uniform sampling approaches are not generally subspace embeddings for

reasonable dimensions s , because they do not take into account that some rows are more ‘important’ than others. This concept is captured by the ‘leverage scores’ of a matrix. The extent to which information is clustered in a few important rows is captured by the ‘coherence’ of A . These two quantities are defined as follows.

Definition 2.2 (Leverage scores and coherence [275]). *Let $A \in \mathbb{R}^{m \times n}$, $m > n$, and let $U \in \mathbb{R}^{m \times r}$ be a matrix with orthonormal columns spanning the column space of A . The i th leverage score l_i^2 of A is defined as*

$$l_i^2 = \|U_{(i)}\|_2^2, \quad (2.4)$$

where $U_{(i)}$ denotes the i th row of U . The coherence $\mu(A)$ of A is

$$\mu(A) = \max_{1 \leq i \leq m} l_i^2.$$

We note that there is some slight ambiguity about this definition in the literature. We follow the notation from [275] and [87] that the leverage scores are row-norms squared, whereas [47] follows [174] and defines the i th leverage score as $l_i = \|U_{(i)}\|_2$. The coherence of a matrix is conventionally defined as the maximum leverage score, and as a result suffers from the same ambiguity.

The effectiveness of uniform sampling is indicated by the coherence. The minimal coherence of a matrix of size $m \times n$ of rank r is $\mu(A) = r/m$ [47]. This condition is met when all the rows of A are equally important, which makes sampling most effective. The worst possible coherence is 1: if there is a row with $l_i^2 = 1$, then not sampling that row will make the sketch rank-deficient.

The sketching dimension s required to ensure $\mathcal{S}$ is a random embedding with high probability is $s = \mathcal{O}(\varepsilon^{-2} m \mu(A) \log(r))$ [37]. Note that $m \mu(A)$ is r at best and m at worst. Higher leverage scores, and so larger coherence, imply we need to sample more rows to obtain a subspace embedding. On top of that, there is an oversampling factor of $\log(r)$ not present for Gaussian sketches.

Uniform sampling does not meet theoretical guarantee standards as a result of the dependency on the coherence. For such guarantees, we need to sample with a distribution related to the leverage scores [64]. If rows are sampled with a probability proportional to l_i^2 , then the required sketching dimension is $s = \mathcal{O}(\varepsilon^{-2} r \log(r))$ [90]. The exact computation of leverage scores is not achievable in less than $\mathcal{O}(mn^2)$ time, but algorithms exist to estimate these quantities quicker [87]. Such leverage score estimation algorithms also result in a sketching dimension of $s = \mathcal{O}(\varepsilon^{-2} r \log(r))$, making these approaches competitive.

2.1.3.3 Sparse random embeddings

Sparse, hashed, or CountSketch random embeddings are matrices with a fixed number of non-zeros per column. These non-zeros are usually scaled random signs. We say a matrix is a ζ -hashing matrix

if it has ζ random signs per column. In their first appearance in [61], sparse random embeddings were presented as 1-hashing matrices, and the sketching dimension required was $\mathcal{O}(n^2\varepsilon^{-2})$. If one has an upper bound on the coherence, however, the dimension can be smaller. Bourgain et al. [36] show that if the coherence of A is $\mathcal{O}(\log^{-3}(r))$ then $s = \mathcal{O}(r \log^2(r)\varepsilon^{-2})$ is required. Cartis et al. show that the optimal sketching dimension $s = \mathcal{O}(r\varepsilon^{-2})$ can be achieved with a 1-hashing matrix if $\mu(A) = \mathcal{O}(r^{-2})$ [47]. Note that this can only hold when $m \gg r$.

Better embedding dimension bounds can be achieved by using ζ -hashing matrices for $\zeta > 1$. [63] showed that if $\zeta = \mathcal{O}(\log r)$ then $s = \mathcal{O}(r \log(r)\varepsilon^{-2})$ results in a subspace embedding, without any conditions on the coherence. Tropp et al. [253] suggest that $\zeta = 8$ leads to good embedding properties in practical applications, whereas Cartis et al. [47] obtain good results with $\zeta = 3$. The main advantage of sparse embedding matrices is that they can be applied to $A \in \mathbb{R}^{m \times n}$ in $\mathcal{O}(\zeta mn)$ flops.

2.1.3.4 Subsampled randomized trigonometric transforms

Considering the dependency on the coherence of a matrix, we might also consider approaches to reduce the coherence before subsampling. This is the idea behind the subsampled randomized trigonometric transform (SRTT) embedding matrix [14, 181, 222, 251]. It has the form

$$S = SFD, \tag{2.5}$$

where D is a square diagonal matrix with ± 1 entries at random, F is an orthogonal trigonometric transform (e.g., Fourier, cosine, or Hadamard), and S is an $s \times m$ scaled uniform sampling matrix with one non-zero entry per row. Application of the random signs and orthogonal transform serves to smooth out the information in the rows, thus reducing the coherence. We make this specific in Section 3.4. After this smoothing, we can perform uniform subsampling with embedding dimension $s = \mathcal{O}(r \log(r)\varepsilon^{-2})$, exactly because the coherence will be close to minimal [251].

Although the performance guarantees of these embeddings are less strong because of the $\log(r)$ factor, practically, the obtained accuracy is comparable to Gaussian random matrices. That means the $\log(r)$ factor in s can often be dropped in practice [181]. Moreover, the matrix-matrix product SA can be computed in $\mathcal{O}(mn \log s)$ operations [181], although the usual computational cost is $\mathcal{O}(mn \log m)$ associated with computing $S(F(DA))$, as fast implementations for the SRTT are not readily available.

2.1.3.5 Hashed randomized trigonometric transforms

Naturally, one might also combine sparse random embeddings with randomized trigonometric ‘pre-conditioning’. The hashed randomized trigonometric transform is equivalent to (2.5), but with the

subsampling matrix $\mathcal{S}$ replaced by a sparse random embedding. To the author's best knowledge, these embeddings were first introduced in [47]. It is shown in [47] that for sufficiently overdetermined matrices ($m > n^3$), combining a randomized trigonometric transform with a 1-hashing matrix requires an embedding dimension of just $s = \mathcal{O}(r\epsilon^{-2})$, thus removing the additional $\log(r)$ factor present for SRTTs.

2.2 Randomized low-rank approximation

We next discuss the most widely-known randomized algorithm for low-rank approximation: the randomized SVD algorithm. Discussed in detail throughout the introduction, the idea is to sketch the dominant range of A and project A upon the subspace spanned by the sketch. Let $A = U\Sigma V^*$ be an SVD of $A \in \mathbb{C}^{m \times n}$, and let $A = U_1 \Sigma_1 V_1^*$ be its rank- r truncated SVD. We form the sketch $AS \in \mathbb{C}^{m \times r}$ using a random embedding S , and then orthogonalize its columns (for instance, with a QR decomposition $AS = QR$) to obtain a matrix $Q \in \mathbb{R}^{m \times r}$ with the same range. We can then project A onto the range of Q to obtain a low-rank factorization:

$$\hat{A}_r = QQ^*A \approx U_1 U_1^* A = A_r. \quad (2.6)$$

This is the main idea in the randomized SVD algorithm (see also Figure 1.3).

By letting $B = Q^*A$, we see that this is already a low-rank approximation of the form $A \approx QB$, where $Q \in \mathbb{C}^{m \times r}$ and $B \in \mathbb{C}^{r \times n}$. We can reformulate it as an approximate truncated SVD by computing the SVD of the small matrix $B = \tilde{U}\hat{\Sigma}\hat{V}^*$ (using a classical algorithm) and setting $\hat{U} = Q\tilde{U}$. Then, $A_r \approx \hat{U}\hat{\Sigma}\hat{V}^*$, where $\hat{U} \in \mathbb{C}^{m \times r}$ and $\hat{V} \in \mathbb{C}^{n \times r}$ are matrices with orthonormal columns and $\hat{\Sigma}$ is a diagonal $r \times r$ matrix.

In practice, the quality of the space spanned by Q as an approximation for U_1 is greatly improved by setting the sketching dimension s of S slightly larger than the dimension r of U_1 . This is referred to as oversampling: $s = r + p$, where p is usually a small integer [127] or a small multiple of r [196, 208]. The algorithm is displayed in pseudocode in Algorithm 1.

Algorithm 1 Randomized SVD [127] or randomized rangefinder [181]. Given an $m \times n$ matrix A , a target rank r , and an oversampling parameter p , this scheme computes a low rank approximation $\hat{A}_{\text{RF}} = QB$ ($Q \in \mathbb{F}^{m \times (r+p)}$) or $\hat{A}_{\text{RSVD}} = \hat{U}\hat{\Sigma}\hat{V}^*$, where $\hat{\Sigma} \in \mathbb{R}^{(r+p) \times (r+p)}$ is diagonal and $\hat{U}$ and $\hat{V}$ have orthogonal columns.

- 1: Draw a $n \times (r + p)$ random matrix S .
 - 2: Form the $m \times (r + p)$ matrix $Y = AS$.
 - 3: Construct $m \times (r + p)$ factor Q of the QR decomposition of Y .
 - 4: Compute the factor $B = Q^*A$.
 - 5: Let $\hat{A}_{\text{RF}} = QB$, or compute the SVD of $B = \tilde{U}\hat{\Sigma}\hat{V}^*$, set $\hat{U} = Q\tilde{U}$, and let $\hat{A}_{\text{RSVD}} = \hat{U}\hat{\Sigma}\hat{V}^*$. These factors can be truncated to obtain a rank- r approximation.
-

Error bounds have been extensively studied and both typical error bounds and tail bounds are available. We present the expected error in the Frobenius norm here. These bounds are later used in Section 3.5.2.

Lemma 2.3 (Halko, Martinsson, and Tropp (2011)). *Let $A \in \mathbb{R}^{m \times n}$ have an SVD $A = U_1 \Sigma_1 V_1^T + U_2 \Sigma_2 V_2^T$ as in Figure 1.1, where U_1 ($m \times r$) and V_1 ($n \times r$) have orthonormal columns and Σ_1 is a diagonal matrix of size $r \times r$ (the other factors are of conforming size). Let $\hat{A} = \hat{A}_{RF} = \hat{A}_{RSVD}$ denote the low-rank approximation of A computed using Algorithm 1, then*

$$\|A - \hat{A}\|^2 \leq \|\Sigma_2\|^2 + \|\Sigma_2 V_2^T S(V_1^T S)^\dagger\|^2, \quad (2.7)$$

in both the spectral and Frobenius norm. Furthermore, assuming $S \in \mathbb{R}^{n \times (r+p)}$ is a Gaussian random matrix and $r, p \geq 2$, then

$$\mathbb{E}\|A - \hat{A}\|_F \leq \left(1 + \frac{r}{p-1}\right)^{1/2} \left(\sum_{j>r} \sigma_j^2\right)^{1/2} = \left(1 + \frac{r}{p-1}\right)^{1/2} \|\Sigma_2\|_F. \quad (2.8)$$

Recall that the optimal error of a rank- r approximation is $\|\Sigma_2\|_F$. This typical error bound, as well as numerical experiments, confirm that the approximation is quite close to optimal [127]. The particular error bound (2.8) is specific to Gaussian embedding matrices, although in practice observed errors are similar for other kind of embedding matrices [126]. Invoking these error bounds a priori to ensure a certain accuracy on the approximation requires knowledge about the singular value spectrum of A . This is one application of the numerical rank estimation scheme discussed in Chapter 3. Ensuring a specific accuracy on the approximation can also be done adaptively, which we discuss in Section 3.6.4

The quality of the sketch AS is not trivially deduced from the embedding properties of the sketching matrices used. The embedding property (2.1) is not immediately related to the errors terms in low-rank approximations, e.g. $\|\Sigma_2 V_2^T S(V_1^T S)^\dagger\|$ in (2.7). Analysis of the such terms in deterministic error bounds of randomized low-rank approximation algorithms is key, but requires separate analysis for different types of sketching matrices (see [37, 227]).

2.3 Randomized algorithms for LS problems

There is a wide range of randomized algorithms available to solve linear least squares problems where the task is to compute

$$x^* = \operatorname{argmin}_{x \in \mathbb{R}^n} \|Ax - b\|, \quad A \in \mathbb{R}^{m \times n}, m \gg n, b \in \mathbb{R}^m. \quad (2.9)$$

We discuss the two main randomized paradigms. The first, sketch-and-solve, sacrifices some accuracy to substantially speed up the computation of an approximate solution. The second, sketch-and-precondition, is a framework to design preconditioners to be used in deterministic iterative solvers. Other methods that we do not discuss include iterative sketching [100, 159, 200, 216] and randomized Kaczmarz [240].

2.3.1 Sketch-and-solve

In the sketch-and-solve paradigm, we map the entire least squares problem (1.2) to a lower dimension. The idea was first introduced and analysed in [229]. The (much smaller) sketched problem is then solved using a classical method to find an approximate solution x_{SAS} to $\min \|Ax - b\|$ by computing

$$x_{\text{SAS}} = \operatorname{argmin}_{x \in \mathbb{R}^n} \|SAx - Sb\|, \quad (2.10)$$

where $S \in \mathbb{R}^{s \times m}$, with $n \leq s < m$, is an embedding matrix. Classical algorithms such as the QR decomposition solve this problem in $\mathcal{O}(sn^2)$ operations. One would compute $SA = QR$ and then $x_{\text{SAS}} = R^{-1}Q^T Sb$.

The accuracy of sketch-and-solve solutions are well-understood [90, 229, 275]. For appropriately sized embedding matrices, we are able to attain a residual within a small constant of the optimal residual. This is made specific in the following lemma.

Lemma 2.4 (Woodruff (2014)). *Let $A \in \mathbb{R}^{m \times n}$ and $S \in \mathbb{R}^{s \times m}$, where $n \leq s < m$, and assume A and SA are of full column rank. Let $Q_{[A,b]}$ denote a matrix whose columns form an orthonormal basis for the column space of $[A, b]$. Let x_{SAS} denote the sketch-and-solve solution as defined in (2.10), and let x^* be the solution to the original LS problem (1.2). Then*

$$\|Ax_{\text{SAS}} - b\| \leq \kappa_2(SQ_{[A,b]}) \|Ax^* - b\|. \quad (2.11)$$

Note the relation between $\kappa_2(SQ_{[A,b]})$, the singular values of $SQ_{[A,b]}$, and Definition 2.1. If S is a subspace embedding for an $(n+1)$ -dimensional space, then the condition number above will be bounded by $(1+\varepsilon)/(1-\varepsilon)$. The embedding distortion $\varepsilon \in (0, 1)$ is usually $o(1)$. It follows that for a sufficiently large sketch dimension, the accuracy (in terms of residual) of the sketch-and-solve solution is of the same order of magnitude as that of the optimal solution. This brings the total cost of the method to $\mathcal{O}(mn \log(n) + n^3 \log(n))$, where the first term is the cost of sketching and the second of computing the solution to the sketched problem.

The smaller sketched system may also be solved iteratively, for instance with the normal equations or in the dual space for underdetermined problems [170]. Chen et al. [54] propose an algorithm to sketch-and-solve the normal equations efficiently.

2.3.2 Sketch-and-precondition

An important deterministic class of algorithms to solve LS problems are iterative Krylov subspace method [224, 250]. These methods minimize a specific quantity in a growing solution space. For LS problems, the solution space at the k th iteration is the k -dimension Krylov subspace associated to the normal equations [187, 205]

$$\mathcal{K}_k(A^T A, A^T b) = \text{span} \{A^T b, (A^T A)A^T b, \dots, (A^T A)^{k-1} A^T b\}.$$

LSQR [205], for instance, computes iterates that minimize the residual $x_k = \operatorname{argmin}_{x \in \mathcal{K}_k} \|Ax - b\|$, whereas in LSMR [187] we look to minimize the normal residual $\|A^T(Ax - b)\|$ in each iteration. The cost per iteration is $\mathcal{O}(mn)$ for dense matrices and $\mathcal{O}(\text{nnz}(A))$ for sparse matrices, where $\text{nnz}(A)$ is the number of non-zero entries in A . The number of iterations required to reach a desired accuracy typically depends on the condition number of A [34, Ch. 7.4]. The condition number $\kappa_2(A)$ is defined as the ratio of the largest and smallest singular values of A :

$$\kappa_2(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}. \quad (2.12)$$

If A has a wide spectrum (large $\kappa_2(A)$), then any Krylov subspace method will need many iterations to converge to a reasonable solution, making them computationally infeasible.

A solution to this is to use a preconditioner. A preconditioner is a full-rank matrix $P \in \mathbb{R}^{n \times n}$ that applies a change of variables to turn the LS problem into

$$y^* = \operatorname{argmin}_{y \in \mathbb{R}^n} \|APy - b\|, \quad x^* = Py^*, \quad (2.13)$$

where AP should be better conditioned than A . The rate of convergence of this equivalent problem is then dominated by $\kappa_2(AP)$. With a good preconditioner, we can ensure robustness and a reasonable speed of convergence for the iterative solver. The difficulty is, of course, how to find such a P at low computational cost. Various methods exist for sparse LS problems [115]. For dense problems, however, good algebraic preconditioners were not available before the sketch-and-precondition framework was introduced.

The idea behind the sketch-and-precondition technique, first proposed in [222], is that an excellent preconditioner can be constructed from a sketch of A . There are many details of sketch-and-precondition which vary in approaches to sketching, constructing P , and which iterative LS solver is employed [14, 47, 187, 222].

One of the most prominent sketch-and-precondition algorithms is known as Blendenpik [14] (see Algorithm 2). In Blendenpik (see Figure 1.4), the matrix A is sketched to a small tall-skinny

matrix SA and the preconditioner is taken to be the inverse of the upper triangular factor from a QR factorization of $SA = QR$, i.e., $P = R^{-1}$. The problem is then solved using the iterative solver LSQR [205]. Another popular sketch-and-precondition technique is LSRN [187], which specifically uses a Gaussian sketch and the SVD of $SA = U\Sigma V^T$ instead of the QR decomposition. The preconditioner is then $P = V\Sigma^{-1}$. A general presentation of the sketch-and-precondition framework is provided in [47], accompanied by a C++ implementation named Ski-LLS. This framework most notably can account for rank-deficiency by allowing the use of rank-revealing and truncated decompositions of the sketch in the computation of a preconditioner.

We take the point-of-view that the most standard sketch-and-precondition algorithm uses QR to compute a preconditioner and LSQR as the iterative solver, but that other choices are part of the sketch-and-precondition framework.

Algorithm 2 A sketch-and-precondition LS solver for (5.1). Here, HHQR refers to the Householder QR algorithm.

- 1: Draw a random sketching matrix $S \in \mathbb{R}^{s \times m}$, where $m \gg s > n$
 - 2: Compute $B = SA$
 - 3: Compute the triangular factor R of a QR factorization of B using HHQR
 - 4: Solve $\min \|Ax - b\|$ with LSQR and right-preconditioner $P = R^{-1}$
-

With careful sketching, the condition number of AP with $P = R^{-1}$ from Algorithm 2 ($SA = QR$) is small with high probability (see [222, Lem. 1] and Lemma 2.5). In particular, the condition number of AP is equal to the condition number of SQ_A , where Q_A is an orthonormal basis for the column space of A .

Lemma 2.5. *Let $A \in \mathbb{R}^{m \times n}$ have linearly independent columns, $S \in \mathbb{R}^{s \times m}$ with $s \geq n$ have linearly independent rows, and $B = SA$. If $A = Q_A R_A$ and $B = QR$ are economized QR factorizations of A and B , respectively, then*

$$\kappa_2(AP) = \kappa_2(SQ_A).$$

where $P = R^{-1}$.

Proof. The proof follows the same argument by Meng, Saunders, and Mahoney in [187, Lemma 4.2] and Rokhlin and Tygert in [222, Thm. 1]. Note $AP = Q_A R_A R^{-1}$, so that $\kappa_2(AP) = \kappa_2(R_A R^{-1})$. Now, since $B = SA = QR = S(Q_A R_A)$, we have that $(SQ_A)R_A R^{-1} = Q$ is orthonormal. Hence, $\kappa_2(R_A R^{-1}) = \kappa_2(SQ_A)$. $\square$

Lemma 2.5 shows that the main idea behind sketch-and-precondition techniques is excellent: the value of $\kappa_2(AP)$ is independent of $\kappa_2(A)$, and can be made $\mathcal{O}(1)$ using the sketching matrices previously introduced. As a result, one expects rapid convergence of iterative LS solvers using the preconditioner P . The only task at hand is choosing a sketching matrix S so that $\kappa_2(SQ_A)$ is close to 1 with high probability. Again, note that this is equivalent to finding a subspace embedding for

Q_A , due to (2.2). If S is a subspace embedding for Q_A with distortion $\varepsilon \in (0, 1)$, then $\kappa_2(SQ_A) \leq (1 + \varepsilon)/(1 - \varepsilon)$. The theory in Section 2.1.3 gives us appropriate embedding dimensions to achieve a well-conditioned matrix.

Sketch-and-precondition algorithms require $\mathcal{O}((\log(n) + \log(1/\varepsilon))mn + n^3 \log(n))$ operations to obtain a solution to relative precision ε . The cost of applying a subspace embedding is upper bounded by $\mathcal{O}(mn \log(n))$. The embedding dimension is then usually $\mathcal{O}(n \log(n))$ for a condition number $\kappa_2(SQ_A) = \mathcal{O}(1)$. The associated computational cost for computing the preconditioner is $\mathcal{O}(n^3 \log(n))$. Because of the well-conditioning of AP , a Krylov subspace solver converges in $\mathcal{O}(\log(1/\varepsilon))$ iterations for

$$\|Ax_k - b\| \leq (1 + \varepsilon)\|Ax^* - b\|$$

relative accuracy at $4mn$ cost per iteration [222]. Recent high-performance implementations can be found in [47, 148, 149].

Rokhlin and Tygert [222] included a sketch-and-solve initial guess in the description of their original sketch-and-precondition algorithm. This initial guess is the solution to (2.10), and can be computed with the QR decomposition of SA (with almost no additional cost). Notable later works, such as Blendenpik [14] and LSRN [187], have not (clearly) included this initialization, so we do not include it in the standard framework at this point. A sketch-and-solve initialisation is included in the standard framework presented by Cartis et al. (2021) and in the Ski-LLS implementation [47]. Indeed, throughout this thesis, we argue that this inclusion is crucial for good performance of the sketch-and-precondition algorithms.

2.4 Looking ahead

The randomized SVD and sketch-and-precondition frameworks are two very successful randomized algorithms. Their practical significance has been observed countlessly and the algorithms have been adopted by the wider computational science community. They also serve as the main inspiration for this thesis.

As noted before, a significant drawback of the randomized SVD algorithm in Algorithm 1 is that the target rank must be user-specified. Although there might be some upper bound on the rank available through, e.g., memory limitations, it is often more natural to look for a low-rank approximation $\hat{A}$ such that $\|A - \hat{A}\| \leq \varepsilon$ for some tolerance ε . Even this tolerance might not be obvious, but rather determined by some gap in the singular value spectrum of A . In the next chapter, we provide an algorithm aimed at answering these questions.

Chapter 3

Fast randomized rank estimation for numerically low-rank matrices

3.1 Introduction

Low-rank matrices are ubiquitous in scientific computing and data science. Sometimes a matrix of interest can be shown to be of numerically low rank [29,257], for example by showing that the singular values decay rapidly. More often, matrices that arise in applications may have a hidden low-rank structure, such as low-rank off-diagonal blocks [176,181]. As is well known, low-rank approximation is also the basis for principal component analysis.

Algorithms and methods that are able to exploit low-rank structure usually require a low-rank approximation. As discussed in Chapter 1 for large-scale matrices, traditional deterministic algorithms for low-rank approximation based on computing a truncated singular value decomposition (SVD) might be infeasible. This can be caused by the sheer scale of the matrix and the complexity of classical algorithms, as well as the fact that access to the matrix may be restricted because it is stored in RAM or cannot be stored at all (streaming model) [181]. Randomized algorithms have become a powerful and reliable tool for efficiently computing a near-best low-rank approximation for such matrices. Landmark references on randomized low-rank factorizations are [275] and [127], which extensively analyze the randomized SVD (also introduced in Section 2.2).

A key step in low-rank algorithms, including randomized SVD, is the determination of the numerical rank (based on the spectral norm). For instance, a variety of low-rank factorization techniques require the target rank of the factorization as input information. Once a low-rank approximation $\hat{A}_r$ of the specified target rank has been computed, a posteriori estimation of the error $\|A - \hat{A}_r\|_2$ via a small number of matrix-vector multiplications [181] is a reliable means of checking if the input rank

r was sufficient. If r was too low, one would need to sample the matrix with more vectors. This clearly requires more computational work, and can be a major difficulty in the streaming model, wherein revisiting the matrix is impossible [253]. Conversely, if the input rank r was too high, the computational cost of computing $\hat{A}_r$ would be higher than it could be. Having a fast and reliable rank estimator is therefore highly desirable.

In this work we propose and analyze a fast randomized algorithm for estimating the numerical rank of an approximately low-rank matrix $A \in \mathbb{R}^{m \times n}$ or $A \in \mathbb{C}^{m \times n}$. The algorithm is based on sketching both the column as well as the row space of A , forming S_2AS_1 , where S_1 and S_2 are random (subspace embedding) matrices. The key idea is that with high probability, the singular values of AS_1 are good estimates of the (leading) singular values $\sigma_i(A)$ of A . Therefore the decay of $\sigma_i(A)$ can be reliably estimated by the decay of $\sigma_i(AS_1)$. To estimate $\sigma_i(AS_1)$ we once again sketch AS_1 to obtain the much smaller matrix S_2AS_1 ; it is only $\sigma_i(S_2AS_1)$ (or their estimates) that we actually need to compute. It is noteworthy that in the algorithm it is only necessary to view the matrix A once. Furthermore, since we obtain estimates $\sigma_i(S_2AS_1)$ for the leading singular values $\sigma_i(A)$, we will be able to detect a gap in the spectrum of A and hence have the possibility of setting a tolerance for the numerical rank in a data-driven manner. We must note that for $\sigma_i(S_2AS_1)$ to be order-of-magnitude estimates of $\sigma_i(A)$, the tail singular values of A must be light as compared to the singular values we aim to estimate. As a result, the algorithm is suitable for matrices with rapidly decaying singular value spectrum.

The singular values of the sketch AS have been used previously in randomized numerical linear algebra. In particular, it is a well-established technique to employ a small Gaussian sketch for a posteriori error estimation for low-rank approximation algorithms. Halko, Martinsson, and Tropp propose an error estimation algorithm for the approximation $A \approx QQ^*A$ “(almost) for free” [127, p. 242]. Fundamentally, for a Gaussian vector g (or a small number of Gaussian vectors [181]), $\|(I - QQ^*)Ag\|_2$ is an excellent estimate for the error $\|(I - QQ^*)A\|$ and can be computed cheaply. Another important application of the spectrum of AS is in the generalized Nyström approximation. This low-rank approximation has the form $A \approx AS_1(S_2AS_1)^\dagger S_2A$. It is immediate that the entire spectrum of the approximation is determined by the spectra of the sketches AS_1 , S_2A , and S_2AS_1 . Our contribution is to use the spectrum of S_2AS_1 a priori, as a way to estimate the spectrum of A and inform further algorithmic choices.

To our knowledge, the fact that a sketch of the form S_2AS_1 preserves ‘some coarse spectral structure’ was first noted in [9]. We use techniques similar to their proofs for our theoretical results. Related results for the bulk of the eigenvalues of $(AS_1)^*(AS_1)$ were first obtained in [111]. The main theoretical contributions of this work, deterministic and probabilistic bounds on the accuracy of $\sigma_i(AS_1)$ as estimates for $\sigma_i(A)$, build on these results. In particular, we provide a tighter lower bound on $\sigma_i(AS_1)/\sigma_i(A)$ relevant for small singular values of AS_1 .

The complexity of our algorithm is $O(mn \log n + r^3)$ for dense $m \times n$ matrices, and can be lower if A has structure that can be exploited for computing the sketches AS_1 and S_2AS_1 . This is clearly a subcubic complexity (assuming $r \ll m, n$), and it runs significantly faster than computing the full SVD. Moreover, in many cases, computing AS_1 (which is usually the dominant part of our rank estimation algorithm) is a required part of the main algorithm (e.g. randomized SVD); and in some algorithms [61, 196] this is true even of S_2AS_1 . In such cases, the additional work needed for estimating the rank is therefore significantly smaller, such as $O(r^3)$ or sometimes even $O(r)$.¹ Our algorithm is competitive when the target rank r is much smaller than the dimensions of the matrix.

In particular, for the widely used randomized SVD we argue that it is computationally efficient to compute S_2AS_1 and its singular values, after having computed a sketch AS_1 . The reason is that $\sigma_i(S_2AS_1)$ will provide information on how appropriate the size of the sketch AS_1 is. Subsequently, more columns can be added or columns can be removed before computing a QR decomposition $QR = AS_1$ and computing the product $B = Q^*A$. This will save computational cost in either case: an unnecessary large sketch, or a sketch of insufficiently large dimension. We also introduce an algorithm that combines the rank estimation algorithm with randomized SVD, using the information provided by $\sigma_i(S_2AS_1)$. This is a specific example of how the rank estimation algorithm may be useful in combination with a low-rank approximation scheme. However, we argue that its usefulness is broader than that. With a cheap estimate of the leading part of the spectrum of A , one can make an informed decision on different types of low-rank approximation algorithms to use (e.g. randomized SVD or generalized Nyström), how much oversampling is necessary, and whether power iterations are necessary for the desired accuracy.

The analysis in this chapter will be specific to Gaussian or subsampled randomized trigonometric transform (SRTT) [181] embedding matrices at times. We will use $G \in \mathbb{R}^{n \times r}$, where $n > r$, to denote a matrix where each entry is iid $\mathcal{N}(0, 1)$ distributed and call this matrix a real (standard) Gaussian matrix. In case of $\mathbb{F} = \mathbb{C}$, $G \in \mathbb{C}^{n \times r}$ denotes a complex Gaussian matrix with iid $\mathcal{CN}(0, 1)$ entries. That is, $\text{Re}(G_{ij}), \text{Im}(G_{ij}) \sim \mathcal{N}(0, 1/2)$ and independent. A Gaussian matrix can be scaled to an embedding matrix by defining $S = G/\sqrt{r}$ (in both the real and complex case); we call the scaled Gaussian matrices *Gaussian embedding matrices*. SRTT matrices are defined in Section 2.1.3.4 and as the scaling is included in the definition, they are naturally embedding matrices. We will use Θ to denote an SRTT matrix.

3.1.1 Numerical rank and goal of a rank estimator

So far we have been using the term “the numerical rank” informally. Let us now define the notion. This is a standard definition, see for example [29].

¹A computational cost of $\mathcal{O}(r)$ is possible if one uses the diagonal elements of the R factor of a QR decomposition of S_2AS_1 as estimates for the singular values.

Definition 3.1. Let $A \in \mathbb{F}^{m \times n}$. The ϵ -rank of A , denoted by $\text{rank}_\epsilon(A)$, is the integer² i such that $\sigma_i(A) > \epsilon \|A\|_2 \geq \sigma_{i+1}(A)$.

We adopt the *relative* ϵ -rank definition as it is a natural choice in the context of low-rank matrix approximation. One could easily adapt the algorithm presented to estimate the *absolute* ϵ -rank. Let us discuss what the goal of a rank estimator should be. One natural answer of course is that the estimator should output $\text{rank}_\epsilon(A)$ given A and the (user-defined) relative threshold ϵ . However, we argue that the situation is more benign: consider, for example, a matrix with $\|A\| = 1$ and k singular values $> 10\epsilon$, five singular values in $(\epsilon, 1.1\epsilon)$, five more in $(0.9\epsilon, \epsilon)$, and the remaining $n - k - 10$ are $< 0.01\epsilon$. What should the estimator output? Is it crucial that the “correct” value $\text{rank}_\epsilon(A) = k + 5$ is identified?

In this thesis we take the view that the goal of the rank estimator is to find a good ϵ -rank, not necessarily *the* ϵ -rank. In the example above, any number between k and $k + 10$ would be an acceptable output. In most applications that we are aware of (including low-rank approximation, regularized linear systems, etc), there is little to no harm in choosing a rank $\hat{r} \neq \text{rank}_\epsilon$: a slight overestimate $\hat{r} > \text{rank}_\epsilon$ usually results in slightly more computational work, whereas a slight underestimate $\hat{r} < \text{rank}_\epsilon$ is fine if $\sigma_{\hat{r}+1}(A) = O(\epsilon \|A\|)$, that is, a rank- $\hat{r}$ matrix can approximate A to relative $O(\epsilon)$ -accuracy.

On the other hand, it is clearly *not* acceptable if the rank is unduly underestimated in that $\sigma_{\hat{r}+1}(A) \gg \epsilon \|A\|$. It is also unacceptable if $\sigma_{\hat{r}}(A) \ll \epsilon \|A\|$. The goal in this chapter is to devise an algorithm that efficiently and reliably finds an $\hat{r}$ such that

- $\sigma_{\hat{r}+1}(A) = O(\epsilon \|A\|)$ (say, $\sigma_{\hat{r}+1}(A) < 10\epsilon \|A\|$): $\hat{r}$ is not a severe underestimate, and
- $\sigma_{\hat{r}}(A) = \Omega(\epsilon \|A\|)$ (say, $\sigma_{\hat{r}}(A) > 0.1\epsilon \|A\|$): $\hat{r}$ is not a severe overestimate.

Any such $\hat{r}$ is sensible in that there exists a rank- $\hat{r}$ approximation of A with $O(\epsilon)$ relative accuracy, and $\hat{r}$ is not much larger than the best possible for the accuracy required. In addition, the nature of randomized algorithms means that the user must be willing to accept results that do not hold precisely and that small inaccuracies in the results are acceptable. As a result, we assume one is not looking for the exact numerical rank, but an order-of-magnitude estimate. Our rank estimate will satisfy these conditions with high probability. In particular, in situations where the numerical rank is clearly defined, i.e., if a clear gap is present in the spectrum $\sigma_r(A) \gg \epsilon \|A\| \gg \sigma_{r+1}(A)$, the algorithm will reliably find the exact rank $\hat{r} = \text{rank}_\epsilon(A)$.

Additionally, in situations where it is unknown whether the matrix A is low-rank approximable, our rank estimator can tell us (roughly) how well A can be approximated with a low-rank matrix.

²If no such i exists, we take $\text{rank}_\epsilon(A) = \min(m, n)$; however, in this chapter we are never interested in this full-rank case. We also note that MATLAB’s `rank` function takes the *absolute* ϵ -rank, i.e., $\sigma_i(A) > \epsilon \geq \sigma_{i+1}(A)$ for a user-specified tolerance ϵ

Similarly, if it is unknown what a suitable tolerance would be and/or the user would like to detect a gap in the spectrum, the algorithm can be used to plot an estimated spectrum and detect gaps. Furthermore, the general shape and decay of the spectrum can inform further algorithmic choices for low-rank approximation algorithms.

This chapter is organized as follows. Section 3.2 discusses related studies in the literature. In Section 3.3 we show that $\sigma_i(AS_1)$ gives useful information about $\sigma_i(A)$ for leading values of i . Then in Section 3.4 we show that $\sigma_i(AS_1)$ can be estimated via $\sigma_i(S_2AS_1)$. Section 3.5 summarizes the overall rank estimation algorithm. Numerical experiments are presented in Section 3.6.

3.2 Related work

3.2.1 Existing methods for numerical rank estimation

At the core of numerical rank estimation lies singular value estimation; specifically, the rank can be found by counting the number of singular values greater than the tolerance ϵ . The most direct way to do this is to explicitly compute the singular values of a matrix. However, as is well known, this requires $O(mn^2)$ operations for an $m \times n$ matrix [114], and for large matrices this is computationally inadmissible. Another point of view, specifically for Hermitian matrices, is that a numerical rank estimate can follow from estimating the density of states (DOS). The DOS can be interpreted as a probability density function describing the position of the eigenvalues, and can be estimated with algorithms analyzed in Lin et al. [169].

Regarding fast algorithms that can run with subcubic complexity, the literature on rank estimation appears rather scant. Exceptions include the work of Ubaru and Saad [254], Ubaru, Saad and Seghouane [255], Zhang, Wainwright and Jordan [285] and Di Napoli, Polizzi and Saad [82]. These works are all concerned with counting the number of eigenvalues in a certain interval for a Hermitian matrix. Alternatively, [58] discusses an algorithm for computing the exact rank of a matrix.

The first two references [254, 255] employ the idea of density of states (DOS). In [254], the authors use the DOS to locate a gap in the spectrum, derive an appropriate tolerance ϵ and subsequently use polynomial approximation and stochastic trace estimation to count the number of eigenvalues greater than the tolerance. Computationally, the algorithm only requires matrix-vector products with A but consequently also requires many views of A . In the second paper [255], the authors also first estimate the DOS to locate a gap and then estimate the integral of the DOS. We compare against these algorithms in the numerical experiments, although they need to be adjusted slightly to account for matrices that are not symmetric positive semi-definite. An advantage of both methods is that the entire spectrum is estimated, however they require many more views of the matrix than most randomized methods. The results are discussed in Section 3.6.3. The works [285] and [82] are similar in spirit. The first focuses on minimizing communication cost involved with rank estimation

in a distributed setting, the second approximates an eigenvalue count using polynomial and rational approximation filtering.

As for more theoretical contributions, much of this work builds upon Andoni and Nguyen [9], which is to the best of the author’s knowledge the first work that shows rank can be estimated via sketches. The work is focused on estimating eigenvalues of Hermitian matrices via the sketch $S_1^T A S_1$, but also considers the singular values of non-symmetric matrices using $S_2 A S_1$. While Andoni and Nguyen work with the Gram matrix of the sketch, obtaining results in terms of the difference of the squared singular values (or via the Jordan-Wielandt matrix, which is larger), we work with $S_2 A S_1$ directly. Moreover, their work is of a theoretical nature, and do not present a concrete algorithm for estimating the rank. They also state results that only come with relatively low probability guarantees, although it is possible to take a larger sample to improve the probability.

Here we introduce precise algorithms (also for low-rank approximation), show numerical experiments, and phrase our results in terms of popular embedding matrices allowing for results that hold with exponentially small failure probability.

3.2.2 Related work in statistics

Rank estimation, or singular value estimation, is a well-known problem in the context of covariance matrix estimation or PCA in statistics, or the detection of signals in signal processing. Most rank estimation algorithms in this context require computing an SVD or eigendecomposition; exceptions include [256]. Since we consider a context where computing these decompositions exactly is not computationally feasible, we do not take this into account.

The statistics literature is connected to this work in yet another way: relating the singular values of the sketch AG to the singular values of A is equivalent to the problem of covariance matrix estimation. This can be seen as follows: let $A = U\Sigma V^*$ be the SVD of A and note the singular values of AG are the same as the singular values of ΣG_1 , where $G_1 = V^*G$ is a Gaussian embedding matrix. The columns of ΣG_1 can be viewed as r scaled observations from an n -dimensional $\mathcal{N}(0, \Sigma^2)$ distribution. As the number of observations tends to infinity, the singular values of the ‘data matrix’ ΣG_1 tend to the singular values of Σ . Estimation of (approximately) low-rank covariance matrix is studied in, among others, [157, 260, 263] and has a clear relation to PCA.

One specific related example in the statistics literature that is well studied is the spiked covariance matrix model, introduced by Johnstone [153] and analyzed by, among others, Bai and Silverman [18], Nadler [195] and Rao et al. [219]. We find that the theory in these works cannot be applied in our context, since the tail of the singular values in this model is very heavy. In the general context of covariance matrix estimation, various authors have suggested different manipulations to the sample covariance matrix or its eigenvalues to improve the estimator, known as shrinkage. See, for instance, the work by Ledoit and Wolf on linear shrinkage [161], nonlinear computational shrinkage [162]

and nonlinear analytical shrinkage [163], or by El Karoui based on random matrix theory [97]. Our numerical experiments suggest these methods are either inefficient for large-scale matrices or unsuitable for numerically low-rank matrices. One of the main obstacles is the numerical stability of these methods for spectra that span widely varying orders of magnitude, which is not a usual context in (high-dimensional) statistics. We discuss DOS-based alternatives for rank estimation in Section 3.6, but do not further consider any shrinkage-based approaches.

3.2.3 Related algorithms in (randomized) numerical linear algebra

A number of recent papers focus on randomized algorithms to perform numerical linear algebra tasks on large matrices. Specifically, randomized algorithms for either full or low-rank matrix factorization are of interest in the discussion of numerical rank estimation. This relevance is two-fold, as a numerical rank estimate can be a natural consequence of a factorization. In other cases, an a priori rank estimate can aid in performing a (low-rank) factorization.

The former is the case for rank-revealing full factorizations, such as rank-revealing QR decompositions [94] or UTV factorizations [177]. Full factorizations are (relatively speaking) most useful when the ϵ -rank is not much smaller than the matrix rank, and still require cubic $\mathcal{O}(mn\text{rank}(A))$ complexity. As we mainly consider the case where $\text{rank}_\epsilon(A) \ll \min(m, n)$, we focus on methods that have subcubic complexity.

A number of randomized algorithms for a low-rank factorization have been proposed and they can often be modified to yield a (rank r) QB factorization: $A \approx QB$, where $Q \in \mathbb{F}^{m \times r}$ is a matrix with orthonormal columns and $B \in \mathbb{F}^{r \times n}$ is a short and fat matrix. Singular value estimates can then be obtained by computing the exact singular values of B , which in turn leads to a numerical rank estimate. Within randomized algorithms for QB factorizations, we can distinguish between (adaptive) algorithms that focus on solving the fixed-precision problem and non-adaptive algorithms that focus on the fixed-rank problem.

The fixed-precision problem can be summarised as: find QB such that $\|A - QB\| \leq \epsilon\|A\|$. Adaptive algorithms for this problem include the adaptive randomized range finder [127], the incremental rangefinder [181], RandQB.b [182] and RandQB.EI [284]. The fixed-precision problem is discussed in more detail in Section 3.6.4.

Conversely, the fixed-rank problem is: find QB of exact rank r such that $\|A - QB\|$ is as small as possible. One of the most well-known algorithms in randomized fixed-rank factorization is randomized rangefinder [127], which is a core part of the randomized SVD algorithm proposed in [127].

We discuss these algorithms further, comparing with our algorithm, in Sections 3.5–3.6, to demonstrate that our algorithm is much more efficient for rank estimation, and can often be used as a convenient preprocessing step for a low-rank approximation algorithm to determine the input rank.

Conversely, a numerical rank estimate can aid the factorization process. This is discussed in detail in Section 3.6.4 and forms the basis for a non-adaptive low-rank factorization algorithm we propose for the fixed-precision problem.

3.3 Sketching roughly preserves singular values: $\sigma_i(AS)/\sigma_i(A) = \mathcal{O}(1)$ for leading i

In this section we show that if a matrix A has low numerical rank, then sketching preserves its leading singular values sufficiently well to provide an estimate for its numerical rank. Again drawing the connection to sample covariance matrix estimation, it is intuitive that it should be difficult, if not impossible, to obtain information about an n -dimensional distribution using only r samples, when $r \ll n$. As before, for a matrix $A = U\Sigma V^T$ and a Gaussian embedding $G \in \mathbb{R}^{n \times r}$, let $G_1 = V^T G$ be another Gaussian embedding matrix. The sketch AG and $Z = \Sigma G_1$ have the same singular values, and the columns of Z are r (scaled) samples from an $\mathcal{N}(0, \Sigma^2)$ distribution. In general, one would need $\mathcal{O}(n)$ samples to estimate the covariance matrix Σ^2 [260]. However, if the covariance matrix is approximately low-rank, the number of samples need simply be proportional to the stable rank of Σ , which is $\|\Sigma\|_F^2 / \|\Sigma\|_2^2$ (see also [260, Sec. 9.2.3]). This provides us with some intuition, as to why we can get away with estimating (the dominant part of) Σ with much less than n samples.

We make the connection between the low-rank structure of A and our ability to estimate its singular values using AS explicit with the deterministic and probabilistic error bounds on the ratio $\sigma_i(AS)/\sigma_i(A)$. The analysis will focus on Gaussian matrices. As explained before, $G \in \mathbb{R}^{n \times r}$ denotes a Gaussian matrix with iid $\mathcal{N}(0, 1)$ ($\mathbb{F} = \mathbb{R}$) or $\mathcal{CN}(0, 1)$ ($\mathbb{F} = \mathbb{C}$) entries, and a Gaussian embedding matrix will be of the form $S = G/\sqrt{r}$. We start from the unscaled sketch AG . Let $A = U\Sigma V^*$ be the SVD of A and decompose this further to $A = U_1\Sigma_1V_1^* + U_2\Sigma_2V_2^*$, where U_1 contains the leading r singular vectors of A . Now decompose $AG \in \mathbb{F}^{m \times r}$ as

$$AG = U_1\Sigma_1(V_1^*G) + U_2\Sigma_2(V_2^*G) = U_1\Sigma_1G_1 + U_2\Sigma_2G_2, \quad (3.1)$$

where Σ_1 is $r \times r$ and $G_1 \in \mathbb{F}^{r \times r}$ and $G_2 \in \mathbb{F}^{(n-r) \times r}$ are independent Gaussian matrices. Define $B_1 = U_1\Sigma_1G_1 \in \mathbb{F}^{m \times r}$ and $B_2 = U_2\Sigma_2G_2 \in \mathbb{F}^{m \times r}$. We start with the following result, which examines the relation between $\sigma_i(AG)$ and $\sigma_i(A)$.

Lemma 3.2. *Let $A \in \mathbb{F}^{m \times n}$ and $G \in \mathbb{F}^{n \times r}$. Decompose AG as in (3.1). Then, for $i = 1, \dots, r$,*

$$\sigma_{\min}(\hat{G}_{\{i\}}) \leq \frac{\sigma_i(AG)}{\sigma_i(A)} \leq \sqrt{\sigma_{\max}(\tilde{G}_{\{r-i+1\}})^2 + \left(\frac{\sigma_{r+1}(A)\sigma_{\max}(G_2)}{\sigma_i(A)} \right)^2}, \quad (3.2)$$

where $\hat{G}_{\{i\}}$ is an $i \times r$ matrix consisting of the first i rows of G_1 , and $\tilde{G}_{\{r-i+1\}}$ is the matrix of the last $r - i + 1$ rows of G_1 . Furthermore, if G is a Gaussian matrix, then for each $i \in \{1, \dots, r\}$, the random matrices $\hat{G}_{\{i\}}$, $\tilde{G}_{\{r-i+1\}}$, and G_2 are independent Gaussian in $\mathbb{F}$.

Proof. The proof consists of establishing the identities

$$1 \leq \frac{\sigma_i(AG)}{\sigma_i(B_1)} \leq \sqrt{1 + \left(\frac{\sigma_{r+1}(A)\sigma_{\max}(G_2)}{\sigma_i(B_1)} \right)^2}, \quad (3.3)$$

$$\sigma_{\min}(\hat{G}_{\{i\}}) \leq \frac{\sigma_i(B_1)}{\sigma_i(A)} \leq \sigma_{\max}(\tilde{G}_{\{r-i+1\}}), \quad (3.4)$$

for $i = 1, \dots, r$.

It is immediate from the definitions of B_1 and B_2 that $(AG)^*(AG) = B_1^*B_1 + B_2^*B_2$. As a result, we have by Weyl's theorem, for $i = 1, \dots, r$,

$$\sigma_i(B_1)^2 + \sigma_{\min}(B_2)^2 \leq \sigma_i(AG)^2 \leq \sigma_i(B_1)^2 + \|B_2\|^2. \quad (3.5)$$

The upper bound follows from observing $\|B_2\| \leq \|\Sigma_2\|\|G_2\| = \sigma_{r+1}(A)\sigma_{\max}(G_2)$. The lower bound is immediate.

We use the interlacing property of singular values to show (3.4). For a square matrix $B \in \mathbb{F}^{k \times k}$, let $B_{[-p]} \in \mathbb{F}^{k \times (k-p)}$ denote a submatrix of B obtained by deleting any p columns. Then [183]

$$\sigma_j(B) \leq \sigma(B_{[-p]}) \leq \sigma_{j+p}(B), \quad \text{for } j = 1, 2, \dots, k-p. \quad (3.6)$$

First, notice the upper bound (3.4) is trivial for $i = 1$. For $i = 2, \dots, r$, partition $G_1^*\Sigma_1$ as

$$G_1^*\Sigma_1 = \begin{bmatrix} \hat{G}_{\{i-1\}}^*\Sigma_{11} & \tilde{G}_{\{r-i+1\}}^*\Sigma_{12} \end{bmatrix},$$

where $\Sigma_{11} \in \mathbb{R}^{(i-1) \times (i-1)}$ and $\Sigma_{12} \in \mathbb{R}^{(r-i+1) \times (r-i+1)}$ are diagonal matrices which contain the singular values of Σ_1 . We then have

$$\begin{aligned} \sigma_i(B_1) &= \sigma_i \left(\begin{bmatrix} \hat{G}_{\{i-1\}}^*\Sigma_{11} & \tilde{G}_{\{r-i+1\}}^*\Sigma_{12} \end{bmatrix} \right) \\ &\leq \sigma_1(\tilde{G}_{\{r-i+1\}}^*\Sigma_{12}) \\ &\leq \|\Sigma_{12}\|_2 \|\tilde{G}_{\{r-i+1\}}\|_2 = \sigma_i(A) \|\tilde{G}_{\{r-i+1\}}\|_2, \end{aligned}$$

where we applied (3.6) with $j = 1$ and $p = i - 1$ for the first inequality. Similarly, for the lower

bound and $i = 1, \dots, r-1$, consider the partition

$$G_1^* \Sigma_1 = \begin{bmatrix} \hat{G}_{\{i\}}^* \Sigma_{11} & \tilde{G}_{\{r-i\}}^* \Sigma_{12} \end{bmatrix},$$

where now Σ_{11} is $i \times i$ and Σ_{12} is $(r-i) \times (r-i)$. We have

$$\begin{aligned} \sigma_i(B_1) &= \sigma_i([\hat{G}_{\{i\}}^* \Sigma_{11} \quad \tilde{G}_{\{r-i\}}^* \Sigma_{12}]) \\ &\geq \sigma_i(\hat{G}_{\{i\}}^* \Sigma_{11}) \\ &\geq \sigma_{\{i\}}(\Sigma_{11}) \sigma_{\min}(\hat{G}_{\{i\}}) = \sigma_i(A) \sigma_{\min}(\hat{G}_{\{i\}}). \end{aligned}$$

The lower bound is immediate for $i = r$.

If G is Gaussian, its orthogonal invariance implies $\begin{bmatrix} V_1^* \\ V_2^* \end{bmatrix} G$ is also Gaussian. Since $\hat{G}_{\{i\}}$, $\tilde{G}_{\{r-i+1\}}$, and G_2 are submatrices of this matrix with no overlap for a fixed i , it immediately follows that they are Gaussian and independent. $\square$

It is worth noting that (3.2) holds without the assumption that G is Gaussian.

Lemma 3.2 motivates us to look at the singular values of independent Gaussian matrices $\hat{G}_{\{i\}}$, $\tilde{G}_{\{r-i+1\}}$ and G_2 . To this end we use two classic results from random matrix theory to derive probabilistic bounds on the expected error and on the failure probability.

Theorem 3.3 (Marčenko and Pastur [175], Davidson and Szarek [72], Aubrun and Szarek [11]).

Let $G \in \mathbb{R}^{m \times n}$ be a standard Gaussian matrix with $m \geq n$. Then

$$\sqrt{m} - \sqrt{n} \leq \mathbb{E} \sigma_{\min}(G) \leq \mathbb{E} \sigma_{\max}(G) \leq \sqrt{m} + \sqrt{n}. \quad (3.7)$$

Furthermore, for every $t \geq 0$ one has

$$\mathbb{P}\{\sigma_{\min}(G) \leq \sqrt{m} - \sqrt{n} - t\} \leq e^{-t^2/2}, \quad \mathbb{P}\{\sigma_{\max}(G) \geq \sqrt{m} + \sqrt{n} + t\} \leq e^{-t^2/2}. \quad (3.8)$$

Let $G \in \mathbb{C}^{m \times n}$ be a complex Gaussian matrix with $m \geq n$. Then, for every $t \geq 0$ one has

$$\mathbb{P}\{\sigma_{\max}(G) \geq \sqrt{m} + \sqrt{n} + t\} \leq \frac{1}{2} e^{-t^2}. \quad (3.9)$$

Furthermore, for every $t > 4\sqrt{2 \log n}/(\sqrt{m/n} - 1)$,

$$\mathbb{P}\{\sigma_{\min}(G) \leq \sqrt{m} - \sqrt{n} - t\} \leq e^{-t^2/4}. \quad (3.10)$$

A useful way to interpret these results in simple terms is that rectangular random (Gaussian) matrices with aspect ratio $m/n > 1$ are well-conditioned, with singular values supported in $[\sqrt{m} - \sqrt{n}, \sqrt{m} + \sqrt{n}]$. We note that (3.7), which follows from the Marčenko and Pastur rule [175], holds more generally for random matrices with iid entries with mean 0 and variance 1. The second pair of bounds (3.8) do not generally hold for other random matrices, see [223] for details. As far as the author is aware, there does not exist a lower bound on the expectation of the smallest singular value of a complex normal matrix. An upper bound for the largest singular value can be found in [11, Prop. 6.33]. We use these results to bound the singular values of $\hat{G}_{\{i\}}$, $\tilde{G}_{\{r-i+1\}}$, and G_2 , which leads to the following theorem. We present the result only in the real case, but it can easily be extended to the complex case by combining Lemma 3.2 with (3.9) and (3.10).

Theorem 3.4. *Let $S \in \mathbb{R}^{n \times r}$ be a Gaussian embedding matrix, i.e., $S = G/\sqrt{r}$, and let $A \in \mathbb{R}^{m \times n}$, where $m \geq n$. Then, for $i = 1, \dots, r$*

$$1 - \sqrt{\frac{i}{r}} \leq \mathbb{E} \frac{\sigma_i(AS)}{\sigma_i(A)} \leq 1 + \sqrt{\frac{r-i+1}{r}} + \frac{\sigma_{r+1}(A)}{\sigma_i(A)} \left(1 + \sqrt{\frac{n-r}{r}}\right). \quad (3.11)$$

Additionally, for each $i = 1, \dots, r$ we have, with failure probability at most $3e^{-t^2/2}$,

$$\begin{aligned} 1 - \sqrt{\frac{i}{r}} - \frac{t}{\sqrt{r}} &\leq \frac{\sigma_i(AS)}{\sigma_i(A)} \\ &\leq 1 + \sqrt{\frac{r-i+1}{r}} + \frac{\sigma_{r+1}(A)}{\sigma_i(A)} \left(1 + \sqrt{\frac{n-r}{r}}\right) + \frac{t}{\sqrt{r}} \left(1 + \frac{\sigma_{r+1}(A)}{\sigma_i(A)}\right). \end{aligned} \quad (3.12)$$

Proof. Note first from Marčenko-Pastur (3.7) that we have

$$\begin{aligned} \mathbb{E} \sigma_{\min}(\hat{G}_{\{i\}}) &\geq \sqrt{r} - \sqrt{i}; \\ \mathbb{E} \sigma_{\max}(\tilde{G}_{\{r-i+1\}}) &\leq \sqrt{r} + \sqrt{r-i+1}; \\ \mathbb{E} \sigma_{\max}(G_2) &\leq \sqrt{n-r} + \sqrt{r}. \end{aligned}$$

We can combine this with the result of Lemma 3.2 to find

$$\begin{aligned}
\mathbb{E} \frac{\sigma_i(AG)}{\sigma_i(A)} &\leq \mathbb{E} \left[\sigma_{\max}(\tilde{G}_{\{r-i+1\}})^2 + \left(\frac{\sigma_{r+1}(A) \sigma_{\max}(G_2)}{\sigma_i(A)} \right)^2 \right]^{1/2} \\
&\leq \mathbb{E} \left[\sigma_{\max}(\tilde{G}_{\{r-i+1\}}) + \frac{\sigma_{r+1}(A) \sigma_{\max}(G_2)}{\sigma_i(A)} \right] \\
&= \mathbb{E} \sigma_{\max}(\tilde{G}_{\{r-i+1\}}) + \frac{\sigma_{r+1}(A)}{\sigma_i(A)} \mathbb{E} \sigma_{\max}(G_2) \\
&\leq \sqrt{r} + \sqrt{r-i+1} + \frac{\sigma_{r+1}(A)}{\sigma_i(A)} (\sqrt{n-r} + \sqrt{r}),
\end{aligned}$$

and

$$\mathbb{E} \frac{\sigma_i(AG)}{\sigma_i(A)} \geq \mathbb{E} \sigma_{\min}(\hat{G}_{\{i\}}) \geq \sqrt{r} - \sqrt{i}.$$

The first result of Theorem 3.4 then follows by applying the scaling $1/\sqrt{r}$. Similarly, by (3.8) we have with failure probability at most $3e^{-t^2/2}$ that the following statements hold simultaneously:

$$\begin{aligned}
\frac{\sigma_{\min}(\hat{G}_{\{i\}})}{\sqrt{r}} &\geq 1 - \sqrt{\frac{i}{r}} - \frac{t}{\sqrt{r}}, \\
\frac{\sigma_{\max}(\tilde{G}_{\{r-i+1\}})}{\sqrt{r}} &\leq 1 + \sqrt{\frac{r-i+1}{r}} + \frac{t}{\sqrt{r}}, \\
\frac{\sigma_{r+1}}{\sigma_i} \frac{\sigma_{\max}(G_2)}{\sqrt{r}} &\leq \frac{\sigma_{r+1}(A)}{\sigma_i(A)} \left(1 + \sqrt{\frac{n-r}{r}} \right) + \frac{\sigma_{r+1}(A)}{\sigma_i(A)} \frac{t}{\sqrt{r}}.
\end{aligned}$$

Since the deterministic results in Lemma 3.2 imply

$$\frac{\sigma_{\min}(\hat{G}_{\{i\}})}{\sqrt{r}} \leq \frac{\sigma_i(AG)}{\sigma_i(A)} \leq \frac{\sigma_{\max}(\tilde{G}_{\{r-i+1\}})}{\sqrt{r}} + \frac{\sigma_{r+1}(A)}{\sigma_i(A)} \frac{\sigma_{\max}(G_2)}{\sqrt{r}},$$

the result follows. $\square$

3.3.1 Bounds for general subspace embeddings

The results above focus on Gaussian embedding matrices, and show that the ratios of the singular values $\sigma_i(AS)/\sigma_i(A)$ are reasonably close to 1, as specified by (3.11). We now show that much of this carries over to a general subspace embedding S . We let $S \in \mathbb{F}^{n \times s}$ be of general embedding dimension s necessary to embed an r -dimensional subspace.

Theorem 3.5. *Let $V_1 \in \mathbb{F}^{n \times r}$ be the leading r right singular vectors of $A \in \mathbb{F}^{m \times n}$, and suppose $S \in \mathbb{F}^{n \times s}$ is a subspace embedding for the subspace V_1 satisfying $\sigma_i(V_1^T S) \in [1 - \varepsilon, 1 + \varepsilon]$ for some*

$\varepsilon < 1$. Then, for $i = 1, \dots, r$,

$$1 - \varepsilon \leq \frac{\sigma_i(AS)}{\sigma_i(A)} \leq \sqrt{(1 + \varepsilon)^2 + \left(\frac{\sigma_{r+1}(A) \|S\|_2}{\sigma_i(A)} \right)^2}. \quad (3.13)$$

Proof. We start with a variant of (3.2), which follows from the same arguments:

$$\sigma_{\min}(\hat{S}_{\{i\}}) \leq \frac{\sigma_i(AS)}{\sigma_i(A)} \leq \sqrt{\sigma_{\max}(\tilde{S}_{\{r-i+1\}})^2 + \left(\frac{\sigma_{r+1}(A) \sigma_{\max}(S_2)}{\sigma_i(A)} \right)^2}, \quad i = 1, \dots, r.$$

Here $\hat{S}_{\{i\}}, \tilde{S}_{\{r-i+1\}}$ are submatrices of $V_1^T S$ (as in Lemma 3.2), and $S_2 = V_2^T S$, where $V_2 \in \mathbb{F}^{n \times (n-r)}$ are the trailing right singular vectors of A . By the assumption $\sigma_i(V_1^T S) \in [1 - \varepsilon, 1 + \varepsilon]$, we have

$$\sigma_{\min}(\hat{S}_{\{i\}}) \geq \sigma_{\min}(V_1^T S) \geq 1 - \varepsilon,$$

which gives the lower bound in (3.13), and also

$$\sigma_{\max}(\tilde{S}_{\{r-i+1\}}) \leq \sigma_{\max}(V_1^T S) \leq 1 + \varepsilon.$$

Finally, $\sigma_{\max}(S_2) = \|V_2^T S\|_2 \leq \|S\|_2$ to obtain the upper bound. $\square$

We note that the $\|S\|_2$ term in (3.13) is bounded by $\sqrt{n/s}$, often deterministically, in most commonly-used embeddings (see, e.g. (2.5)). This is roughly equal to $\sqrt{\frac{n-r}{r}}$ appearing in (3.4). Thus Theorems 3.4 and 3.5 offer roughly the same level of guarantee in terms of the quality of $\sigma_i(AS)$ as an estimate for $\sigma_i(A)$.

3.3.2 The effect of tail singular values and oversampling

The proof of Lemma 3.2 allows us to see how the tail of the singular values, Σ_2 , affects the lower bound on the ratio $\sigma_i(AG)/\sigma_i(A)$. This allows us to make a case for oversampling based on singular value estimation, whereas usually oversampling is inspired through a discussion on singular vectors. Again using the Courant-Fischer theorem, we have

$$\sigma_i(AG) = \max_{\substack{S \subset \mathbb{F}^r \\ \dim S = i}} \min_{\substack{x \in S \\ \|x\|=1}} \left\| \begin{bmatrix} \Sigma_1 G_1 \\ \Sigma_2 G_2 \end{bmatrix} x \right\|.$$

Now let $\hat{\mathcal{S}} \subset \mathbb{F}^r$ and $\hat{x} \in \hat{\mathcal{S}}$ be such that

$$\max_{\substack{\mathcal{S} \subset \mathbb{F}^r \\ \dim \mathcal{S} = i}} \min_{\substack{x \in \mathcal{S} \\ \|x\|=1}} \|\Sigma_1 G_1 x\| = \min_{\substack{x \in \hat{\mathcal{S}} \\ \|x\|=1}} \|\Sigma_1 G_1 x\| = \|\Sigma_1 G_1 \hat{x}\| = \|B_1 \hat{x}\| = \sigma_i(B_1).$$

Then $\sigma_i(AG) \geq \left\| \begin{bmatrix} \Sigma_1 G_1 & \Sigma_2 G_2 \end{bmatrix}^T \hat{x} \right\| = \sqrt{\|B_1 \hat{x}\|^2 + \|\Sigma_2 G_2 \hat{x}\|^2}$. We can use this to show

$$\frac{\sigma_i(AS)}{\sigma_i(A)} = \frac{\sigma_i(AG)}{\sigma_i(B_1)} \frac{\sigma_i(B_1)}{\sigma_i(A)\sqrt{r}} \geq \frac{\sigma_{\min}(\hat{G}_{\{i\}})}{\sqrt{r}} \sqrt{1 + \frac{\|\Sigma_2 G_2 \hat{x}\|^2}{\sigma_i(B_1)^2}}. \quad (3.14)$$

As $G_2 \hat{x}$ is a Gaussian vector, we know $\mathbb{E}\|\Sigma_2 G_2 \hat{x}\| = \|\Sigma_2\|_F$. Additionally, it is worthwhile to note $\sigma_{\min}(\hat{G}_{\{i\}})/\sqrt{r} \approx 1 - \sqrt{i/r}$ in expectation. The lower bound (3.14) thus suggests that a heavier tail Σ_2 , in terms of $\|\Sigma_2\|_F$, would result in larger values for $\sigma_i(AS)$ than a light tail. Consider in particular the last few singular values of AS , where the term $\sigma_{\min}(\hat{G}_{\{i\}})$ is small. If the tail is light too, i.e. $\|\Sigma_2 G_2 \hat{x}\|$ is small, the lower bound will not be strong. We see in practice that this effect is indeed noticeable for the smallest singular values of AS , making them less reliable estimators. The bulk of the singular values of AS are, however, not affected by the size of the tail given the low-rank structure of the matrix. We display this effect in Figures 3.1 and 3.2 with small-scale experiments.

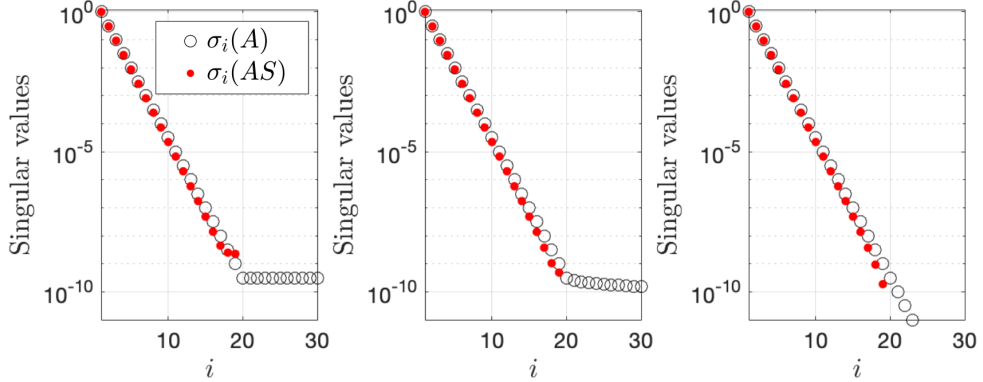


Figure 3.1: How the tail of the singular value spectrum influences the singular value estimates. Each of the graphs shows the first 30 singular values $\sigma_i(A)$ of a square matrix of dimension $n = 1000$. The first 20 singular values are identical for each of the matrices. The matrices in the graphs have constant, slow polynomially decreasing exponentially decreasing tails (from left to right). We sketch with a Gaussian embedding matrix of size $r = 19$. Although $\sigma_{r+1} = \sigma_{20}$ is the same for each of the matrices, we see the tail affects (only) the last few estimates. The results shown here are the average of 1000 trials, yet the behaviour is very typical.

The small-scale experiments agree with bigger numerical results, as well as with theoretical results, in the sense that the extreme (smallest) singular value estimates are not trustworthy. For this reason we recommend oversampling by 10% and disregarding the singular values corresponding to the oversampling.

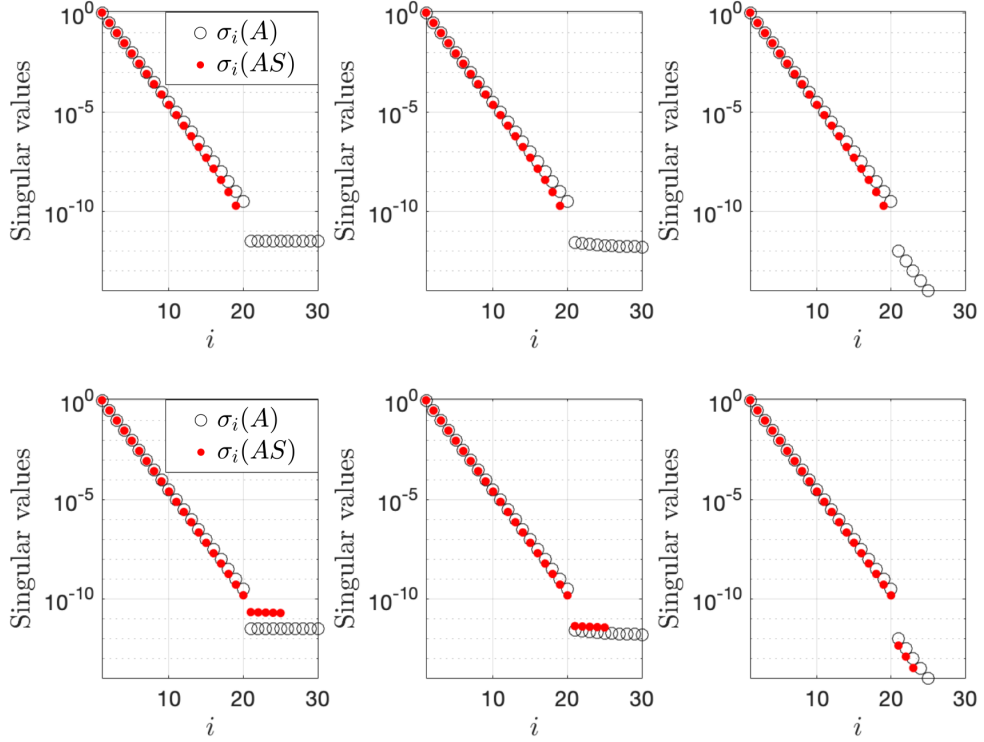


Figure 3.2: One could argue that in none of the cases discussed in Figure 3.1 is the numerical rank very well-defined, because there is no clear gap between any of the singular values. We repeat the experiment of Figure 3.1 but with slightly different singular values. The top row shows the estimates of the first 19 singular values when there is a gap between the 20th and 21st singular value of A . We see that the effects of different tails becomes negligible. In the bottom row we repeat the experiment with $r = 25$ instead of $r = 19$. This illustrates that we can detect the gap in each of the cases.

3.3.2.1 The effect of a heavy tail

The experiments in Figures 3.1 and 3.2 investigate tails of matrices with very small numerical rank and a very rapidly decaying singular value spectrum. Let us briefly investigate the performance of the singular values of AS as estimates for $\sigma_i(A)$ when the decay is much slower. We repeat the same experiments as before, but the first 20 singular values of $A \in \mathbb{R}^{1000 \times 1000}$ are between 1 and 0.01. The tails are again constant, polynomially decaying, and exponentially decaying with Frobenius norms equal to 0.313, 0.0603, and 0.0351 respectively. The results are shown in Figure 3.3.

We first note the top row of Figure 3.3; the tail effects are clearly visible. Especially for the (very heavy) flat tail, the last few singular values of AS are significant overestimates for the singular values of A , as is predicted by (3.14). However, it is noteworthy that for the matrices with slow polynomially and exponentially decaying tails – both also very heavy – the singular values are still good order-of-magnitude estimates. In the middle row, we see that the lighter tails now cause an underestimate in $\sigma_i(AS)$. This effect is diminished when we estimate more singular values, and

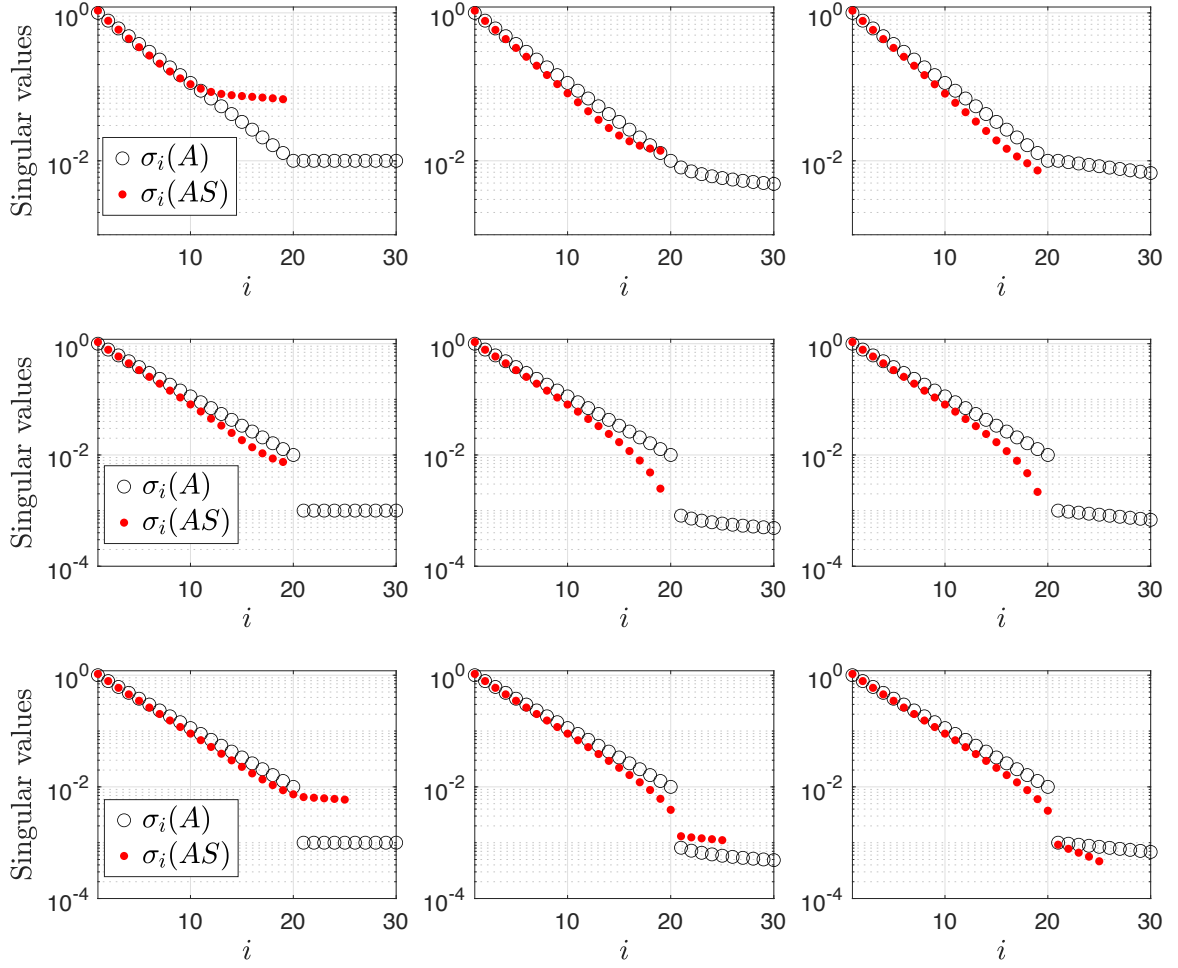


Figure 3.3: The effect of heavy tails on the singular values of AS . The matrices in the top row do not have a significant gap in the spectrum, whereas the matrices in the middle and bottom row do. In the top and middle row, we embedding dimension of S is 19, whereas in the bottom row we estimate the first 24 singular values. On the left: a flat tail; in the middle column: a slow polynomially decaying tail; and on the right: an exponentially decaying tail.

indeed we can identify the gap except for the flat tail.

These small-scale experiments show that the singular values of AS are good estimates for the leading singular values of A provided that the tail singular values (those that we do not estimate) are small relatively to the leading singular values. Extremely heavy tails, like in the case of the flat spectrum, significantly affect the quality of the approximations. In particular, we require $\|\Sigma_2\|_F$ to be small compared to the singular values we estimate, otherwise we cannot guarantee order-of-magnitude estimates (as predicted in (3.14)). Our further experiments will only consider such matrices.

3.3.3 Choice of sketch matrix

In our analysis we have mainly focused on Gaussian embedding matrices, as they are the most well-studied class of random matrices and allow us to derive sharp constants. A Gaussian embedding matrix is a random matrix with iid entries $\mathcal{N}(0, 1/r)$. They are often the simplest sketch to implement. While it requires $O(mnr)$ operations to compute AS , when $r = O(1)$ the complexity is optimal as mn is a lower bound for dense A , as clearly all entries of A need to be read. In fact, Gaussian sketches can be among the fastest to execute when $r = O(1)$. Furthermore, Gaussian is the most efficient type of embedding in the sense that an embedding for an r -dimensional subspace can be achieved with high probability using $\mathcal{O}(r)$ samples.

Otherwise, if $r \gg 1$, other, more structured, classes of sketch matrices have been suggested and employed to speed up the computation of AS . These include subsampled randomized trigonometric transforms (SRTTs) [181], hashed randomized trigonometric transforms (HRTTs) [47], and sparse matrix embeddings [61, 63, 154] (see Section 2.1.3).

In particular, SRTTs such as the subsampled randomized Hadamard, Fourier, discrete cosine or discrete Hartley transforms are random embeddings that allow fast application to a matrix. The use of these sketches can be justified by Theorem 3.5. While they come with weaker theoretical guarantees than Gaussian embeddings [251] (in that the size of the sketch needs to be at least $\mathcal{O}(r \log r)$ to ensure an embedding with high probability), empirical evidence strongly suggests that they usually perform just as well. The performance of SRTT matrices is discussed more in the next section.

Related to SRTTs are the recently introduced class of random embeddings, HRTTs [47]. In these works it is shown an embedding for an r -dimensional subspace can be achieved with high probability using $\mathcal{O}(r)$ samples (as with Gaussians) rather than $\mathcal{O}(r \log r)$, by replacing the subsampling matrix S in (2.5) with a hashing matrix (e.g. Countsketch [275]), which can be done without increasing the complexity. Computational results suggest HRTTs perform adequately in pathological cases where SRTTs fail, such as diagonal matrices (see discussions on coherence in [46, 146]).

When $A \in \mathbb{F}^{m \times n}$ has no structure to take advantage of (e.g. dense), SRTT embeddings allow us to compute AS with $O(mn \log n)$ operations, which is optimal up to an $O(\log n)$ factor. When A has structure such as sparsity, AS could be computed much more efficiently, for example in $O(\text{nnz}(A)r)$ operations for a Gaussian embedding and less for sparse sketches.

The choice of S therefore depends on the structure of A ; but the complexity of the sketching AS can be bounded from above by $O(mn \log n)$ for dense matrices.

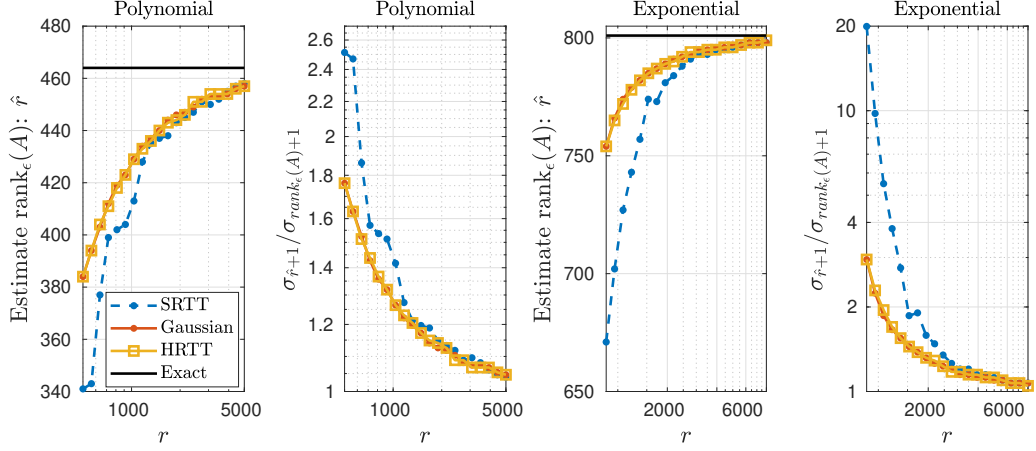


Figure 3.4: Numerical rank estimation of matrices with polynomially decaying singular values and exponentially decaying singular values, where only the sketch $AS \in \mathbb{R}^{m \times r}$ is used. The matrices are real and square diagonal of dimension $n = 10^5$. The results are shown when using a subsampled randomized DCT embedding (SRTT), a Gaussian embedding matrix (Gaussian) and a hashed randomized DCT embedding (HRTT). The hashing matrix in the HRTT has one non-zero per column (1-hashing). The black lines indicate the exact numerical rank, $\text{rank}_\epsilon(A)$.

3.3.4 A numerical experiment

We show how the theoretical result of this section, namely $\sigma_i(AS)/\sigma_i(A) = \mathcal{O}(1)$ for leading i , practically translates to a rank estimation technique. After sketching with an $n \times r$ embedding matrix S , we compute the singular values $\sigma_1(AS), \dots, \sigma_r(AS)$ and estimate $\text{rank}_\epsilon(A)$ for a given tolerance ϵ to be the first $\hat{r}$ such that $\sigma_{\hat{r}}(AS) > \epsilon \|A\| \geq \sigma_{\hat{r}+1}(AS)$. If (an estimate of) $\|A\|$ is unavailable, we may use $\sigma_1(AS)$. We applied this method to two synthetic example matrices, one diagonal matrix with polynomially decaying singular values ($\sigma_i(A) = i^{-3}$), and one diagonal matrix with exponentially decaying singular values ($\sigma_i(A) = 10^{-0.01(i-1)}$). The matrices are square of dimension $n = 10^5$.

Figure 3.4 shows the results of the experiment. We plot the estimates for the numerical rank $\hat{r}$ that result from varying embedding dimensions r , and the ratio $\sigma_{\hat{r}+1}(A)/\sigma_{\text{rank}_\epsilon(A)+1}$. As both matrices do not have large gaps in the spectrum, the latter is an important quantity to judge the effectiveness of the rank estimator (see Section 3.1.1). We see that even for small values of r and when using an HRTT or Gaussian embedding, $\sigma_{\hat{r}+1}(A)$ is very close to $\sigma_{\text{rank}_\epsilon(A)+1}$ and ϵ and well within the acceptable range described in Section 1. Additionally, we see that SRTT matrices perform less than optimal for this pathological example of a diagonal matrix.

3.4 Preserving all singular values with embedding:

$$\sigma_i(S_2AS_1)/\sigma_i(AS_1) = \mathcal{O}(1)$$

We have seen that the numerical rank of AS_1 is a good estimator for the numerical rank of A , provided that A is approximately low-rank. Although AS_1 is of much smaller dimension than A , it may still be prohibitively expensive to compute its exact singular values. Furthermore, one could question whether it is worth calculating the singular values accurately spending $\mathcal{O}(mr^2)$ work, considering the accuracy that was lost in the sketching step.

We propose to sketch again, now from the left. Subspace embeddings specifically preserve the singular values of tall and skinny matrices, as specified in Section 2.1.3. If $S \in \mathbb{F}^{k \times m}$ is a subspace embedding for the column space of $B \in \mathbb{R}^{m \times r}$ with distortion ε , we have

$$(1 - \varepsilon)\sigma_{\min}(B) \leq \sigma_{\min}(SB) \leq \sigma_{\max}(SB) \leq (1 + \varepsilon)\sigma_{\max}(B).$$

It follows that, in our context, we may estimate the singular values of AS_1 using the singular values of S_2AS_1 , where S_2 is another random embedding (independent of S_1) of size $r_2 \times m$. It is then only necessary to compute the numerical rank (by computing the exact singular values) of the very small matrix $S_2AS_1 \in \mathbb{F}^{r_2 \times r_1}$ to estimate the numerical rank of the large matrix A . This is the final step of our rank estimation algorithm, which is discussed fully in the next section.

As for the choice of S_2 : unlike the first sketch of computing AS_1 , in most cases the structure of A (such as sparsity, if present) is lost in $B = AS_1$. It is therefore usually recommended that we take S_2 to be an SRTT embedding, so that $S_2B = \Theta B$ can be computed in $\mathcal{O}(mr \log m)$ operations. We note most importantly that the majority of the inaccuracy in the singular value estimates results from the first sketch AS_1 since approximating the singular values of AS_1 using the singular values of S_2AS_1 is inherently a much easier task. For this reason we argue it is not strictly necessary that the second sketching matrix would be, for instance, an HRTT. This is confirmed by the numerical experiment in Figure 3.5.

We assume the choice for an SRTT below and throughout the remainder of the chapter, switching between the notations S_2AS_1 and ΘAS_1 when appropriate.

Specifically the subsampled randomized Hadamard transform is analyzed extensively in [37, 251]. These results can be readily extended to a general SRTT matrix and used in our context, which we prove here. They are essentially restatements of [251, Thm. 3.1] and [37, Lem. 4.1]. It leads to the following result on the accuracy of $\sigma_i(S_2AS_1)$ as estimates for $\sigma_i(AS_1)$.

Corollary 3.6. *Let $A \in \mathbb{R}^{m \times n}$ and let $\Theta \in \mathbb{R}^{s \times m}$, $n \leq s < m$, be an SRTT matrix as in (2.5), where the trigonometric transform F in Θ satisfies $\eta = m \max |F_{ij}|^2$. Let $0 < \tilde{\varepsilon} < 1/3$ and $0 < \delta < 1$.*

If

$$6\eta\tilde{\epsilon}^{-2} \left[\sqrt{n} + \sqrt{8\log(m/\delta)} \right]^2 \log(n/\delta) \leq s \leq m, \quad (3.15)$$

then, with failure probability at most 3δ

$$\sqrt{1-\tilde{\epsilon}} \leq \frac{\sigma_i(\Theta A)}{\sigma_i(A)} \leq \sqrt{1+\tilde{\epsilon}}, \quad (3.16)$$

for each $i = 1, \dots, n$.

We wish to prove the following lemma from which Corollary 3.6 follows in a straightforward fashion.

Lemma 3.7. *Let $U \in \mathbb{R}^{m \times n}$ have orthonormal columns and let $S \in \mathbb{R}^{r \times m}$ be a SRTT matrix as in (2.5), where the trigonometric transform F in Θ satisfies $\eta = m \max |F_{ij}|^2$. Let $0 < \epsilon < 1/3$ and $0 < \delta < 1$. If*

$$6\eta\epsilon^{-2} \left[\sqrt{n} + \sqrt{8\log(m/\delta)} \right]^2 \log(n/\delta) \leq r \leq m, \quad (3.17)$$

then, with failure probability at most 3δ

$$\sqrt{1-\epsilon} \leq \sigma_{\min}(SU) \leq \sigma_{\max}(SU) \leq \sqrt{1+\epsilon}. \quad (3.18)$$

The use of the number $\eta = m \max |F_{ij}|^2$ in Lemma 3.7 is a natural choice as it only depends on the choice of trigonometric transform, not on the size of the matrices involved. Choices for the trigonometric transform include Fourier, cosine, Hadamard, and Hartley transforms. The optimal η is attained for a Hadamard transform, for which $\eta = 1$. The same holds for a Fourier matrix, but this involves operations in $\mathbb{C}$ and when A is real, we may prefer real transforms. Since the Hadamard transform is only available when the dimension m is a power of 2 (for which one solution is to append zeros to the matrix), we make use of the discrete cosine transform with $\eta = 2$ in our experiments when A is real. Alternatively, we could have chosen the discrete Hartley transform with the same coherence $\eta = 2$. See [14] for more discussion.

The proof of Lemma 3.7 consists of two main steps: first to show that premultiplying an orthonormal matrix with a randomized trigonometric transform smooths out information in the rows (or equivalently, lowers the coherence), and second to show that sampling rows uniformly from a matrix with small coherence works well. The proofs are analogous to those by Boutsidis and Gittens [37], but as their discussion only covers subsampled randomized Hadamard transforms instead of general SRTT matrices, we include the proof for completeness.

A key concept in the proof is coherence, which was defined in Definition 2.2. For a tall and skinny matrix A and an orthogonal matrix U spanning its column space of A , the coherence of A is

defined $\mu(A) = \max \|U_{(i)}\|_2^2$ where $U_{(i)}$ denotes the i th row of U . The following lemma essentially states that applying a randomized trigonometric transform reduces the coherence of a matrix.

Lemma 3.8 (Tropp [251]). *Let $U \in \mathbb{F}^{m \times r}$ have orthonormal columns. Let F be a square orthogonal matrix of dimension m with $\eta = m \max |F_{ij}|^2$ and let D be a $m \times m$ diagonal matrix of independent random signs. Let $0 < \delta < 1$. Then, with probability at least $1 - \delta$*

$$\mu(FDU) \leq \frac{\eta}{m} \left(\sqrt{r} + \sqrt{8 \log(m/\delta)} \right)^2.$$

Recall that $\mu(FDU) \geq r/m$, which is the leading order term here. As a result, the coherence of FDU is close to the minimal with high probability. We use the following proposition in the proof of Lemma 3.8.

Proposition 3.9 (Tropp [251], Prop. 2.1). *Suppose f is a convex function that satisfies the Lipschitz condition*

$$|f(x) - f(y)| \leq L \|x - y\| \quad \forall x, y.$$

Let X be a vector of random signs. Then for all $t \geq 0$,

$$\mathbb{P}\{f(X) \geq \mathbb{E}f(X) + Lt\} \leq e^{-t^2/8}.$$

The proof of Lemma 3.8 then follows from defining such a convex, Lipschitz function.

Proof of Lemma 3.8 [251]. Fix a row index $j \in \{1, 2, \dots, m\}$ and define the function

$$f(x) := \|e_j^* F \text{diag}(x) U\| = \|x^* E U\|,$$

where $E = \text{diag}(e_j^* F)$ is a diagonal matrix constructed from the j th row of F . Let X be a vector of random signs. Since the diagonal of D is also a vector of random signs, we can bound the norm of the j th row of FDU by bounding $f(X)$ using Proposition 3.9. We will first show that f satisfies the conditions of Proposition 3.9. Note that it is convex by the triangle inequality. Since we have that $|F_{ij}| \leq \sqrt{\eta/m}$ for all i, j , the entries of $E = \text{diag}(e_j^* F)$ all have magnitude less than or equal to $\sqrt{\eta/m}$ as well, which means $\|E\| \leq \sqrt{\eta/m}$. We use this to show the Lipschitz constant of f is $L = \sqrt{\eta/m}$.

$$|f(x) - f(y)| = |||x^* E U\| - \|y^* E U\|| \leq \|(x - y)^* E U\| \leq \|E\| \|x - y\|$$

We bound the expected value of $f(X)$ as

$$\mathbb{E}f(X) \leq [\mathbb{E}f(X)^2]^{1/2} = [\mathbb{E}\|X^*EU\|^2]^{1/2} = \|EU\|_F \leq \|E\| \|U\|_F \leq \sqrt{\frac{\eta}{m}}r.$$

Now apply Proposition 3.9 with $t = \sqrt{8\log(m/\delta)}$ to find

$$\mathbb{P}\left\{\|e_j^*F DU\| \geq \sqrt{\frac{\eta}{m}}r + \sqrt{\frac{\eta}{m}8\log(m/\delta)}\right\} \leq \frac{\delta}{m}.$$

We need this to hold for each of the m rows. By the union bound, we obtain

$$\mathbb{P}\left\{\max_{1 \leq j \leq m} \|e_j^*F DU\| \geq \sqrt{\frac{\eta}{m}}(\sqrt{r} + \sqrt{8\log(m/\delta)})\right\} \leq m \frac{\delta}{m} = \delta.$$

Finally, note that $\mu(FDU) = \max_j \|e_j^*F DU\|^2$. □

We need to combine this result with Lemma 4.3 of [37] which is a corollary to Lemma 3.4 of [251].

Lemma 3.10 (Boutsidis and Gittens, Lem. 4.3 [37]). *Let $U \in \mathbb{F}^{m \times r_1}$ have orthonormal columns and let $\mu(U)$ denote its coherence. Let $0 < \tilde{\epsilon} < 1$ and $0 < \delta < 1$. Let r_2 be an integer such that*

$$6\tilde{\epsilon}^{-2}m\mu(U)\log(r_1/\delta) \leq r_2 \leq m.$$

Let $S \in \mathbb{R}^{r_2 \times m}$ sample r_2 rows uniformly at random and without replacement, and define $\mathcal{S} = \sqrt{\frac{m}{r_2}}S$. Then, with probability at least $1 - 2\delta$

$$\sqrt{1 - \tilde{\epsilon}} \leq \sigma_{\min}(\mathcal{S}U) \leq \sigma_{\max}(\mathcal{S}U) \leq \sqrt{1 + \tilde{\epsilon}}.$$

Lemma 3.7 now follows by combining Lemma 3.8 and Lemma 3.10.

3.4.1 A numerical experiment

We explained how *all* singular values of a tall and skinny matrix $AS_1 \in \mathbb{F}^{m \times r_1}$ can be estimated by the singular values of S_2AS_1 , where $S_2 \in \mathbb{F}^{r_2 \times m}$ is an embedding matrix and $r_2 \geq r_1$. In the next section we conclude how this, combined with the results of the previous section, leads to a rank estimation algorithm. In the following numerical experiment, we first illustrate the error one can expect in practice in the step from AS_1 to S_2AS_1 .

We let B be a dense matrix with fast polynomially decaying singular values, $\sigma_i(B) = i^{-3}$, of size $10^5 \times 2000$. We estimate $\sigma_i(B)$ with $\sigma_i(YB)$, where Y is either a subsampled randomized DCT embedding or a Gaussian embedding of dimension 4000. The relative error $|\sigma_i(YB) - \sigma_i(B)|/\sigma_i(B)$

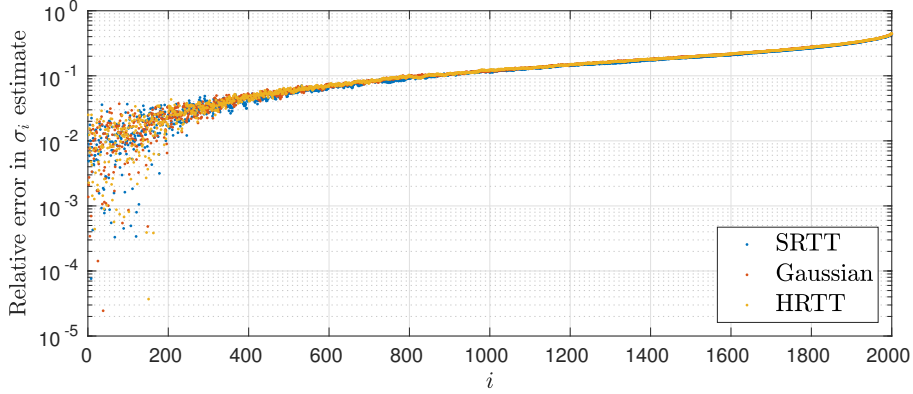


Figure 3.5: Relative error in the singular value estimates of a real matrix B with fast polynomially decaying singular values of size $10^5 \times 2000$ based on the singular values of $S_2 B$, where $S_2 \in \mathbb{R}^{4000 \times 10^5}$ is an embedding matrix (either a subsampled randomized DCT (SRTT), a Gaussian embedding matrix or a hashed randomized DCT (HRTT)).

is plotted in Figure 3.5. We see the relative errors are $o(1)$ for each $i = 1, \dots, 2000$. We use a dense matrix with incoherent left singular vectors to resemble the matrix AS_1 , which will never be very coherent due to the prior application of embedding S_1 . As a result, the SRTT embedding is sufficient.

3.5 A numerical rank estimation scheme

The numerical rank estimation scheme that follows from the observations

1. $\sigma_i(AS_1) \approx \sigma_i(A)$ for an approximately low-rank matrix and leading singular values, and
2. $\sigma_i(S_2 AS_1) \approx \sigma_i(AS_1)$ for all singular values of AS_1 .

We first present the theorem that ties both of these facts together. The remainder of the section focusses on a practical algorithm resulting from the theorem.

We combine Theorem 3.5 with the general embedding property presented in Definition 2.1.

Theorem 3.11. *Let $A \in \mathbb{F}^{m \times n}$ ($m \geq n$) have leading right singular vectors $V_1 \in \mathbb{F}^{n \times r}$. Let $S_1 \in \mathbb{F}^{n \times r_1}$ and $S_2 \in \mathbb{F}^{r_2 \times m}$ be two subspace embedding matrices with $r \leq r_1 \leq r_2$. Let $\tilde{U} \tilde{\Sigma} \tilde{V}^T = AS_1$ be an SVD of AS_1 . Assume*

1. S_1^T is a subspace embedding for V_1 with distortion ε_1 s.t. $1 - \varepsilon_1 \leq \sigma_i(V_1^T S_1) \leq 1 + \varepsilon_1$.
2. S_2 is a subspace embedding for $\tilde{U}$ with distortion ε_2 s.t. $1 - \varepsilon_2 \leq \sigma_i(S_2 U) \leq 1 + \varepsilon_2$.

Then, for $i = 1, \dots, r$,

$$(1 - \varepsilon_1)(1 - \varepsilon_2) \leq \frac{\sigma_i(S_2 AS_1)}{\sigma_i(A)} \leq \sqrt{(1 - \varepsilon_1)^2 + \left(\frac{\sigma_{r+1}(A) \|S_1\|}{\sigma_i(A)} \right)^2} (1 + \varepsilon_2). \quad (3.19)$$

Proof. Note first that $\sigma_i(S_2AS_1) = \sigma_i(S_2\tilde{U}\tilde{\Sigma}\tilde{V}^T)$ and thus that

$$\sigma_{\min}(S_2\tilde{U})\sigma_i(AS_1) \leq \sigma_i(S_2AS_1) \leq \sigma_{\max}(S_2\tilde{U})\sigma_i(AS_1).$$

The result then follows immediately from Theorem 3.5 and the subspace embedding property of S_2 . $\square$

3.5.1 Algorithm and parameter choices

The rank estimation algorithm we propose based on Theorem 3.11 is displayed in pseudocode in Algorithm 3. In the first stage we sketch from the right, in the second stage from the left, and in the final stage we compute the singular values of the small sketch.

Algorithm 3 Randomized rank estimation. Given an $m \times n$ matrix A , a tolerance ϵ (optional) and an estimate for the upper bound for rank r_1 , this scheme computes an estimate for the ϵ -rank of A .

1. Sketching the leading singular values:

- 1: Set $\tilde{r}_1 = \text{round}(1.1r_1)$.
- 2: Draw $n \times \tilde{r}_1$ random embedding S_1 .
- 3: Form AS_1 .

2. Sketching to embed all singular values:

- 4: Set $r_2 = 2\tilde{r}_1$.
- 5: Draw $r_2 \times m$ SRTT embedding Θ .
- 6: Form ΘAS_1 .

3. Singular value estimates:

- 7: Compute the first r_1 singular values of ΘAS_1 .
 - 8: Let the numerical rank estimate be the first $\hat{r}$ s.t. $\sigma_{\hat{r}+1}(\Theta AS_1) \leq \epsilon\|A\|$. The norm of A can be estimated by $\sigma_{\max}(S_2AS_1)$ or $\|AS_1\|_2$.
 - 9: If no such $\hat{r}$ exists, repeat the algorithm with a larger r_1 , e.g. $r_1 := 2r_1$, by appending to the sketches as described in Section 3.5.1.
-

There are a few parameter choices that must be considered. Primarily, the algorithm requires an upper bound for the numerical rank as an input (r_1). This input informs the size of first embedding S_2 , and is the total number of singular values estimates that will be obtained. If the input rank r_1 was not large enough, i.e. $r_1 < \text{rank}_\epsilon(A)$, Algorithm 3 is naturally only able to detect that r_1 is a lower bound for $\text{rank}_\epsilon(A)$. To get a proper estimate, we then need to rerun the algorithm with a larger input rank, taking advantage of the preceding computation by appending extra columns to S_1 and rows to Θ .

The choice of r_1 can be based on a number of possible considerations. For example, one might choose r_1 based on memory requirement, e.g. when one is unwilling to store more than $(m+n)r_1$ numbers. If some information is available on A (e.g. it is updated from a matrix with known singular values), that can provide a good choice of r_1 . If sketching is not a computationally expensive part of the routine (e.g. because of sparsity in A), then a generous overestimate of the rank can provide a lot of information at limited cost.

If no information is available one might choose a small value of $r_1 = O(1)$ initially, and then increase the rank until a desired tolerance is reached. Naturally, this approach results in some inefficiency. If the sketches employed are Gaussian, the complexity is $\mathcal{O}(mnr_1)$ (dominated by sketching) and the overhead of underestimating r_1 is minimal. If the update rule is $r_1 := 2r_1$, then the inefficiency is bounded by a factor 2. If an SRTT or HRTT is used, then increasing r_1 without resketching the whole matrix requires some care. For an SRTT $\mathcal{S}FD$ or a HRTT $\mathcal{H}FD$, where $\mathcal{S}$ ($\mathcal{H}$) is a subsampling (hashing) matrix, F is a trigonometric transform, and D is a diagonal matrix of random signs, we can choose to temporarily store ADF so we can subsample increasingly from this matrix. The same is true for the second round of sketching, and one might decide to store $F_2D_2AD_1F_1$ for independent random sign matrices D_1 and D_2 and trigonometric transform F_1 and F_2 of conforming size.

When an appropriate upper bound r_1 has been identified, the actual embedding dimension for S_1 will be chosen as $\tilde{r}_1 = \text{round}(1.1r_1)$. This is because we have identified that the smallest singular values of AS_1 are not reliable estimators of the corresponding singular values of A ; this was discussed in Section 3.3.2.1. By oversampling with 10% and disregarding the smallest singular values corresponding to oversampling, we diminish tail effects.

In the second stage, we set the standard embedding dimension to be $r_2 = 2\tilde{r}_1$. This is a small sketch compared to theoretical guarantees, but we argue it is still reasonable. The bulk of the accuracy is lost in the first sketch: in this step we preserve the order of magnitude of the singular values. Since we are applying an embedding in the second stage, the theoretical guarantees on singular values are much stronger. Although we do not need it, we preserve the singular values of AS_1 up to a small multiple. For this reason, we focus on reducing computational complexity and take a small sketch dimension.

Provided the algorithm runs successfully, the computational complexity is $\mathcal{O}(mn \log n + r_1^3)$ for a dense matrix $A \in \mathbb{F}^{m \times n}$, where the two terms come from steps 3 and 7, assuming both S_1 and S_2 are SRTT matrices. The cost of step 3 can be lower if A has structure, in which case the $\mathcal{O}(mr_1 \log m)$ cost in step 6 may become significant.

Regarding step 7, computing the singular values of S_2AS_1 , one can of course compute them via the SVD. We do this in most cases as the $O(r_1^3)$ cost is usually negligible. Alternatively, since it suffices to retrieve estimates for $\sigma_i(S_2AS_1)$, one can compute the QR factorization $S_2AS_1 = QR$

and take the diagonal elements of R as approximations of the singular values; this works as a QR factorization of a matrix whose (right) singular vectors are randomized are rank-revealing with high probability [22]. As a result, the numerical rank of S_2AS_1 is approximately preserved in the rank of R , and it can be used to estimate the rank of A .

Finally, we note that this algorithm has merit even when a tolerance ϵ is not provided. We have shown that the singular values of S_2AS_1 are good order of magnitude estimates for the singular values of A . This provides much more knowledge than just numerical ranks. Most straightforwardly, one can visually investigate the spectrum to identify gaps and make algorithmic choices (such as determining an appropriate ϵ) based on the spectrum. The ability to identify the rate at which singular values decay is also useful in, e.g. inverse problems, where the level of ill-posedness informs appropriate computational routines. In the general context of low-rank approximation, the spectrum does not just inform a suitable rank, but can also inform what algorithm to use (e.g. randomized SVD or generalized Nyström) or whether to apply a power iteration. We discuss the specific combination of rank estimation with the randomized SVD algorithm in the next section.

3.5.2 Rank estimation as part of randomized low-rank approximation

It is worth discussing the case where Algorithm 3 is used as an initial step for randomized low-rank approximation, such as RSVD [127] or generalized Nyström [196]. In such cases, computing the sketch AS_1 is needed anyway, so step 3 incurs no additional cost.

Moreover, with generalized Nyström, even S_2AS_1 is required, as the low-rank approximant is of the form $AS_1(S_2AS_1)^\dagger S_2A$. Furthermore, to evaluate $(S_2AS_1)^\dagger$ a QR factorization $S_2AS_1 = QR$ is computed. It follows that the additional work needed to execute generalized Nyström is just to estimate the singular values of R . As described above, to do so one can simply take the diagonal elements of R , requiring just $O(r_1)$ work. Thus in the context of generalized Nyström, a rank estimate can be obtained essentially for free.

In the next section we explain how this rank estimate can be employed to speed up fixed-precision low-rank approximation. In particular, we note that using the singular value estimates we can reduce the size of a sketch as appropriate before performing the dominant computational work.

3.5.2.1 Fixed-precision low-rank approximation problem

Rank estimation as part of low-rank approximation is inherently linked to the fixed-precision problem³, where one aims to compute a low-rank factorization $\hat{A}$ of unspecified rank such that $\|A - \hat{A}\| \leq \epsilon\|A\|$ for some user-specified tolerance ϵ . We propose a scheme to combine rank estimation and

³We note that the terminology ‘fixed-precision’ here is unrelated to mixed-precision algorithms. We use fixed precision to denote a fixed tolerance (see also Section 1.1.1).

low-rank approximation through existing error estimates. The algorithm will be used here for approximation in the Frobenius norm but can easily be extended to approximation in any other norm for which randomized low-rank approximation error estimates are available. The aim will be to find an approximation of the form $\hat{A} = QB$, where $Q \in \mathbb{F}^{m \times r}$ has orthonormal columns and $r \leq n$.

In their landmark paper of 2011, Halko, Martinsson and Tropp [127] describe and extensively analyze the randomized rangefinder (or randomized SVD) algorithm to compute such an approximation. This algorithm is displayed in pseudocode in Algorithm 1 in Section 2.2. A limitation of this factorization routine, as well as of other randomized low-rank approximation algorithms, is that we often wish to solve a fixed precision problem instead of a fixed rank algorithm, where the appropriate choice of the input rank r is usually unknown.

As mentioned, we propose to use rank estimation to tackle this problem using a priori error estimates. Under the conditions of Algorithm 1 with a Gaussian embedding matrix, it is known that [127, Thm. 10.5] (see also Lemma 2.3)

$$\mathbb{E}\|A - QB\|_F \leq \left(1 + \frac{r}{p-1}\right)^{1/2} \left(\sum_{j>r} \sigma_j^2\right)^{1/2}. \quad (3.20)$$

We can use the Rank Estimation Algorithm 3 to compute estimates of the trailing singular values which appear in the estimate (3.20). Suppose we have estimates $\hat{\sigma}_i = \sigma_i(S_2AS_1) \approx \sigma_i(A)$ for the first r_1 singular values of A , as a result of step 6 of Algorithm 3. Conservatively, and taking into account that there could be underestimation in the last few estimates if the tail is light, we set $\hat{\sigma}_i = \hat{\sigma}_{r_1}$ for $i = r_1 + 1, \dots, n$. We can then choose the target rank r in the rangefinder algorithm to be the first r such that

$$\left(1 + \frac{r}{p-1}\right)^{1/2} \left(\sum_{j>r} \hat{\sigma}_j^2\right)^{1/2} \leq \epsilon \|A\|,$$

where p is fixed to be 5 or 10 as suggested in [127] or where p is fixed relative to r , say $p = 0.1r$, as suggested in [196].

As mentioned before, the rank estimation part of this scheme requires only $\mathcal{O}(mr_1 \log m + r_1^3)$ additional cost. More importantly, the singular value estimates it provides allow us to reduce the size of the sketch AS_1 ahead of the rangefinder process. This could result in considerable speed-up, as the QR factorization of AS_1 and the matrix-matrix multiplication Q^*A are usually the dominant costs in the algorithm. Finally, we also know the rank of QB is nearly minimal while (approximately) satisfying the accuracy criteria; by contrast, reducing the rank of a QB factorization without rank estimation requires computing the SVD of B and truncating it.

Algorithm 4 Rank Estimation combined with Randomized Rangefinder. Given an $m \times n$ matrix A , a tolerance ϵ , an oversampling parameter $p \geq 2$ (either fixed absolutely or fixed relatively to r) and an upper bound for the numerical rank r_1 , this scheme computes a QB factorization of A such that $\|A - QB\|_F \lesssim \epsilon \|A\|$.

1. Rank Estimation:

- 1: Set $\tilde{r}_1 = \text{round}(1.1r_1)$, draw $n \times \tilde{r}_1$ random embedding S_1 , and form AS_1
- 2: Set $r_2 = 2\tilde{r}_1$, draw $r_2 \times m$ SRTT Θ , and form ΘAS_1
- 3: Compute the first r_1 singular values of ΘAS_1 .
- 4: Set $\hat{\sigma}_i = \sigma_i(\Theta AS_1)$ for $i = 1, \dots, r_1$ and $\hat{\sigma}_i = \sigma_{r_1}(\Theta AS_1)$ for $i = r_1 + 1, \dots, n$

2. Randomized Rangefinder:

- 5: Set r to be the smallest integer s.t.

$$\left(1 + \frac{r}{p-1}\right)^{1/2} \left(\sum_{j>r} \hat{\sigma}_j^2\right)^{1/2} \leq \epsilon \|A\| \quad (3.21)$$

If no such r exists, repeat the algorithm with a larger r_1 , e.g. $r_1 := 2r_1$, by appending to the sketches. If (an estimate of) $\|A\|$ is unavailable, use $\sigma_1(\Theta AS_1)$ to approximate $\|A\|$

- 6: $AS_1 \leftarrow AS_1(:, 1 : (r + p))$
 - 7: $[Q, \sim] = \text{qr}(AS_1, 0)$ (thin QR factorization)
 - 8: $B = Q^* A$
-

3.6 Numerical experiments

We introduce synthetic example matrices to experimentally support the theoretical results. Each of the matrices is diagonal and square of dimension $n = 10^5$. They are, however, treated as dense matrices by the algorithms here. The singular values are on the diagonal in decreasing order. We distinguish the following singular value spectra and resulting matrices, inspired by [253]:

1. Gaps in spectrum: we define a matrix A_G with $\sigma_i = 1$ for $i = 1, \dots, 100$, $\sigma_i = 10^{-4}$ for $i = 101, \dots, 200$, $\sigma_i = 10^{-8}$ for $i = 201, \dots, 300$, $\sigma_i = 10^{-12}$ for $i = 301, \dots, 400$ and $\sigma_i = 10^{-16}$ for $i > 400$.
2. Polynomial decay: each singular value takes the value $\sigma_i = i^{-p}$ for a parameter p . We define a matrix A_{SP} with a slow polynomial decaying spectrum (SP) with $p = 1$, and a matrix A_{FP} with fast polynomial decay (FP) with $p = 3$.
3. Exponential decay: the singular values are logarithmically equally spaced between 1 and σ_n , that is $\sigma_i = 10^{-q(i-1)}$ for a parameter q . We study a matrix A_{SE} with slow exponential decay (SE) for which $q = 0.01$ and a matrix A_{FE} with fast exponential decay (FE) for which $q = 0.5$.

Each of the matrices has spectral norm $\|A\| = 1$ so the *relative* accuracy notation will be suppressed in this section. Considering the discussion in Section 3.3.2.1, we restrict our experiments to matrices with very rapidly decaying singular values.

In each of the experiments in subsections 3.6.2-3.6.4, we will choose the embedding matrix S_1 to be a hashed randomized DCT matrix with a 1-hashing matrix and $S_2 = \Theta$ to be a subsampled

randomized DCT matrix. We display the performance of Algorithms 3 and 4 in various numerical experiments.

3.6.1 Detecting gaps in the spectrum

Here we demonstrate that Algorithm 3 works extremely well for approximately low-rank matrices that have gaps in their singular value spectrum. The algorithm can in fact be used in two ways: 1) if the specified tolerance is within the gap between two singular values, the algorithm will perform very well in identifying the correct rank even for a small number of samples, and 2) by considering the gaps in the singular value spectrum of S_2AS_1 , one can identify gaps in the singular value spectrum of A and use this to inform the ϵ value and target rank.

To illustrate this we compare the singular values $\sigma_i(S_2AS_1)$ to $\sigma_i(A)$ for various values of r_1 (see Algorithm 3). Note that we oversample by 10% and disregard the singular values associated with oversampling. The results are shown in Figure 3.6. The figure displays the remarkable effectiveness of the algorithm, where gaps in the spectrum are clearly identified even for small values of r_1 .

We use Gaussian embeddings for the column space and SRCTs for the row space, to allow for an adaptive procedure. That is, we start with a Gaussian S_1 of dimension 110, compute S_2AS_1 and its singular values, and then increase the sketch dimension of S_1 by a 100 until we can identify the gap. This is how Algorithm 3 could be used if there is no upper bound on r_1 available. We compare against the performance of the DOS algorithms [254] (Chebyshev-based) and [255] (Lanczos-based). Our rank estimation algorithm took 18.33 seconds to reveal the gap at the 400th singular value. The runtime of the DOS algorithms was 2223 sec. for [254] and 199.1 sec. for [255]. The DOS plots are also shown in Figure 3.6.

It is inherently difficult to detect gaps visible on a log-scale using a DOS algorithm. The algorithms are based on estimating a sum of Dirac delta functions using a sum of smooth functions such as Gaussians. To be able to distinguish between very small eigenvalues, the width of the Gaussian has to be as small as the smallest eigenvalue. That is, it would be necessary to have a very high-resolution spectral density. The resolution refers to the size of the interval in which we estimate the number of eigenvalues [169]. However, a fine resolution implies the number of sample points needed to detect larger eigenvalues is large. As a result, the computational effort necessary is not tractable. Figure 3.6 shows that even if the DOS is estimated perfectly for the chosen Gaussian standard deviation, the gaps are not visible.

3.6.2 Robustness of the rank estimator

The other four example matrices discussed (SP, FP, SE, and FE) do not have significant gaps in their singular value spectra, which makes the numerical rank less well-defined. Rather than focusing

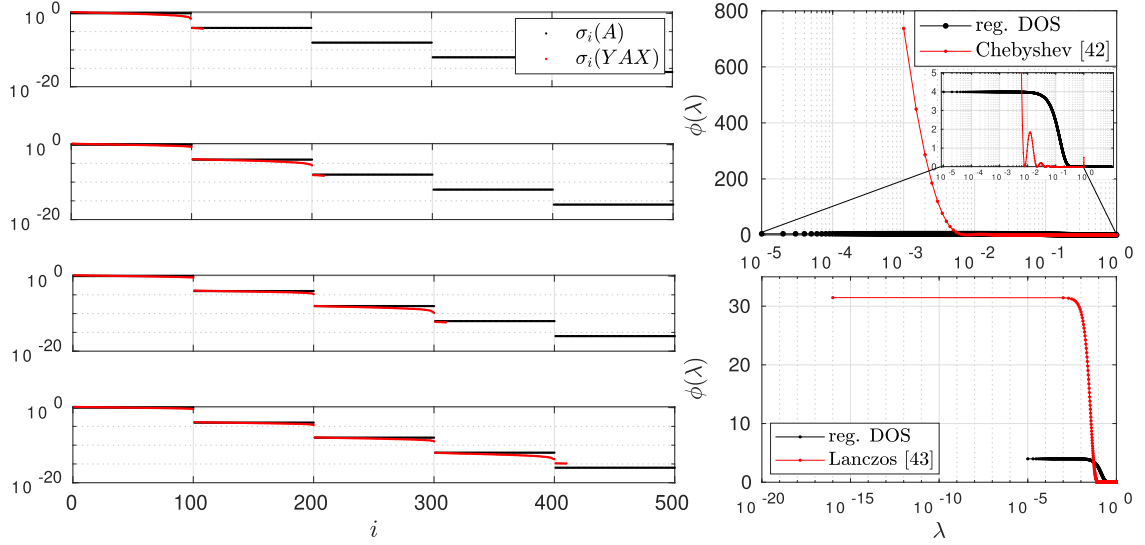


Figure 3.6: On the left, the singular values of A_G and $S_2 A_G S_1$, where S_1 is a Gaussian embedding and S_2 is an SRTT matrix with a discrete cosine transform (SRCT). The plots from top to bottom correspond to $r_1 = 110, 210, 310, 410$ respectively. The gaps that exist in the spectrum of A are also visible in the spectrum of $S_2 A_G S_1$, even for values of r_1 close to the location of the gaps. On the right, the density of state plots resulting from the Chebyshev-based [254] and Lanczos-based algorithms [255]. The algorithms are unable to identify the gaps for very small singular values. Even the exact regularized DOS can only identify the bulk of eigenvalues is of order 10^{-2} or smaller. See the references for further details on interpretation of these graphs.

on the numerical rank estimate, $\hat{r}$, it thus makes sense to look at $\sigma_{\hat{r}+1}(A)$ and its proximity to ϵ . Figure 3.7 shows the results of Algorithm 3 applied 100 times to each of the example matrices. We see that, although the rank estimate $\hat{r}$ we recover is often an underestimate when there are no significant gaps between singular values near the desired accuracy ϵ , $\sigma_{\hat{r}+1}(A)$ is always very close to ϵ . One could argue this is not necessarily a shortcoming of the algorithm, but rather a display of the difficulties associated to the concept of the exact numerical rank $\text{rank}_\epsilon(A)$ when $\sigma_r(A) \approx \sigma_{r+1}(A)$. Importantly, the algorithm always found a rank that satisfies the two desiderata set out in Section 3.1.1. In particular, when there are significant gaps in the singular value spectrum we recover the exact numerical rank in all except one instance; see the rightmost column of the figure (and the forthcoming Figure 3.6).

3.6.3 Comparison with other numerical rank estimation algorithms

Suppose that the goal is to obtain the numerical rank of a given matrix for some tolerance ϵ . We compare five different methods to achieve this for a $10^5 \times 10^5$ matrix with slow exponential decaying spectrum (SE) and various values of ϵ ranging from 10^{-1} to 10^{-12} . The results are shown in Figure 3.8.

The algorithms we compare are rank estimation (RE) with different parameters r_1 , the rank estimate obtained from RSVD with different parameters for r , the rank estimate obtained from

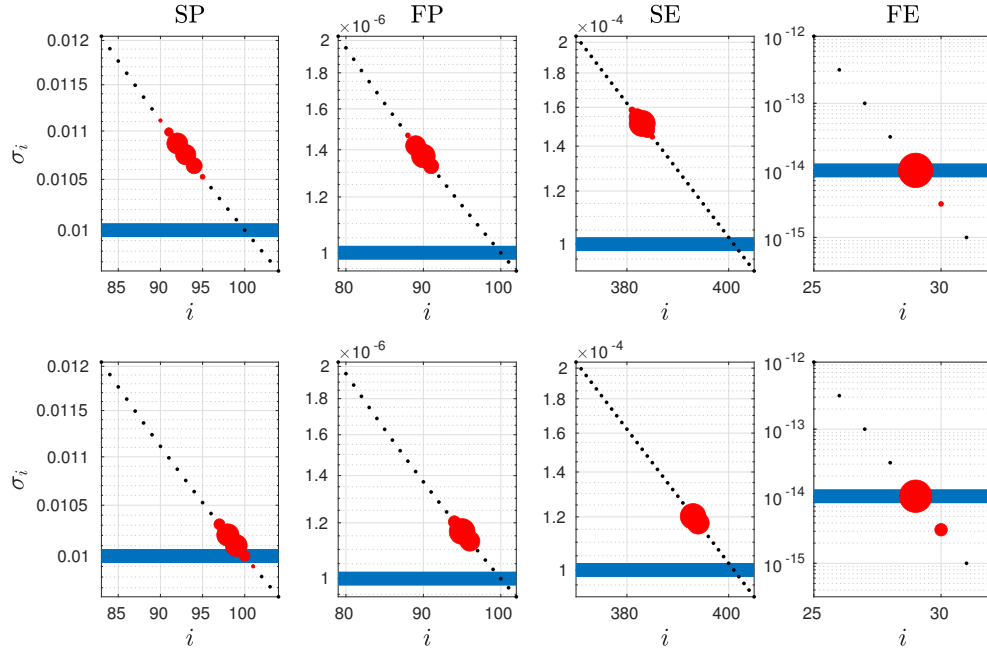


Figure 3.7: The results of Algorithm 1 applied 100 times to four dense synthetic test matrices with singular value spectra that decay Slow Polynomially (SP), Fast Polynomially (FP), Slow Exponentially (SE), and Fast Exponentially (FE), as defined at the start of Section 3.6. Each matrix is square diagonal with $n = 10^5$. The blue line indicates the tolerance ϵ and the black dots represent the singular values $\sigma_i(A)$. The red dots indicate the singular values $\sigma_{\hat{r}+1}(A)$, where $\hat{r}$ is the numerical rank estimate found by the algorithm. The area of the dot represents the frequency with which it a specific $\hat{r}$ was found. The algorithm would work perfectly if the singular value on the blue line were found in each iteration, as is (almost) the case for the FE matrix. The upper and lower row correspond to upper limits $r_1 = 2\text{rank}_\epsilon(A)$ and $r_1 = 4\text{rank}_\epsilon(A)$ respectively.

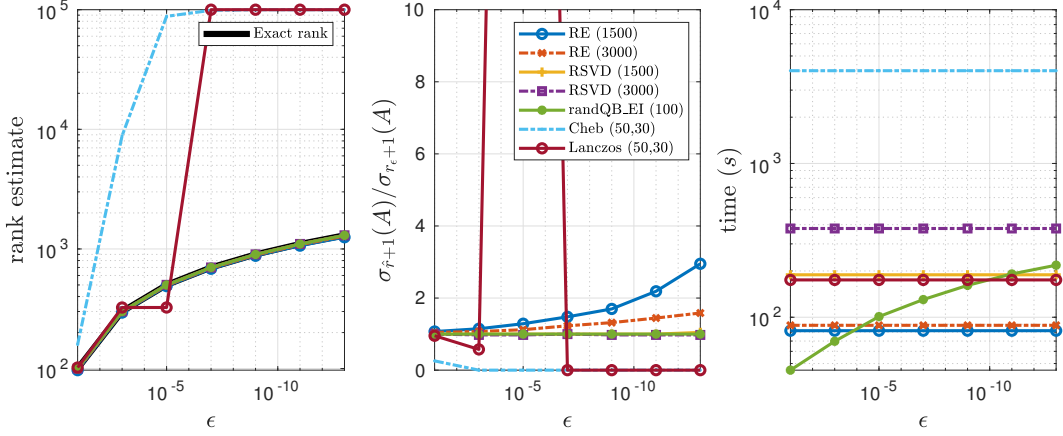


Figure 3.8: We compare four algorithms to compute the numerical rank of a real square $10^5 \times 10^5$ matrix with slow exponentially decaying singular values (SE). The algorithms we compare are rank estimation (RE) with different parameters $r_1 = 1500, 3000$, the rank estimate obtained from RSVD with different parameters for $r = 1500, 3000$, the rank estimate obtained from randQB_EI with parameter $b = 100$, the DOS algorithm with Chebyshev filters introduced in [254] (Cheb) and the Lanczos DOS algorithm introduced in [255] (Lanczos), both with degree 50 and $nvecs = 30$.

randQB_EI [284] with parameter $b = 100$, the DOS algorithm with Chebyshev filters introduced in [254] (Cheb) and the Lanczos DOS algorithm introduced in [255] (Lanczos), both with degree 50 and 30 samples, which are the default parameters in the authors' implementation⁴. We see that the latter two DOS algorithms do not accurately compute the rank for smaller values of ϵ . A higher degree polynomial approximation would be necessary for this, which will slow down the computation. The DOS algorithms work very well for smaller matrices, but in the case of small tolerances and extremely large data, where each matrix-vector operation is costly in communication, the algorithms are outperformed by RE, RSVD, and randQB_EI.⁵

As for these three, we see that RE is considerably faster in the context of rank estimation. Note also that the difference in running time will be more pronounced when the matrix is too large to fit into fast memory, as the other algorithms need to pass over the data more than once. Although the numerical rank is slightly underestimated for each value of ϵ with the Rank Estimation algorithm, and is exact for the other algorithms, we see that $\sigma_{\hat{r}+1}$ is always extremely close to $\sigma_{\text{rank}_\epsilon(A)+1} \approx \epsilon$. This requirement for a good rank estimation set out in the introduction is thus satisfied.

3.6.4 The fixed-precision problem

Often when dealing with large-dimensional approximately low-rank matrices, our aim is to obtain a low-rank approximation that is accurate within a given tolerance, as discussed in Section 3.5.2. We compare three different algorithms to obtain such an approximation, Rank Estimation combined

⁴The implementation used is available at <https://shashankaubaru.github.io/codes.html>.

⁵We used the implementation of randQB_EI by the authors of [284], available at <https://github.com/WenjianYu/randQB-auto>.

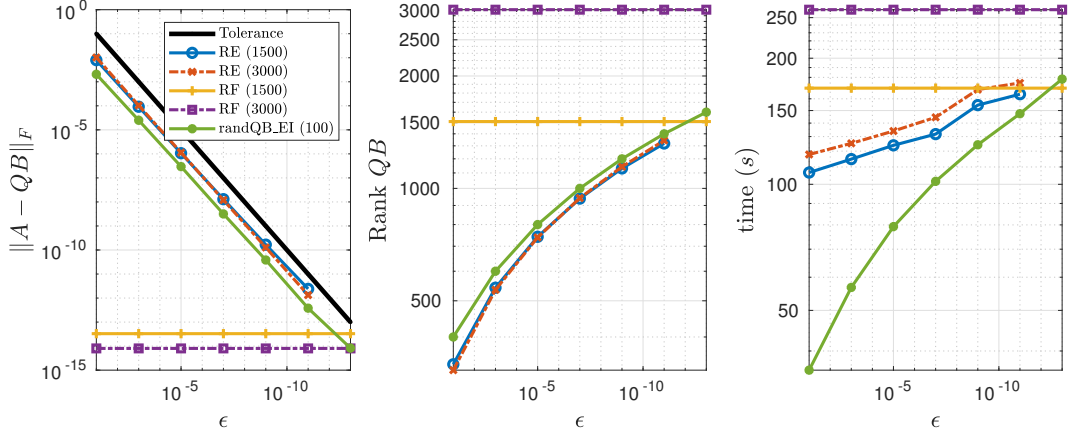


Figure 3.9: We compare four algorithms to compute a low-rank QB approximation to solve the fixed precision problem $\|A_{SE} - QB\| \leq \epsilon$, where A_{SE} is a $10^5 \times 10^5$ square matrix with slow exponential decaying singular values. The algorithms include rank estimation with rangefinder (RE), rangefinder (RF) and RandQB_EI. We compare for the parameter choices $r_1 = 1500, 3000$, $r = 1500, 3000$ and $b = 100$ for each of the algorithms respectively.

with Randomized Rangefinder (Algorithm 4), Randomized Rangefinder (Algorithm 1), and the adaptive algorithm RandQB_EI [284]). The results are shown in Figure 3.9. The figure shows that the adaptive algorithm RandQB_EI is the fastest in most cases.

It is difficult to compare these algorithms as the performance is very dependent on the context of the problem, and the parameters chosen. We argue that our algorithm is the best choice when it is very expensive to communicate with the matrix, and a reliable prior rank estimate is unavailable. In this case, overestimating the target rank in randomized rangefinder and randomized SVD is much more expensive than it is in Algorithm 4, rank estimation with rangefinder. This is portrayed in Figure 3.10.

3.7 Summary and outlook

We have presented theory and algorithms related to the rank estimation of approximately low-rank matrices. The main theoretical contribution is the analysis of the singular value spectrum of the sketch AS , where A is a large matrix in both dimensions and S is a tall random embedding matrix. We showed that, provided A is approximately low-rank, the order of magnitude of its singular values is approximately preserved by the sketching procedure. A second sketch from the left, to obtain a very small matrix $S_2^T AS_1$, further preserves the singular values. This naturally leads to an algorithm to approximate singular values coarsely, which as a result provides a numerical rank estimate. We showed the algorithm effectively detects gaps in the spectrum, provides sufficiently good rank estimates, and has a small failure probability. We note that A need to be sufficiently low-rank for the spectrum of AS to provide order-of-magnitude estimates. In particular, the tail

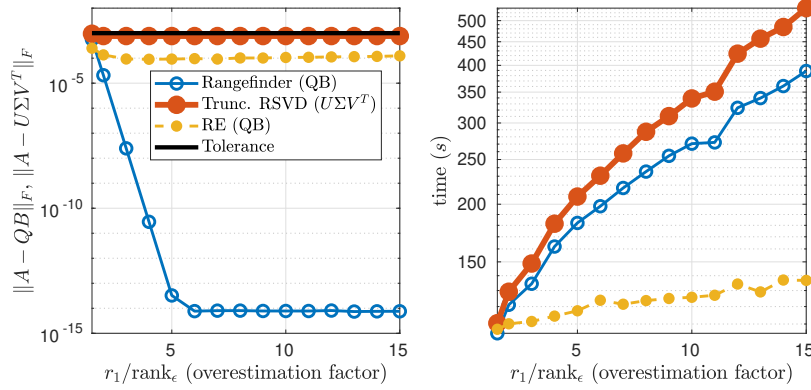


Figure 3.10: We compare three algorithms for fixed precision estimation that require only two views of the matrix, randomized rangefinder (Algorithm 1), randomized SVD with truncation and rank estimation combined with randomized rangefinder (Algorithm 4, shown as RE (QB)). For a fixed tolerance $\epsilon = 10^{-3}$, we vary the input target rank/upper bound for the rank. Although each of the algorithms satisfy the condition of an error smaller than ϵ , we see that Algorithm 4 is much faster, especially for greater overestimates. The matrix is A_{SE} of dimension $n = 10^5$.

singular values of A should not be heavy compared to the dominant singular values.

We also show that the process can improve the performance of algorithms for fixed-precision low-rank approximation. Although in its standard form still not entirely parameter-free, we have shown that combining rank estimation with randomized low-rank approximation algorithms scales much better in the user-specified upper bound on the rank than existing approaches (see Figure 3.10). We have achieved this by leveraging information from the spectrum of the (cheaply obtained) sketch AS to update the dimensions of AS accordingly, before performing expensive steps of the low-rank approximation algorithm (orthogonalization and matrix-multiplication). We believe there is a wide range of both randomized and deterministic algorithms that can benefit from such a cheaply available estimate, especially if there is already sketching involved. This cheap estimate can provide guidance for more general algorithmic choices, such as the selection of a suitable low-rank approximation algorithm and parameter choices within (e.g. oversampling, power iterations).

From a theoretical point of view, we believe this work also highlights the possible connections between random matrix theory (RMT) and high-dimensional statistics, on the one hand, and on the other hand randomized numerical linear algebra. The sketch AS can be seen as random samples drawn from an m -dimensional distribution determined by the singular values and vectors of A . The analysis of the spectrum of AS is thus equivalent to covariance matrix estimation in high-dimensional statistics. This provides avenues to improve the accuracy of $\sigma_i(AS)$ as approximations to $\sigma_i(A)$ through statistical procedures as shrinkage [161–163].

Furthermore, much of the theory in randomized NLA comes from a decoupling of the linear algebra and the statistics [174]. As a result there may be theory developed in our community, that provides new insights in the statistics community. Additionally, statistics theory on, for instance,

PCA is deeply related to low-rank approximation through sketching. In general, we believe there is a wealth of theory in RMT and high-dimensional statistics left unused by numerical analysts.

Chapter 4

Iterative solvers for least squares problems and the numerical stability of sketch-and-precondition

4.1 Introduction

We consider the linear least squares (LS) problem:

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2, \quad A \in \mathbb{R}^{m \times n}, \quad b \in \mathbb{R}^m, \quad (4.1)$$

where $m > n$. The LS problem arises in a variety of contexts, including inverse problems [99, 130, 243], approximation of functions [67, 123, 249], and regression analysis [137, 250].

The most well-known example of a least squares problem is perhaps linear regression. In its simplest form, we have a response variable y and one or more explanatory variables $x_1, \dots, x_n$. If we assume a linear relation between these variables,

$$y = \beta_0 + \beta_1 x_1 + \dots + \beta_n x_n + \varepsilon,$$

where ε represents some error in the model, we can estimate the regression coefficients $\beta_0, \dots, \beta_n$ from a dataset $\{x_1^{(i)}, \dots, x_n^{(i)}, y^{(i)}\}_{i=1}^m$ of m samples. The vector of coefficients that best explain the variation present in the dataset is the solution to the LS problem:

$$\beta^* = \min_{\beta \in \mathbb{R}^n} \|X\beta - Y\|,$$

where $X_{ij} = x_j^{(i)}$ and $Y_i = y^{(i)}$ are a matrix and vector of respective size $m \times n$ and $n \times 1$. Linear problems like this appear in diverse fields, including economics, finance, medicine, biology, chemistry, physics, sociology, and numerous other areas [53].

Another important class of problems leading to LS problems of the form (4.1) are linear inverse problems. In an inverse problem, we have some measurements of an observable, and we wish to reconstruct the values of some variables on which the observable depends, given some relation between them; i.e. we want to invert the relationship. An example is the problem of gravity surveying in geophysics (following [130]), in which we wish to reconstruct an unknown mass distribution $f(t)$ over some line $0 \leq t \leq 1$ at a particular depth from measurements of the vertical component of the gravity field at the surface $g(t)$. These quantities are related through an integral equation,

$$\int_0^1 K(s, t) f(t) dt = g(s),$$

where the kernel $K(s, t)$ is a function relating f and g (see [130, Sec. 2.1] for details). In a real problem, the measurements $g(t_i)$, $i = 1, \dots, m$, are discrete, and we actually solve a LS problem $\min \|Kf - g\|$, where K is a discretization of the integral transform, and f and g are discretizations of $f(t)$ and $g(s)$ respectively. Inverse problems are discussed in more detail in Chapter 6.

The LS problem is ubiquitous in data science and scientific computing [174], and has received significant attention from the randomized NLA community in the last two decades [14, 47, 61, 90, 93, 105, 146, 148, 149, 158, 187, 222, 275]. One of the big successes in the field is the development of the sketch-and-precondition framework introduced in Section 2.3.2. The idea in this framework is the use a sketch $SA \in \mathbb{R}^{s \times n}$, $n \leq s \ll m$ to find a preconditioner for (4.1) so that it can then be solved with an iterative LS solver. This framework is also the foundation of the work in Chapters 5 and 6 of this thesis. This chapter serves primarily to provide necessary background details and context to Chapters 5 and 6.

In this chapter, we will primarily discuss the case where A is full-rank and dense, and $m \gg n$. One of the reasons is that the perturbation bounds due to Wedin [272] we present here are only suitable for full rank matrices. Furthermore, stability results for direct methods for LS problems are usually dependent on an assumption that the condition number of A is not large compared to the reciprocal of machine precision. In Section 4.5.1, we briefly discuss existing approaches in the literature for adapting the sketch-and-precondition framework to rank-deficiency. In Chapters 5 and 6, we drop the the assumption of full-rankedness. In either chapter, we present a different approach to tackling rank-deficiency. Additionally, the underdetermined problem, i.e. $m < n$, is discussed briefly in Chapter 6.

We review linear algebra preliminaries relating to LS problems and introduce perturbation theory for LS in 4.2.3. This perturbation theory explains the effects of small perturbations on the quality

of approximate solutions. We accompany this discussion with an introduction to the concepts of backward error and backward stability in Section 4.2.4. We then discuss some classical direct methods for solving LS problems, demonstrate their behaviour in finite precision in Section 4.2.5, and try to explain this behaviour by combining perturbation theory with numerical stability theory. We follow classical textbooks [34, 137, 140, 145] throughout these sections and include illustrative numerical experiments.

We then turn to classical iterative algorithms for LS problems, specifically LSQR [205] and LSMR [103]. We derive the LSQR algorithm in full and describe how it can be efficiently implemented in finite precision in Section 4.3. These algorithms are crucial to the sketch-and-precondition framework. We briefly discuss the convergence behaviour and numerical stability of LSQR and LSMR in Section 4.4.

We derive the preconditioned version of LSQR¹ in Section 4.5. We are then fully set-up to investigate the finite precision behaviour of sketch-and-precondition. In Section 4.6, we show that preconditioned LSQR or LSMR with a randomized preconditioner (as in the sketch-and-precondition framework) performs poorly on various quality metrics, namely the residuals, backward error, and (forward) error of solutions. In particular, we show the sketch-and-precondition framework as presented in Algorithm 2 is not numerically stable. Although we are unable to pinpoint the exact source of these instabilities, we hope this chapter serves as necessary background information to investigate this further.

4.1.1 Accuracy metrics for approximate solutions to LS problems

Let x^* denote an exact minimizer to (4.1). In finite precision arithmetic, a computed solution $\hat{x}$ is clearly not exactly equal to x^* . Even with numerically stable algorithms, small rounding errors are incurred. There are a few metric which we might require a solution $\hat{x}$ to minimise, depending on the problem context.

1. Residual minimization: a computed solution should explain the observed data well:

$$\|A\hat{x} - b\| \approx \|Ax^* - b\|.$$

2. Small (forward) error: a computed solution should be close to the exact minimizer:

$$\min \|\hat{x} - x^*\|.$$

¹Preconditioned LSQR is implemented in MATLAB [144] without the explicit computation of the iterates in the preconditioned system, similar to preconditioned conjugate gradient implementations. However, we are unaware of any texts explicitly deriving the preconditioned LSQR algorithm as implemented in MATLAB. We note the SciPy implementation in Python has implemented preconditioned LSQR with explicit computation of preconditioned iterates.

3. Small error with respect to some ground truth: in a model where there exists a ground truth x_{true} and b_{true} such that $Ax_{\text{true}} = b_{\text{true}}$, but we observe $b = Ax_{\text{true}} + e$ where e is some noise, we want $\hat{x}$ to recover the ground truth:

$$\min \|\hat{x} - x_{\text{true}}\|.$$

4. Small backward error: the backward error associated to a solution $\hat{x}$ is the distance between the problem (4.1) we set out to solve, and the problem that is exactly solved by $\hat{x}$. This distance should be small:

$$\min \|\Delta A, \Delta b\|_F \quad \text{such that} \quad \hat{x} = \operatorname{argmin}_{x \in \mathbb{R}^n} \|(A + \Delta A)x - (b + \Delta b)\|.$$

In Sections 4.2.3, 4.2.4, and 4.2.5, we discuss perturbation theory of least squares problems together with empirical investigation to investigate the performance of classical algorithms on the metrics listed above.

The *backward* error is the quantity that can most naturally be controlled through the specific way in which an algorithm is designed. Solutions that have backward errors close to machine precision are referred to as ‘backward stable’, and algorithms that always compute backward-stable solutions are themselves backward or numerically stable. At the end of this chapter in Section 4.6, we show that the sketch-and-precondition framework introduced in Section 2.3.2 is in fact not numerically stable, despite being claimed to be so in [14]. Chapter 5 is consequently concerned with stabilizing the sketch-and-precondition framework and with developing an algorithm based on randomized preconditioning that always computes backward stable solutions.

In Chapter 6, we rather focus on minimizing the *forward* error and the error with respect to some ground truth. It is necessary to regularize the LS problem to achieve this, changing the structure of (4.1). The perturbation theory and experiments in this chapter demonstrate why it is necessary to regularize LS problems if the goal is a good solution $\hat{x}$ in terms of forward error.

4.2 Exact and approximate solutions to LS problems

4.2.1 The exact solution to a LS problem

We denote by x^* the solution to (4.1), that is:

$$x^* = \operatorname{argmin}_{x \in \mathbb{R}^n} \|Ax - b\|_2. \tag{4.2}$$

If A has full column rank, then there exists a unique minimizer x^* . For rank-deficient problems, the minimizer is not unique. As a resolution, we are often interested in the minimum-norm solution amongst the set of minimizers (following [140]). We denote this solution by x^* . Unlike for full-rank square problems, it will not generally be true that $Ax^* = b$. An important quantity in least squares problems is the residual $r = b - Ax$. We denote the optimal residual as

$$r^* = b - Ax^*. \quad (4.3)$$

The norm of the optimal (relative) residual is a measure for the *consistency* of the least squares problem. We say (4.1) is consistent if $r^* = 0$; the larger the norm of the residual, the more inconsistent the problem is.

There are various ways to express the solution to (4.1). We first note that x^* can be expressed as the solution to the normal equations

$$A^T Ax^* = A^T b.$$

This square system is always consistent (even when A is low-rank). An immediate consequence is that the optimal residual r^* is orthogonal to the column space of A :

$$A^T(b - Ax^*) = 0.$$

The singular value decomposition (SVD) offers a useful tool to further inspect x^* . Let $A = U\Sigma V^T$ be an SVD of a rank r matrix where the columns of $U \in \mathbb{R}^{m \times r}$ span the range of A , and let $U_\perp \in \mathbb{R}^{m \times (m-r)}$ be the orthogonal complement of U . Furthermore, let u_i , v_i , and σ_i denote the left singular vectors, right singular vectors, and singular values of A , respectively. The exact (minimum-norm) solution can be expressed as [137]

$$x^* = A^\dagger b = V\Sigma^{-1}U^T b = \sum_{i=1}^r \frac{u_i^T b}{\sigma_i} v_i,$$

and the residual satisfies

$$\|r^*\|^2 = \|Ax^* - b\|^2 = \|U^T b - \Sigma V^T x^*\|^2 + \|U_\perp^T b\|^2 = \|U_\perp^T b\|^2.$$

The norm of the residual is exactly equal to the norm of b projected onto the orthogonal complement of the range of A .

An alternative characterization of x^* and r^* is provided by the augmented system [34]:

$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}.$$

4.2.2 Approximate solutions to a LS problem

It is impossible to numerically compute x^* in finite precision. Instead, we compute an approximate solution $\hat{x}$ to the least squares problem (4.1) and hope that it is a “good” solution according to some definition. We argue there are two main factors that impact the quality of an approximate solution. On the one hand, there is the inherent sensitivity of the problem to perturbations. This sensitivity exists outside of the context of computing and depends on the conditioning of the problem. Perturbation theory considers $\hat{x}$ as the exact solution to a nearby, perturbed problem, and then quantifies upper bounds on the size of the error $\|x^* - \hat{x}\|$ and distance from the optimal residual $\|r^* - \hat{r}\|$. On the other hand there are errors introduced by algorithms that may amplify this sensitivity, or compute solutions that cannot be expressed as exact solutions of nearby problems. This is related to the stability of the particular algorithm that is used. We discuss both concepts in this section.

4.2.3 Perturbation theory

In practically all real contexts, the best we can do when setting out to solve (4.1) is to solve a perturbed problem. That is, we actually compute a solution to the problem:

$$\min_{x \in \mathbb{R}^n} \|(b + \Delta b) - (A + \Delta A)x\|.$$

The perturbations Δb and ΔA can arise due to multiple sources. Most intuitively, one should note that data often has measurement errors that can perturb the true A and b . In this case, we assume some underlying model $A_{\text{true}}x_{\text{true}} = b_{\text{true}}$ or $A_{\text{true}}x_{\text{true}} \approx b_{\text{true}}$, but assume that we only have access to perturbed versions. This is the case, for instance, in many inverse integral problems, in which the operator A is a discretization of some integral operator, and b is a measurement. Ideally, we solve $Ax_{\text{true}} = b_{\text{true}}$, but we end up attempting to numerically compute a solution to $\min \|Ax - b\|$, where $b = b_{\text{true}} + e$ is perturbed by some noise e .

Additionally, without assuming any errors in the collected data, we note that by representing A and b , as well as x^* , in finite precision, one may incur rounding errors.

We are interested in the sensitivity of our solutions to these perturbations. As for square linear systems, this is highly dependent on the condition number of the matrix A , $\kappa_2(A)$. The condition

number of a rectangular matrix is defined as

$$\kappa_2(A) = \|A\| \|A^\dagger\| = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)},$$

where $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$ denote the largest and smallest singular values of A , respectively. The condition number of any matrix is lower bounded by 1, which would be a matrix with orthogonal columns. Within the context of double precision arithmetic, where machine precision $u \approx 10^{-16}$, we refer to matrices as being moderately ill-conditioned when $\kappa_2(A) \approx 10^8$, and extremely ill-conditioned when $\kappa_2(A) \approx 10^{16}$. A matrix with condition number smaller than, say, 10^4 , would often be considered well-conditioned. We use this terminology because we are particularly interested in the numerical behaviour of iterative solvers for systems with condition number of at least $\kappa_2(A) \geq 10^8$.

The following perturbation theory due to Wedin [272] provides us with attainable upper bounds on the error and residual in terms of the conditioning of the problem. See also [140] and [34] for details.

Theorem 4.1 (Wedin, 1973). *Let $A \in \mathbb{R}^{m \times n}$ be a full rank matrix, and let x^* denote the solution to the LS problem $\min \|Ax - b\|$ with optimal residual $r^* = b - Ax^*$. Consider the perturbed problem $\min \|(b + \Delta b) - (A + \Delta A)x\|$ with minimizer $\hat{x}$ and residual $\hat{r}$. Assume the perturbations satisfy*

$$\frac{\|\Delta A\|}{\|A\|} \leq \varepsilon, \quad \frac{\|\Delta b\|}{\|b\|} \leq \varepsilon,$$

and that $\kappa_2(A)\varepsilon < 1$. Then

$$\frac{\|x^* - \hat{x}\|}{\|x^*\|} \leq \frac{\kappa_2(A)\varepsilon}{1 - \kappa_2(A)\varepsilon} \left(1 + \frac{\|b\|}{\|A\|\|x^*\|} + \frac{\kappa_2(A)\|r^*\|}{\|A\|\|x^*\|} \right), \quad (4.4)$$

and

$$\|r^* - \hat{r}\| \leq \varepsilon (\kappa_2(A)\|r^*\| + \|A\|\|x^*\| + \|b\|). \quad (4.5)$$

Consider first the bound on the error (4.4); in this case we distinguish between two different cases. If $\kappa_2(A)\|r^*\| \leq 1$, that is, the optimal residual is small compared to the condition number of A , we see that the dependence is

$$\frac{\|x^* - \hat{x}\|}{\|x^*\|} \lesssim \kappa_2(A)\varepsilon.$$

Alternatively, if the optimal residual is large compared to the condition number ($\kappa_2(A)\|r^*\| > 1$) then,

$$\frac{\|x^* - \hat{x}\|}{\|x^*\|} \lesssim \|r^*\| \kappa_2(A)^2 \varepsilon.$$

This explicit dependency on the sensitivity of b through the presence of r^* is unlike the case of square linear systems [250], and it is noteworthy that one suffers from $\kappa_2(A)^2$ dependency in the

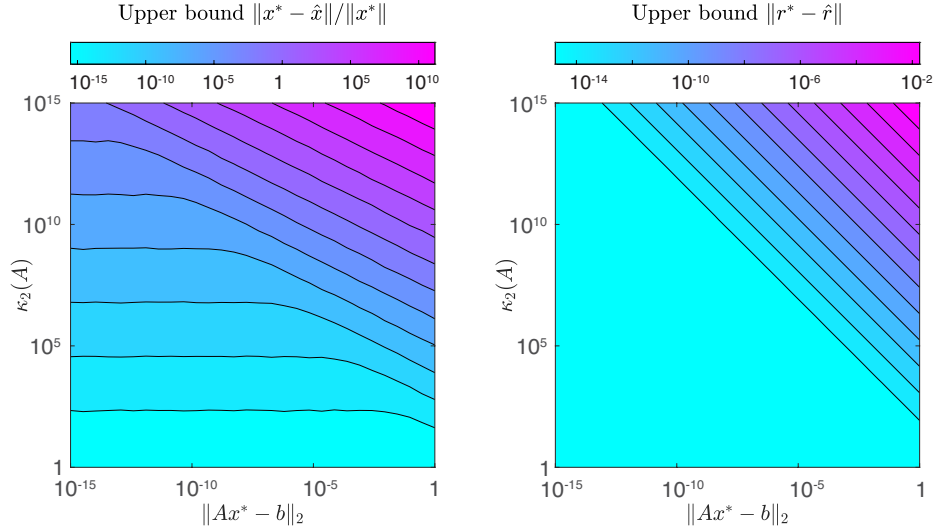


Figure 4.1: The perturbation bounds of Theorem 4.1 with ε equal to machine precision $u \approx 10^{-16}$.

error for (highly) inconsistent systems [140]. As for the residual, we see that if $\kappa_2(A)\|r^*\| \leq 1$, the residuals are close up to the size of the perturbation: $\|r^* - \hat{r}\| \lesssim \varepsilon$. If $\kappa_2(A)\|r^*\| > 1$, we have $\|r^* - \hat{r}\|/\|r^*\| \lesssim \varepsilon\kappa_2(A)$. We visually display the upper bounds from Theorem 4.1 in Figure 4.1, assuming a perturbation $\varepsilon = u$, where $u \approx 10^{-16}$ is machine tolerance in double precision. Both panels in Figure 4.1 portray the change in regime when moving from the left bottom to the right top, as $\kappa_2(A)\|r^*\|$ becomes greater than 1. These bounds are generally attained by direct backward stable solvers, such as using a QR decomposition; this can be seen in Figure 4.2.

4.2.4 Backward error and backward stability

We next turn to computational errors that might affect the quality of the computed solutions. The most natural quantity to consider is the backward error of a computed solution. Suppose we aim to solve the problem $\min \|Ax - b\|$; we run some algorithm to solve the problem, and it outputs an approximate solution $\hat{x}$. The backward error of this solution is defined as [264]:

$$\eta_F(\hat{x}) := \min\{\|[\Delta A, \Delta b]\|_F : \hat{x} = \operatorname{argmin}_{x \in \mathbb{R}^n} \|(A + \Delta A)x - (b + \Delta b)\|\}. \quad (4.6)$$

In words, how far away is the closest problem that $\hat{x}$ solves exactly? In double precision arithmetic, we say a solution is backward stable if its backward error is a small multiple of machine precision u . An algorithm is backward stable if it always computes backward stable solutions, regardless of the specific problem properties (such as ill-conditioning). Backward stability, unlike properties such as the error $\|x^* - \hat{x}\|$ or difference in residual $\|r^* - \hat{r}\|$, is a concept that allows us to state that we have computed the best possible solution we can expect to compute.

Theorem 4.1 clearly states the relationship between the backward error and the forward error: backward stability corresponds to small perturbations, and as such corresponds to the tightest possible bound on the forward error. Of course, if there are measurement errors that are significantly bigger than the backward error, they will dominate the size of the perturbation and limit the accuracy of $\hat{x}$. This is why, in practice, we aim for a backward error smaller than the size of other errors in the data.

Backward stability is important when a number of LS problems are solved as part of larger computational routine. In this case, backward errors might accumulate and eventually cause errors greater than acceptable.

4.2.5 Direct methods and their stability

So, what algorithms solving (4.1) are backward stable? Most commonly known, QR solvers based on Householder or Givens rotations will compute backward stable solutions [140]. Note that given a (thin) QR decomposition $A = QR$, where $R \in \mathbb{R}^{n \times n}$ is upper triangular and $Q \in \mathbb{R}^{m \times n}$ has orthogonal columns, the solution $x^* = R^{-1}Q^Tb$. If we compute an approximate solution $\hat{x}$ using this method, then the backward error is proportional to a small multiple of machine precision [140]. In fact, if the modified Gram-Schmidt procedure is employed on the augmented matrix $[A, b]$, one can also find a backward stable solution with careful computation [33].

As is also commonly known, using the normal equations to solve a LS problem causes instability [140, Ch. 20]. A common method, however, is still to compute the Gram matrix $C = A^T A$, and then compute the Cholesky or LU factorization of $C = R^T R$. Since $A^T A x^* = A^T b$, we can then solve $R^T y = A^T b$ and $R x^* = y$ to compute $\hat{x}$. As noted in [143], we then find that the backward error bound contains a factor $\kappa_2(A)$. Higham [140] notes that as a result, the forward error bound worsens to

$$\frac{\|x^* - \hat{x}\|}{\|x^*\|} \lesssim u \kappa_2(A)^2, \quad (4.7)$$

where u is machine precision. Although we might find this particular dependency also for a backward stable method if $\kappa_2(A)\|r^*\| > 1$, it can be much worse than optimal for consistent problems.

We perform an experiment to display the behaviour of direct methods to solve (4.1). The experiments are performed in MATLAB [144] using `backslash`, either on the rectangular system or on the square normal equations. For the rectangular system, `backslash` performs a QR factorization to solve for x ; for the normal equations MATLAB attempts a Cholesky factorization $A^T A = R^T R$ and computes $\hat{x}$ with two triangular solves. If that fails, which happens when $\kappa_2(A) \gtrsim 10^8$, it instead computes an LU factorization of $A^T A = LU$, and solves the linear system using triangular solves with L and U .

We construct LS problems at random by setting $A = U\Sigma V^T$, where $U \in \mathbb{R}^{m \times n}$ and $V \in \mathbb{R}^{n \times n}$ are random orthogonal matrices from the Haar distribution and $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_n)$, where $\sigma_1, \dots, \sigma_n$ are logarithmically spaced between $\sigma_1 = 1$ and $\sigma_n = (\kappa_2(A))^{-1}$. We set $m = 2000$ and $n = 100$. The right-hand side of the LS problem is generated as $b = Ax^* + r^*$, where x^* is a random vector with i.i.d. standard Gaussian entries and r^* represents a noise vector. The noise is chosen to be orthogonal to the column space of A so that x^* is the exact solution to $\min \|Ax - b\|$. As a result, the computed residual is bounded from below by $\|r^*\|_2$. We design 900 such problems, varying the condition number $\kappa_2(A)$ and the optimal residual r^* . The results of the experiment are shown in Figure 4.2.

Figure 4.2 shows the backward error $\eta_F(\hat{x})$, the relative (forward) error $\|x^* - \hat{x}\|/\|x^*\|$, and the relative residual $\|b - A\hat{x}\|/\|b\|$ attained using the QR factorization or the normal equations. We first consider the results for the QR method (left in Figure 4.2). The top panel shows that regardless of the conditioning of the problem, we have a backward error of order machine precision, as expected. In the middle row, we see that the QR method always attains an optimal relative residual, even for very ill-conditioned and highly consistent problems. The optimal residual is the residual attained by the exact solution x^* . In the bottom row, we can see that the (forward) error is in line with Wedin's perturbation bounds in Theorem 4.1. The relative residual and error are qualitatively and quantitatively very close to the upper bounds shown in Figure 4.1.

In the right panels of Figure 4.2, we first note two different algorithms are used. Both the Cholesky decomposition and the LU factorization are (componentwise) backward stable to solve a positive definite square linear system $Cx = d$ in the sense that [140]

$$(C + \Delta C)\hat{x} = d, \quad |\Delta C| \lesssim \mathcal{O}(u|C|),$$

where $\mathcal{O}(u)$ is a low-degree polynomial in machine precision u and $|C|$ denotes the matrix $|C|_{ij} = |C_{ij}|$. However, as the backward error plot shows, this does not translate to a backward stable method to solve LS problems (also noted in [143]).

The top-right panel of Figure 4.2 shows the dependency that the backward error achieved through solving the normal equations depends on $\kappa_2(A)$, as was noted in [143]. We also note this effect is diminished for very inconsistent systems.

The errors clearly show a sensitivity behaving like $\kappa_2(A)^2$ to the conditioning, as predicted in (4.7). Notably, for highly consistent problems, the computed residual is also suboptimal, even when the condition number is moderate.

Although commonly regarded as the dominant direct LS solver due to its excellent numerical properties, the QR algorithm requires $2n^2(m - n/3)$ flops [140]. The normal equations algorithm is

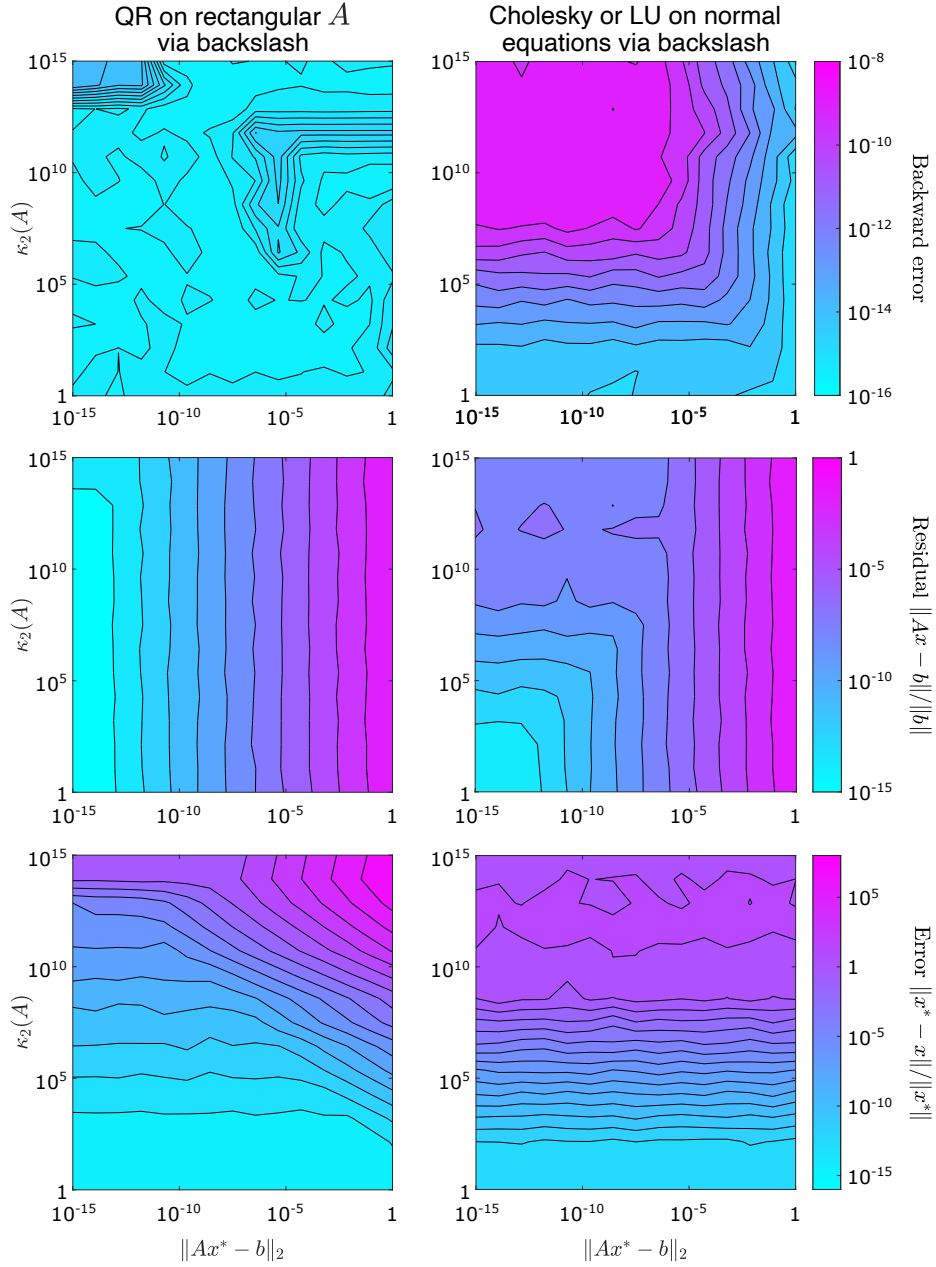


Figure 4.2: The backward error (top), attained residual (middle), and forward error (bottom) when using MATLAB's `backslash` on either the rectangular A ($x = A \backslash b$) or on the normal equations via $x = (A' * A) \backslash (A' * b)$. MATLAB performs a Householder QR-based routine in the rectangular case, and a Cholesky or LU factorization-based routine for the square case. The matrix A is 2000×100 with exponentially decaying singular values ranging from $\sigma_1(A) = 1$ to $\sigma_{100}(A) = (\kappa_2(A))^{-1}$ and Haar distributed singular vectors. The solution x is a random Gaussian vector and b is created as $b = A * x + \text{noise}$, where the noise is orthogonal to the range of A and of specified magnitude.

twice as fast in flop count, but must be used with care in the presence of ill-conditioning, as Figure 4.2 shows.

A completely separate class of LS solvers are iterative solvers, to which the remainder of this chapter is dedicated. Although the behaviour of these solvers in exact arithmetic and their theo-

retical convergence behaviour is well-understood, their behaviour in finite precision is much more theoretically opaque. We will investigate the numerical stability of various solvers through experiments as in Figure 4.2. We start by introducing LSQR [205] and LSMR [103], two popular iterative Krylov solvers for LS problems.

4.3 Iterative Krylov subspace solvers: LSQR and LSMR

Iterative solvers for linear systems are a type of algorithm that, on each iteration, aim to improve the quality of the current iterate. A Krylov subspace solver does this by growing the subspace in which it looks for solutions. This subspace is the Krylov subspace, which is defined for a square matrix C and a starting vector d as

$$\mathcal{K}_k(C, d) = \text{span} \{d, Cd, \dots, C^{k-1}d\}.$$

In the LS case that subspace is, at the k th iteration,

$$\mathcal{K}_k(A^T A, A^T b) = \text{span} \{A^T b, (A^T A)A^T b, \dots, (A^T A)^{k-1}A^T b\}, \quad (4.8)$$

which is the Krylov subspace associated to the normal equations $A^T A x = A^T b$.

Conjugate gradients (CG), invented in 1952 by Hestenes and Stiefel [139], is the first and most famous of the Krylov subspace methods [250]. It is used on symmetric positive definite square linear systems $Cx = d$, and on each iteration minimises the C -norm of the (forward) error. That is, letting x^* denote the exact solution to $Cx = d$, we look for iterates

$$x_k = \underset{x \in \mathcal{K}_k(C, d)}{\text{argmin}} \|x - x^*\|_C^2 = (x^* - x)^T C (x^* - x).$$

By exploiting the symmetric positive definite nature of the problem, one can compute x_k extremely efficiently, using just one matrix-vector product and a few inner products per iteration. We will not provide a detailed description of this algorithm (see e.g. [119]), but instead show its equivalence in exact arithmetic to LSQR so we may use CG convergence theory.

LSQR is an iterative Krylov subspace solver for LS problems that, at an iteration k , computes an iterate x_k that minimizes the residual $\|Ax - b\|_2$ over $\mathcal{K}_k$. That is,

$$x_k = \underset{x \in \mathcal{K}_k(A^T A, A^T b)}{\text{argmin}} \|Ax - b\|_2.$$

It is straightforward to show that for $x^* = (A^T A)^{-1} A^T b$:

$$\|x - x^*\|_{A^T A}^2 = \|Ax - b\|_2^2 + \|Ax^* - b\|_2^2,$$

which means that

$$\operatorname{argmin}_{x \in \mathcal{K}_k(A^T A, A^T b)} \|x - x^*\|_{A^T A} = \operatorname{argmin}_{x \in \mathcal{K}_k(A^T A, A^T b)} \|Ax - b\|_2.$$

That is, LSQR is mathematically equivalent to CG on the normal equations.

LSMR (Least Squares Minimum Residual) is another Krylov subspace solver for LS problems, built on the same ingredients as LSQR. Instead of minimizing the residual $\|Ax - b\|$, LSMR minimizes the normal residual at each iteration:

$$x_k = \operatorname{argmin}_{x \in \mathcal{K}_k(A^T A, A^T b)} \|A^T(b - Ax)\|_2.$$

This is equivalent to the Minimal Residual Method (MINRES) algorithm applied to the normal equations. This equivalence is immediate from the fact that MINRES is a Krylov subspace solver for real and symmetric linear systems $Cx = d$ that minimizes the residual

$$x_k = \operatorname{argmin}_{x \in \mathcal{K}_k(C, d)} \|d - Cx\|_2.$$

The basis of both LSQR and LSMR is Golub-Kahan bidiagonalization [112]. This is an iterative method to reduce a matrix to bidiagonal form through orthogonal transformations. Using iterative matrix-vectors multiplications with A and A^T , we obtain an orthogonal basis for the relevant Krylov subspace $\mathcal{K}_k(A^T A, A^T b)$. We then project onto this basis, and by applying identities resulting from the bidiagonalization, we are left with a $(k + 1) \times k$ lower-bidiagonal least squares problems at the k th iteration. The convenient structure allows us to easily update the QR decomposition of the bidiagonal matrix, and as a result find the new iterate x_k .

4.3.1 Ingredients of LSQR and LSMR

We discuss the various ingredients of LSQR and LSMR in this section, and derive the LSQR algorithm. LSMR is discussed in less detail as its derivation is very similar in nature.

4.3.1.1 Golub-Kahan bidiagonalization

Golub-Kahan bidiagonalization [112] is a procedure to compute orthogonal matrices U and V such that $U^T A V$ is bidiagonal. The steps are as follows:

Algorithm 5 Golub-Kahan bidiagonalization. Note, α_k and β_k are chosen to ensure u_k and v_k have unit norm.

```

1:  $\beta_1 u_1 := b$ 
2:  $\alpha_1 v_1 := A^T u_1$ 
3: for  $k = 1, 2, \dots$  do
4:    $\beta_{k+1} u_{k+1} = Av_k - \alpha_k u_k$ 
5:    $\alpha_{k+1} v_{k+1} = A^T u_{k+1} - \beta_{k+1} v_k$ 
6: end for

```

The vectors $v_1, \dots, v_k$ span an orthonormal basis for the Krylov subspace $\mathcal{K}_k(A^T A, A^T b)$, and the vectors $u_1, \dots, u_k$ span $\mathcal{K}_k(AA^T, b)$. Furthermore, by collecting the vectors v_k and u_k in matrices $V_k \in \mathbb{R}^{n \times k}$ and $U_k \in \mathbb{R}^{m \times k}$ respectively, we have

$$AV_k = U_{k+1}B_k, \quad A^T U_k = V_k L_k^T, \quad (4.9)$$

where

$$L_k = \begin{bmatrix} \alpha_1 & & & & \\ \beta_2 & \alpha_2 & & & \\ & \ddots & \ddots & & \\ & & \beta_k & \alpha_k & \end{bmatrix}, \quad B_k = \begin{bmatrix} L_k \\ \beta_{k+1} e_k^T \end{bmatrix} = \begin{bmatrix} \alpha_1 & & & & \\ \beta_2 & \alpha_2 & & & \\ & \beta_3 & \ddots & & \\ & & \ddots & \ddots & \alpha_k \\ & & & \beta_{k+1} & \end{bmatrix}. \quad (4.10)$$

The vector e_k denotes the k -dimensional vector with zeros everywhere except in the k th entry, which is 1.

4.3.1.2 Projection onto the Krylov subspace and minimization

As the columns of V_k span a basis for the Krylov subspace $\mathcal{K}_k := \mathcal{K}_k(A^T A, A^T b)$, we may express any $x_k \in \mathcal{K}_k$ as $x_k = V_k y_k$ for some $y_k \in \mathbb{R}^k$. This leads to the following identities for the minimizations involved in LSQR and LSMR.

$$\min_{x \in \mathcal{K}_k} \|b - Ax\| = \min_{y \in \mathbb{R}^k} \|b - AV_k y\| = \min_{y \in \mathbb{R}^k} \|U_{k+1}(\beta_1 e_1 - B_k y)\| = \min_{y \in \mathbb{R}^k} \|\beta_1 e_1 - B_k y\|, \quad (4.11)$$

where $e_1 \in \mathbb{R}^{k+1}$ is the first canonical basis vector with 1 in the first coordinate and 0 everywhere else. Similarly,

$$\min_{x \in \mathcal{K}_k} \|A^T(b - Ax)\| = \min_{y \in \mathbb{R}^k} \|V_{k+1} L_{k+1}^T(\beta_1 e_1 - B_k y)\| = \min_{y \in \mathbb{R}^k} \|\alpha_1 \beta_1 e_1 - L_{k+1}^T B_k y\|. \quad (4.12)$$

We can solve (4.11) and (4.12) using QR decompositions of B_k and $L_{k+1}^T B_k$, which we discuss next.

4.3.1.3 Updating the QR factorization

We will only consider the QR factorization of B_k for the LSQR algorithm here, as the derivation is somewhat tedious. The ideas are very similar for LSMR. In fact, instead of computing the QR decomposition of B_k we decompose the augmented matrix $[B_k, \beta_1 e_1]$ using Givens rotations. The final result is

$$Q_{k+1}^T \begin{bmatrix} \alpha_1 & & & \beta_1 \\ \beta_2 & \alpha_2 & & 0 \\ & \beta_3 & \ddots & \vdots \\ & & \ddots & \alpha_k & 0 \\ & & & \beta_{k+1} & 0 \end{bmatrix} = \begin{bmatrix} \rho_1 & \theta_2 & & \phi_1 \\ & \rho_2 & \ddots & \phi_2 \\ & & \ddots & \theta_k & \vdots \\ & & & \rho_k & \phi_k \\ & & & & \hat{\phi}_{k+1} \end{bmatrix} = \begin{bmatrix} R_k & f_k \\ 0 & \hat{\phi}_{k+1} \end{bmatrix}, \quad (4.13)$$

where Q_{k+1}^T is a product of Givens rotations, say $Q_{k+1}^T = J(k, k+1) \dots J(1, 2)$, where

$$J(i, i+1) = \begin{bmatrix} I & & \\ & c_i & s_i \\ & -s_i & c_i \\ & & & I \end{bmatrix}$$

only acts on the i th and $(i+1)$ th rows. In particular, these rotations will be of the form

$$\begin{bmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{bmatrix} \begin{bmatrix} \alpha_1 & 0 & \beta_1 \\ \beta_2 & \alpha_2 & 0 \end{bmatrix} = \begin{bmatrix} \rho_1 & \theta_2 & \phi_1 \\ 0 & \hat{\rho}_2 & \hat{\phi}_2 \end{bmatrix}, \quad (4.14)$$

$$\begin{bmatrix} c_i & s_i \\ -s_i & c_i \end{bmatrix} \begin{bmatrix} \hat{\rho}_i & 0 & \hat{\phi}_i \\ \beta_{i+1} & \alpha_{i+1} & 0 \end{bmatrix} = \begin{bmatrix} \rho_i & \theta_{i+1} & \phi_i \\ 0 & \hat{\rho}_{i+1} & \hat{\phi}_{i+1} \end{bmatrix}, \quad i > 1 \quad (4.15)$$

We can obtain recursive formulas using the structure of the rotations. From the definition of a Givens matrix, we have

$$\rho_1 = \sqrt{\alpha_1^2 + \beta_2^2}, \quad s_1 = \frac{\beta_2}{\rho_1}, \quad c_1 = \frac{\alpha_1}{\rho_1},$$

which allows us to define $\theta_2, \hat{\rho}_2, \phi_1$, and $\hat{\phi}_2$. Then, recursively, for $i > 1$

$$\rho_i = \sqrt{\hat{\rho}_i^2 + \beta_{i+1}^2}, \quad s_i = \frac{\beta_{i+1}}{\rho_i}, \quad c_i = \frac{\hat{\rho}_i}{\rho_i}.$$

This leads us to a sequence of updates defined in Algorithm 6.

Algorithm 6 A recursively defined QR decomposition of the augmented matrix $[B_k, \beta_1 e_1]$, where B_k is defined as in (4.14) and ρ_i, θ_i , and ϕ_i define the R -factor of the decomposition as in (4.13).

```

1:  $\hat{\rho}_1 = \alpha_1$ 
2:  $\hat{\phi}_1 = \beta_1$ 
3: for  $i = 1, 2, \dots, k$  do
4:    $\rho_i = \sqrt{\hat{\rho}_i^2 + \beta_{i+1}^2}$ 
5:    $s_i = \beta_{i+1}/\rho_i$ 
6:    $c_i = \hat{\rho}_i/\rho_i$ 
7:    $\theta_{i+1} = s_i \alpha_{i+1}$ 
8:    $\phi_i = c_i \hat{\phi}_i$ 
9:    $\hat{\phi}_{i+1} = -s_i \hat{\phi}_i$ 
10:   $\hat{\rho}_{i+1} = c_i \alpha_{i+1}$ 
11: end for
```

4.3.1.4 Updating the iterates

Now that we have a QR factorization of the problem, we can update the iterate x_k accordingly. We have

$$\|b - Ax_k\|^2 = \|b - AV_k y_k\|^2 = \|B_k y_k - \beta_1 e_1\|^2 = \left\| \begin{bmatrix} R_k y_k - f_k \\ -\hat{\phi}_{k+1} \end{bmatrix} \right\|^2 = |\hat{\phi}_{k+1}|^2,$$

and $R_k y_k = f_k$ with $x_k = V_k y_k$. We do not need to explicitly compute y_k , but instead we can recursively add columns to $D_k = V_k R_k^{-1}$. This equation can be solved with forward substitution by noting that $R_k^T D_k^T = V_k^T$ implies

$$d_1 = \frac{1}{\rho_1} v_1^T, \quad d_{i+1} = \frac{1}{\rho_{i+1}} (v_{i+1} - \theta_{i+1} d_i),$$

where the d_i are the columns of D_k . Then finally, since $x_k = D_k f_k$, we have

$$x_k = \sum_{i=1}^k \phi_i d_i = x_{k-1} + \phi_k d_k.$$

We can now state the full LSQR algorithm.

Algorithm 7 The LSQR algorithm to solve $\min \|Ax - b\|$.

```

1:  $\beta_1 u_1 = b$ 
2:  $\alpha_1 v_1 = A^T u_1$ 
3:  $\hat{\rho}_1 = \alpha_1, \hat{\phi}_1 = \beta_1$ 
4:  $\theta_1 = 0, d_0 = 0$ 
5: for  $i = 1, 2, \dots$  do
    1. Golub-Kahan bidiagonalization:
6:    $\beta_{i+1} u_{i+1} = A v_i - \alpha_i u_i$ 
7:    $\alpha_{i+1} v_{i+1} = A^T u_{i+1} - \beta_{i+1} v_i$ 
    2. Update QR decomposition:
8:    $\rho_i = \sqrt{\hat{\rho}_1^2 + \beta_{i+1}^2}$ 
9:    $s_i = \beta_{i+1} / \rho_i$ 
10:   $c_i = \hat{\rho}_1 / \rho_i$ 
11:   $\theta_{i+1} = s_i \alpha_{i+1}$ 
12:   $\phi_i = c_i \hat{\phi}_i$ 
13:   $\hat{\phi}_{i+1} = -s_i \hat{\phi}_i$ 
14:   $\hat{\rho}_{i+1} = c_i \alpha_{i+1}$ 
    3. Update iterate:
15:    $d_i = (v_i - \theta_i d_{i-1}) / \rho_i$ 
16:    $x_i = x_{i-1} + \phi_i d_i$ 
17: end for
```

4.3.2 Stopping criteria

As should be clear from the discussion in Section 4.2.3, we would ideally terminate the algorithm if the backward error reaches a user-specified tolerance, informed by acceptable errors.

The original stopping criteria for LSQR and LSMR, and those still implemented in MATLAB, are based on this reasoning. Specifically, we have:

S1 Stop if $\|r_k\| \leq \tau_b + \tau_A \|x_k\|$.

S2 Stop if $\|A^T r_k\|/\|r_k\| \leq \tau_A$.

S3 Stop if $\tilde{\kappa}_2(A) > \kappa_{\max}$.

Here, the tolerances τ_A and τ_b are (estimates for) the magnitude of measurement error. Let A_{true} and b_{true} be the ‘true’ data without error, then ideally $\tau_A = \|A - A_{\text{true}}\|$ and $\tau_b = \|b - b_{\text{true}}\|$. Although these quantities are often not known in practice, the tolerances still provide a way for the user to specify the level of desired accuracy.

The first two stopping criteria are related to the backward error defined in (4.6). Recall that in the perturbed problem an iterate x_k exactly solves

$$(A + \Delta A)x_k = b + \Delta b,$$

and by the definition of the backward error, we desire the quantities ΔA and Δb to satisfy

$$\|\Delta A\| \leq \|A - A_{\text{true}}\|, \quad \|\Delta b\| \leq \|b - b_{\text{true}}\|.$$

Define

$$\delta_k = \tau_b + \tau_A \|x_k\|,$$

and note that

$$\left(A + \tau_A \frac{r_k x_k^T}{\delta_k \|x_k\|} \right) x_k = Ax_k + \tau_A \|x_k\| \frac{r_k}{\delta_k} = b - \tau_b \frac{r_k}{\delta_k},$$

where the last inequality follows from the identity $(\tau_b + \tau_A \|x_k\|)r_k/\delta_k = r_k$. In other words, x_k exactly solves a perturbed system. If S1 holds, we have that $\|r_k\| \leq \delta_k$, and so

$$\|\Delta A\| \leq \tau_A \frac{\|r_k\|}{\delta_k} \leq \tau_A, \quad \|\Delta b\| = \tau_b \frac{\|r_k\|}{\delta_k} \leq \tau_b.$$

Note that $\|r_k\|$ is lower bounded by $\|r^*\|$, so that this stopping criterion might never be reached, even for a backward stable solution. The criterion S2 can be used for inconsistent systems. To see how it relates to the backward error, we note that the backward error can be computed as [264] (see also [140, Thm. 20.5])

$$\eta_F(x) = \min \left\{ \phi, \sigma_{\min} \left(\begin{bmatrix} A & \phi(I_m - rr^\dagger) \end{bmatrix} \right) \right\}, \quad \phi = \frac{\|r\|}{\sqrt{1 + \|x\|^2}}, \quad r = b - Ax.$$

From this we can find an upper bound:

$$\eta_F(x_k) \leq \sigma_{\min} \left(\begin{bmatrix} A & \phi(I_m - r_k r_k^\dagger) \end{bmatrix} \right) \leq \min_{\|z\|=1} \left\| \begin{bmatrix} A^T \\ \phi(I_m - r_k r_k^\dagger) \end{bmatrix} z \right\| \leq \frac{\|A^T r_k\|}{\|r_k\|},$$

by choosing $z = r_k / \|r_k\|$. This estimate was first noted by Stewart [237]. Therefore, in infinite precision, if S2 is satisfied then we should have $\|\Delta A\| \leq \tau_A$.

The purpose of the final criterion S3 is to implicitly regularize the problem by stopping early. The value $\tilde{\kappa}_2(\max)$ is the estimate of the condition number of A which is updated throughout the iterations. In some sense, one can interpret the Krylov-subspace nature of the solver as iteratively resolving directions corresponding to singular values, starting with the largest one. If we stop before we resolve very small singular values, we have implicitly ignored the corresponding singular vector directions leading to a regularized solution.

The estimate $\tilde{\kappa}_2(\max)$ is computed through the scalar values used in the LSQR iterations. Note first that $B_k^T B_k = V_k^T A^T A V_k$ so that $B_k^T B_k$ and $A_k^T A_k$ have interlacing eigenvalues by the Courant-Fischer theorem [114], and

$$\sigma_{\min}(A) \leq \sigma_{\min}(B_k) \leq \sigma_{\max}(B_k) \leq \sigma_{\max}(A).$$

Similarly, since $B_k^T B_k = R_k^T R_k$ and $D_k = V_k R_k^{-1}$, we have $\sigma_{\min}(B_k) = \sigma_{\min}(R_k) = \|D_k\|^{-1}$. This leads to a lower bound on the condition number,

$$\kappa_2(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)} \|B_k\| \|D_k\|.$$

Instead of using the spectral norm, the stopping criterion uses the Frobenius norm of B_k and D_k which can be accumulated cheaply through the identities

$$\|B_k\|_F^2 = \|B_{k-1}\|_F^2 + \alpha_k^2 + \beta_{k+1}^2, \quad \|D_k\|_F^2 = \|D_{k-1}\|_F^2 + \|d_k\|^2.$$

The LSQR stopping criteria are, however, a delicate aspect, and significant research has been devoted to the subject [52, 129, 238]. We refer to [128] for an excellent overview.

4.4 Convergence behaviour of LSQR and LSMR

The behaviour of Krylov subspace solvers in exact arithmetic can differ significantly from finite precision behaviour. We briefly discuss both in this section.

4.4.1 Convergence in exact arithmetic

The convergence of many Krylov subspace methods for symmetric systems is determined by the spectral properties of the matrix. For an $m \times n$ matrix A , it is well-known that both LSQR and LSMR in exact arithmetic converge after at most n iterations. This can be deduced from the

definition of the Krylov subspace (4.8), which spans all of $\mathbb{R}^n$ after n iterations. If A has t distinct singular values, they converge after at most t steps [137].

For LSQR, the decrease in the norm of the residual $r_k = b - Ax_k$ satisfies

$$\|r^* - r_k\| < 2 \left(\frac{\kappa_2(A) - 1}{\kappa_2(A) + 1} \right)^k \|r^* - r_0\|, \quad (4.16)$$

where r^* is the optimal residual [34]. We can use this result to bound the number of iterations necessary to reach a certain accuracy ε [34]:

$$\frac{\|r^* - r_k\|}{\|r^* - r_0\|} < \varepsilon \iff k > \frac{1}{2} \kappa_2(A) \log(2/\varepsilon).$$

The solution x_k converges to the minimum-norm solution $x^* = A^\dagger b$ [139, 203]. The (forward) error $\|x_k - x^*\|$ and the residual norm $\|r_k\|$ are both known to decay monotonically [139].

In Section 2.3.2 we wrote that sketch-and-precondition algorithms converge in $\mathcal{O}(\log(1/\varepsilon))$ iterations for ε relative accuracy. This is a result of (4.16) combined with the fact that the preconditioned matrix has a condition number of $\mathcal{O}(1)$.

We can derive the convergence bounds for LSMR by considering the convergence theory for MINRES applied to positive definite systems. Using Greenbaum's result [119, Eq. (3.12)], we find that LSMR iterates x_k satisfy

$$\|A^T(Ax_k - b)\| \leq 2 \left(\frac{\kappa_2(A) - 1}{\kappa_2(A) + 1} \right)^k \|A^T(Ax_0 - b)\|. \quad (4.17)$$

These iterates are also well-known to have monotonically decaying error, residual, and normal residual. LSMR might be preferred to LSQR exactly because $\|A^T r_k\|$ is monotonically decreasing, allowing us to stop early more safely [103].

4.4.2 Finite precision behaviour

In finite precision, LSQR and LSMR might need considerably more than n iterations to converge. This is caused (at least in part) by the loss of orthogonality amongst the basis vectors for the Krylov subspace. The results in Section 4.4.1 are based on the vectors $v_1, \dots, v_k$ being an orthogonal basis for the Krylov subspace $\mathcal{K}_k(A^T A, A^T b)$. It is not obvious that the convergence bounds (4.16) and (4.17) should also apply to finite precision [119].

We refer to [119] and [188] for discussions of the finite precision behaviour of CG and MINRES, the algorithms for symmetric systems from which LSQR and LSMR can be derived. The analyses of these algorithms are often rooted in finite precision analysis of the Lanczos algorithm, which is the symmetric counterpart to Golub-Kahan bidiagonalization, and Paige's revolutionary work into

the subject [202]. It is clear there is some maximal attainable accuracy of CG in finite precision due to the rank-deficiencies in the Krylov subspace [188]. Various works seem to strongly hint at backward stability of CG under assumptions on the condition number, but use computational results to complement the claim, see [32, 118]. A recent result by Musco, Musco and Sidford [193, Thm. 2.1], when specialized to well-conditioned positive definite linear systems, implies that ε forward error is achieved by using $\mathcal{O}(\log \frac{1}{\varepsilon})$ bits, using the Lanczos method with modified Gram-Schmidt orthogonalization. For well-conditioned linear systems, taking $\varepsilon = u$, this would imply that the solution has an $\mathcal{O}(u)$ backward error, i.e. it is backward stable.

However, the performances of CG and MINRES in finite precision depend strongly on the exact implementation [168]. For this reason, it is not at all straightforward to extend results on CG and the Lanczos method to LSRM and LSMR, despite the clear connections in exact arithmetic. An analysis of the finite precision behaviour of precise LSQR implementations would be of great interest, but are beyond the scope of this thesis. We restrict ourselves to observations on the empirical maximal attainable accuracy similar to the investigation in Figure 4.3.

4.4.2.1 Empirical investigation of the finite precision behaviour

We perform an empirical investigation of the maximal attainable accuracy of LSQR and LSMR. We do not set a maximum number of iterations, and let the algorithms iterate until stagnation. The problems are the same as in Figure 4.2. The results are shown in Figure 4.3.

We first must comment that this does not represent realistic usage of LSQR and LSMR, but that the experiment serves to demonstrate the optimal accuracy that these algorithms can achieve offer before stagnating. For ill-conditioned matrices however, stagnation occurs after more than a million iterations. This is clearly unacceptable computational cost for a problem of dimension 2000×100 . We also note that this experiment was repeated with various different singular vectors that are not Haar distributed, and the same qualitative results were obtained.

Figure 4.3 shows immediately that for any of the considered metrics (backward error, error, residual, and normal residual), LSQR and LSMR perform similarly. Both algorithms can likely not be considered backward stable, considering the panels in the first row of Figure 4.3. For very ill-conditioned matrices with small residuals, both iterative methods cannot obtain a backward error smaller than 10^{-14} . However, it is perhaps surprising that the backward error is still relatively small, even after millions of iterations. For all well-conditioned problems, LSQR and LSMR are backward stable.

The forward errors are surprisingly small when compared to the forward error of the QR algorithm in Figure 4.2. For highly ill-conditioned and inconsistent problems, the backward stable QR algorithm lost all accuracy in terms of the forward error, as we were able to explain with the perturbation theory described in Section 4.2.3. The forward error for LSQR and LSMR appears

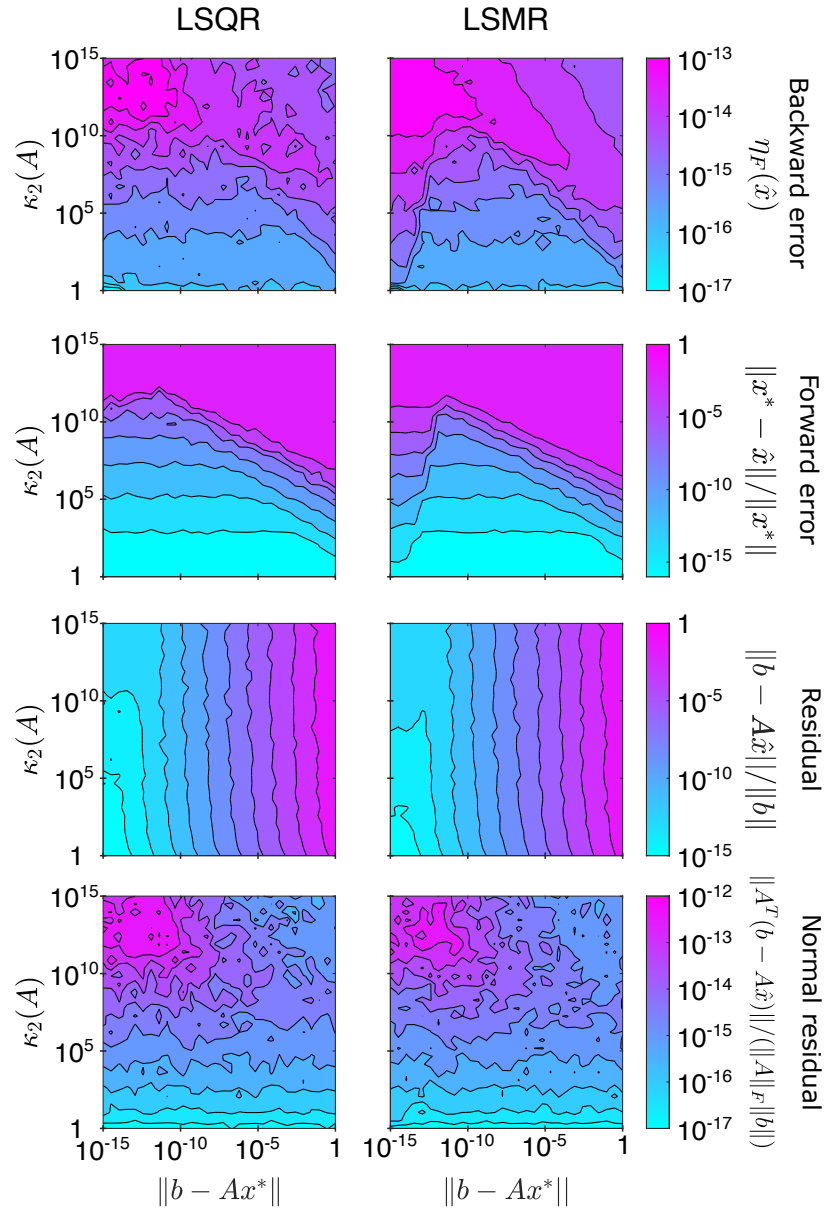


Figure 4.3: The maximal attainable accuracy of LSQR and LSMR (without a preconditioner). All problems are randomly generated 2000×100 with exponentially decaying singular values, as in Figure 4.2.

to be empirically bounded by 1 in these example problems. This relates to a regularizing effect of Krylov iterations [34, 137] and is related to iterative regularization algorithms (see Section 6.1.2.2).

4.5 Preconditioned LSQR

The rate of convergence and maximal attainable accuracy of LSQR is greatly influenced by the condition number $\kappa_2(A)$, as we saw in (4.16) and Figure 4.3. Indeed, to obtain the small backward errors observed in Figure 4.3 many more than n iterations were needed, even for moderately ill-conditioned problems. We can improve the conditioning through preconditioning of the problem to ensure rapid convergence and a robust solver. A preconditioner aims to ensure that the new system is sufficiently well-conditioned, so that iterative methods converge in much less than n iterations.

A (right-)preconditioner is a matrix $P \in \mathbb{R}^{n \times n}$ that can be applied quickly and approximately ‘orthogonalizes’ A , in the sense that $\kappa_2(AP)$ is small (say, $\kappa_2(AP) = \mathcal{O}(1)$). If we find such a matrix, we apply a change of variables to solve

$$w^* = \operatorname{argmin}_{w \in \mathbb{R}^{n \times n}} \|APw - b\|_2, \quad x^* = Pw^*. \quad (4.18)$$

The convergence will then be determined by $\kappa_2(AP)$ instead of $\kappa_2(A)$.

One way to algorithmically resolve the preconditioner in (4.18) is to call, for instance, LSQR with an operator $\mathbf{v} \mapsto \mathbf{A}(\mathbf{P}\mathbf{v})$ instead of $v \mapsto Av$. For instance, if $P = R^{-1}$, where R is an upper triangular matrix, we could call $\mathbf{v} \mapsto \mathbf{A}^*(R \setminus \mathbf{v})$ for the non-transposed operation. Within the LSQR routine we then compute $w^* = \operatorname{argmin} \|APw - b\|_2$ and we have to explicitly transform back to $x^* = Pw^*$, with, for example, another triangular solve.

Alternatively, we can implicitly keep track of w without ever computing it. This is the preconditioned version of LSQR implemented in MATLAB [144]. However, we are unaware of any text reference explicitly deriving this algorithm.

We start with the preconditioned variant of Golub-Kahan bidiagonalization:

$$\beta_1 u_1 := b, \quad \alpha_1 v_1 := P^T A^T u_1 \quad (4.19)$$

$$\text{For } i = 1, 2, \dots \quad (4.20)$$

$$\beta_{i+1} u_{i+1} := APv_i - \alpha_i u_i, \quad (4.21)$$

$$\alpha_{i+1} v_{i+1} := P^T A^T u_{i+1} - \beta_{i+1} v_i, \quad (4.22)$$

which leads to the identity $APV_k = U_{k+1}B_k$ in exact arithmetic. The columns of V_k and U_k now

span the Krylov subspaces:

$$\text{span}(v_1, \dots, v_k) = \mathcal{K}_k(P^T A^T A P, P^T A^T b), \quad \text{span}(u_1, \dots, u_k) = \mathcal{K}_k(A P P^T A^T, b),$$

and we essentially solve the well-conditioned problem

$$w_k = \underset{w \in \mathcal{K}_k(P^T A^T A P, P^T A^T b)}{\text{argmin}} \|A P w - b\|_2, \quad x_k = P w_k.$$

We now show how to avoid the computation of the iterates w_k , but rather immediately change back to iterates $x_k = P w_k$ without an extra application of the preconditioner. Note that we now have

$$x_k = P w_k, \quad w_k = V_k y_k, \quad R_k y_k = f_k,$$

where R_k and f_k are defined as in (4.13). As a result we find the identity $x_k = P V_k R_k^{-1} f_k$. Defining $D_k = P V_k R_k^{-1}$ and solving $R_k^T D_k^T = V_k^T P^T$ with forward substitution leads to the recursive identity

$$d_1 = \frac{1}{\rho_1} P v_1 = \frac{1}{\rho_1} z_1, \quad d_{i+1} = \frac{1}{\rho_{i+1}} (P v_{i+1} - \theta_{i+1} d_i) = \frac{1}{\rho_{i+1}} (z_{i+1} - \theta_{i+1} d_i), \quad \text{for } i = 1, 2, \dots$$

where $z_i = P v_i$. The computation of z_i is necessary as part of the next diagonalization step and so comes at no extra computational cost, although we do need to store one additional n -dimensional vector. The full algorithm is displayed in Algorithm 8.

Algorithm 8 The preconditioned LSQR algorithm to solve $\min \|A x - b\|$ with a right preconditioner P .

- 1: $\beta_1 u_1 = b$
- 2: $\tilde{v}_1 = A^T u_1$
- 3: $\alpha_1 v_1 = P^T \tilde{v}_1$
- 4: $\hat{\rho}_1 = \alpha_1, \hat{\phi}_1 = \beta_1$
- 5: $\theta_1 = 0, d_0 = 0$
- 6: **for** $i = 1, 2, \dots$ **do**
 - 1. Golub-Kahan bidiagonalization:**
 - 7: $z_i = P v_i$
 - 8: $\beta_{i+1} u_{i+1} = A z_i - \alpha_i u_i$
 - 9: $\alpha_{i+1} v_{i+1} = P^T (A^T u_{i+1}) - \beta_{i+1} v_i$
 - 2. Update QR decomposition:**
 - 10: $\rho_i = \sqrt{\hat{\rho}_1^2 + \beta_{i+1}^2}$
 - 11: $s_i = \beta_{i+1} / \rho_i$
 - 12: $c_i = \hat{\rho}_1 / \rho_i$

$$13: \quad \theta_{i+1} = s_i \alpha_{i+1}$$

$$14: \quad \phi_i = c_i \hat{\phi}_i$$

$$15: \quad \hat{\phi}_{i+1} = -s_i \hat{\phi}_i$$

$$16: \quad \hat{\rho}_{i+1} = c_i \alpha_{i+1}$$

3. Update iterate:

$$17: \quad d_i = (z_i - \theta_i d_{i-1}) / \rho_i$$

$$18: \quad x_i = x_{i-1} + \phi_i d_i$$

19: **end for**

The dominant cost in the algorithm is the bidiagonalization. At each iteration, we perform $2C_A + 2C_P$ flops, where C_A is the cost of applying A or A^T to a vector and C_P is the cost of applying P or P^T to a vector. For a dense A and $P = R^{-1}$, where R is triangular, we have $C_A = 2mn$ and $C_P = n^2$.

The number of iterations necessary now depends on the spectrum of AP , and all convergence theory from Section 4.4 carries over. The finite-precision behaviour of preconditioned Krylov subspace methods is even more elusive than the unpreconditioned variant. We are unaware of any work that investigates this. Our analysis of the stability will be purely empirical.

Since we now are interested in the spectrum of AP considering (4.16), P should be chosen such that AP has small condition number of a clustered singular value spectrum. The question that remains is how to find such a matrix P that improves the conditioning of A . Various methods exist for sparse data matrices such as an incomplete Cholesky factorization [30, 224, 230] or the Multilevel Incomplete QR factorization [166] (see [115] for a discussion on state-of-the-art methods for sparse LS). These exploit the sparsity pattern of A to cheaply compute preconditioners. For dense matrices, however, sketch-and-precondition has shown to be a powerful preconditioning method when A is highly-overdetermined [270].

4.5.1 The sketch-and-precondition framework

This sketch-and-precondition framework [222] has been discussed in detail in Section 2.3.2. This class of algorithms uses sketching to design a preconditioner for a LS problem. In the most common implementation, a tall matrix $A \in \mathbb{R}^{m \times n}$ is sketched to a smaller (yet still tall) matrix $SA \in \mathbb{R}^{s \times n}$ using a random embedding matrix $S \in \mathbb{R}^{s \times m}$, where the embedding dimension s satisfies $n \leq s \ll m$. The sketch SA is then decomposed using some classical algorithm, e.g. a QR decomposition or an SVD, and the decomposition is used to find a preconditioner P . In the QR-based algorithm for full-rank A , if $SA = QR$ then $P = R^{-1}$. If we rather compute an SVD of $SA = U\Sigma V^T$ then the preconditioner is $P = V\Sigma^{-1}$. A specific implementation can be found in Algorithm 2.

The conditioning of AP depends only on the quality of S as an embedding for the column space of A . In Lemma 2.5 we showed that $\kappa_2(AP) = \kappa_2(SQ_A)$, where Q_A is an orthogonal basis for the

column space of A . Through the choice of embedding matrix and dimension, one can control the conditioning of SQ_A with high probability, as explained in Section 2.1.3. Most fundamentally, the conditioning of the preconditioned system $\kappa_2(AP)$ does not depend at all on the condition number of A .

Cartis et al. present a general framework for the sketch-and-precondition algorithm in [47]. This framework consists of the following steps: 1) drawing an embedding matrix and sketching A and b , 2) computing a (rank-revealing) decomposition of the sketch SA , 3) computing a sketch-and-solve initial guess based on the full-rank part of the decomposition, and 4) iteratively solving the least squares system with a preconditioner based on (the full-rank part of) the decomposition, with the initialization from step 3, and using LSQR. This framework allows for rank-deficient matrices and a range of decompositions, making it extremely general. The inclusion of the sketch-and-solve initialization (see Section 5.2.1 for details) is incredibly important for the performance of any sketch-and-precondition algorithm, as we will argue in the next chapter.

So far, we have not included sketch-and-solve initialization as a standard part of the sketch-and-precondition algorithm. Although it was introduced in Rokhlin and Tygert's original paper [222], Blendenpik [14] and LSRN [187] do not explicitly include it. In the remainder of this chapter, we consider sketch-and-precondition without initialization. We aim to demonstrate the inherent lack of stability in the algorithm without this step. In the next chapter, we argue (as does [47]) that it should be a standard part of the algorithm.

The framework and problems we consider here are also less general as compared to [47] in terms of applicability to rank-deficient problems. Both [47] and [187] allow the sketch SA to be of rank $r < n$ (numerically or exactly). In LSRN, the sketch is decomposed as $SA = U\Sigma V^T$, where $\Sigma \in \mathbb{R}^{r \times r}$, and the preconditioner is $P = V\Sigma^{-1}$. More generally, the Ski-LLS general framework allows for a decomposition [47]

$$SA = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix},$$

where $R_{11} \in \mathbb{R}^{r \times r}$ is non-singular and Q_1 , Q_2 , V_1 , and V_2 have orthonormal columns. In this chapter, we consider only sketches that are of full column rank, but given the convergence results in [47] and [187] conjecture that the results carry over. In Chapters 5 and 6, we allow for rank-deficiency.

Therefore, the specific sketch-and-precondition framework we consider in this chapter is as follows. Throughout, let $A = Q_A R_A$ denote a QR decomposition of $A \in \mathbb{R}^{m \times n}$.

1. Any embedding matrix $S \in \mathbb{R}^{s \times m}$ can be used to sketch A provided that $n \leq s < m$ and $\kappa_2(SQ_A) = \mathcal{O}(1)$.

2. Any decomposition of SA can be used to find a preconditioner $P \in \mathbb{R}^{n \times n}$, as long as $\kappa_2(SQ_A) = \kappa_2(AP)$ and P can be applied to a vector in $\mathcal{O}(n^2)$ time. We focus on the QR decomposition $SA = QR$ for $P = R^{-1}$ and the SVD $SA = U\Sigma V^T$ for $P = V\Sigma^{-1}$.
3. Any iterative solver that solves $\min \|APw - b\|$ can be used. We restrict ourselves to Krylov subspace solvers LSQR and LSMR.

In the next section, we experimentally investigate the numerical stability of these types of algorithms.

4.6 The numerical stability of sketch-and-precondition without initialization

We now demonstrate the numerical instabilities that occur with **sketch-and-precondition techniques without initialization**. We let S be an SRTT embedding with embedding dimension $s = 4n$, we use the QR decomposition to compute a preconditioner, and we employ LSQR as the iterative solver. LS problems are constructed at random in the same way as described in Section 4.2.5. Figure 4.4 compares the performance of unpreconditioned LSQR, sketch-and-precondition (SAP), and Householder QR in terms of the relative residual $\|Ax - b\|_2 / \|b\|_2$ (left column), the normwise backward error $\eta_F(x)$ (middle column, see (4.6)), and the (forward) error $\|x^* - x\| / \|x^*\|$ (right column).

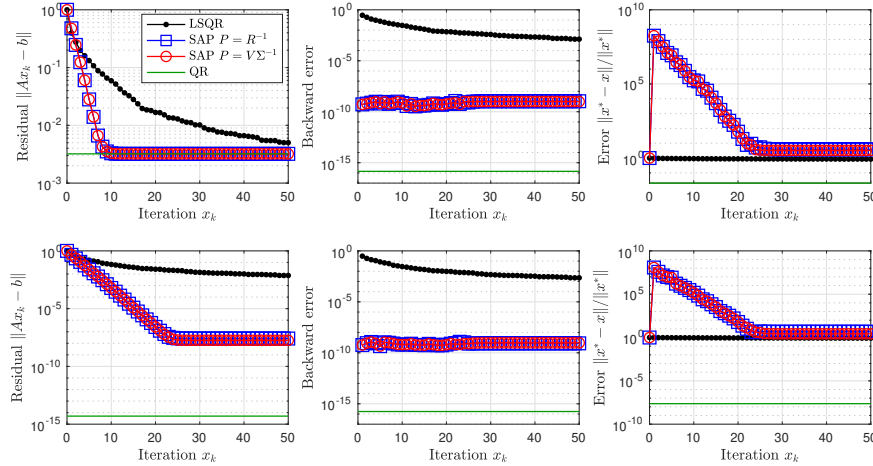


Figure 4.4: Unpreconditioned LSQR, sketch-and-precondition (SAP) with $P = R^{-1}$, sketch-and-precondition (SAP) with $P = V\Sigma^{-1}$, and the QR decomposition to solve two noisy and randomly generated LS problems. We display the residual, the backward error, and the forward error for each LSQR iteration. The experiments are performed with two noise levels: $\|Ax^* - b\|_2 = \|e\|_2 = 10^{-2}$ (top row) and $\|Ax^* - b\|_2 = \|e\|_2 = 10^{-12}$ (bottom row). The matrix A is 10000×200 with condition number $\kappa_2(A) = 10^{10}$.

Consider first the top row of Figure 4.4: an inconsistent, moderately ill-conditioned problem. We see that all algorithms attain the optimal residual. However, considering the backward error,

both sketch-and-precondition varieties fail to converge to a backward stable solution. The same behaviour can be observed for the forward error. In general we find that for inconsistent problems, i.e., problems with large optimal residual, sketch-and-precondition converges to a solution with good residual. However, for moderately ill-conditioned problems, these solutions are often not backward stable.

The bottom row of Figure 4.4 shows the more obvious instabilities of sketch-and-precondition: the minimal attainable residual is not optimal. Consider the final residuals: the sketch-and-precondition solution stagnates around 10^{-8} residual. This instability is also visible in the backward error and forward error, where we see sketch-and-precondition is not backward stable.

In all of the experiments, the tolerance `tol` in LSQR is set to be equal to machine precision, to allow us to investigate the maximal attainable accuracy. In each of these experiments, the iterates stagnate before the maximum number of iterations is reached.

4.6.1 Sensitivity to the condition number and consistency

Similar to Figure 4.2 and Figure 4.3, we investigate the sensitivity of sketch-and-precondition without initialization to problems with varying condition number and optimal residual. The results are shown in Figure 4.5. The left plot shows that the maximal attainable residual appears to be upper bounded by approximately $\kappa_2(A)u$. This is portrayed by the bottom-right corner of the figure (where $\|Ax^* - b\| > \kappa_2(A)u$) showing residuals close to optimal, and the top-left corner (where $\|Ax^* - b\| < \kappa_2(A)u$) showing a rounding-error effect. There appears to be no clear connection between the optimal residual and the size of the backward error of solutions, portrayed in the middle figure of Figure 4.5. This panel shows there is apparently very little influence from $\|Ax^* - b\|$ on the final backward error, only the conditioning of the problem affects the size of the backward error. The most difficult problem is moderately ill-conditioned.

4.6.2 Variants of sketch-and-precondition

We repeat these experiments for SVD-based randomized preconditioners and LSMR as the iterative solver. We create matrices using different singular vectors and a different singular value spectrum. The left and right singular vectors are given by the singular vectors of `well1033` from the SuiteSparse Matrix Collection ($m = 1033$, $n = 320$, $\text{nnz}(A) = 4732$), and create a polynomially decaying singular value spectrum setting $\sigma_i(A) = i^{-p}$, where $p = \log(\kappa_2(A))/\log(n)$. The backward errors are shown in Figure 4.6. We also note that the qualitative behaviour shown in Figures 4.4, 4.5, and 4.6 is unaffected by different types of embedding matrices or embedding dimensions. These choices influence $\kappa_2(AP)$ and $\kappa_2(P)$ slightly, which can somewhat affect the convergence behaviour in terms

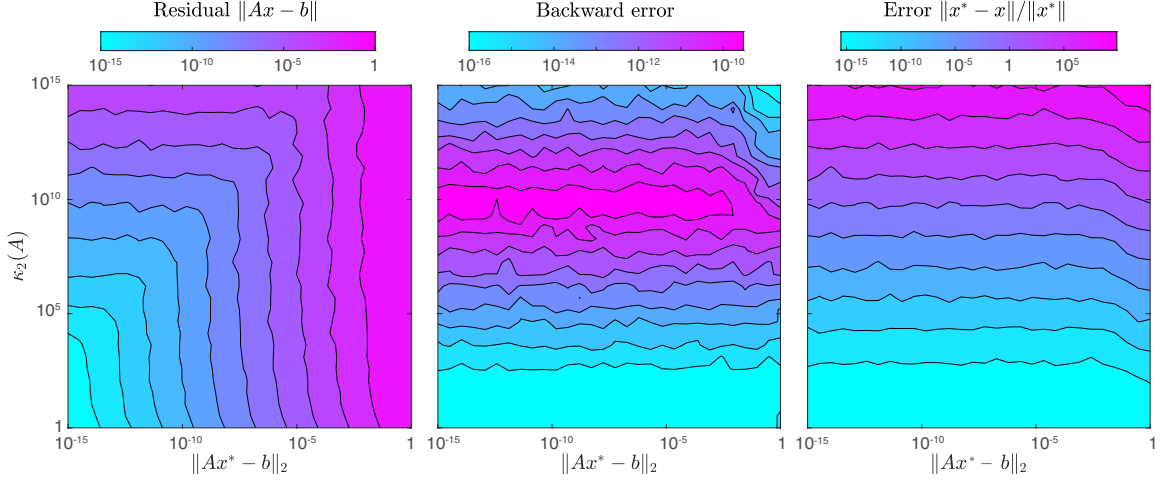


Figure 4.5: The sketch-and-precondition algorithm (without initialization) with LSQR and preconditioner $P = R^{-1}$ applied to $A \in \mathbb{R}^{2000 \times 100}$ with varying condition number and consistency. We investigate the minimal attainable residual (left), the final backward error (middle), and the error (right).

of speed and maximal attainable accuracy. However, we do not observe qualitative differences provided that $\kappa_2(AP) = \mathcal{O}(1)$.

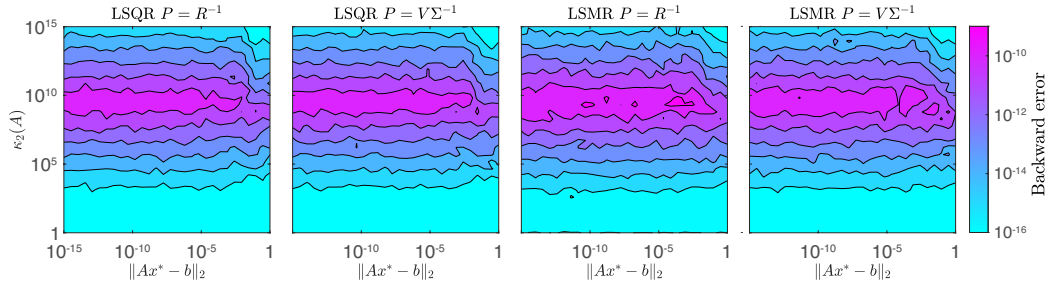


Figure 4.6: The backward error of all considered (non-initialized) sketch-and precondition variants for problems with varying consistency and optimal residual. The matrices all have the same singular vectors, taken from `well11033` from the SuiteSparse Matrix Collection [74]. The singular values are polynomially decaying. No matter whether one uses LSQR or LSMR, or a QR-based or SVD-based randomized preconditioner, sketch-and-precondition is not always backward stable.

4.7 The numerical stability of preconditioned LSQR

The instability of sketch-and-precondition in various forms begs the question: is the problem the ‘sketch’-part or the ‘precondition’-part? We perform one more experiment to demonstrate that (at least in some cases) it is the way that the preconditioner is employed, and not how it is formed. We take the transpose of the `Maragal_8` matrix from the SuiteSparse Matrix Collection [74] and regularize it to ensure its full-rankedness. The final matrix $A \in \mathbb{R}^{m \times n}$ ($m = 108289$, $n = 33212$, $\text{nnz}(A) = 1341627$) is the transpose of `Maragal_8`, appended with $\sqrt{\lambda}I_{33212}$ where $\sqrt{\lambda} = 10^{-11}$. The condition number of A is approximately $\kappa_2(A) \approx 10^{12}$. We generate a right-hand side b by forming a normalized random solution x drawn from $\mathcal{N}(0, I_n)$ and letting $b = Ax + 10^{-11}e$, where $e \in \mathbb{R}^m$ is

normalized random noise.

We solve the LS problem with unpreconditioned LSQR, sketch-and-precondition without initialization, the QR decomposition – which all behave as expected – and with preconditioned LSQR with a different preconditioner. This preconditioner is obtained by computing the Householder QR factorization A . In particular, let $A = Q_A R_A$ be QR decomposition, then $P = R_A^{-1}$. This approach is not practical because it requires the full factorization of A , but is done to demonstrate that the quality of the preconditioners is not at fault. The result is shown in Figure 4.7. Even with the ideal preconditioner for which $(\kappa_2(AR_A^{-1}) = 1$, preconditioned LSQR stagnates before an optimal residual is attained.

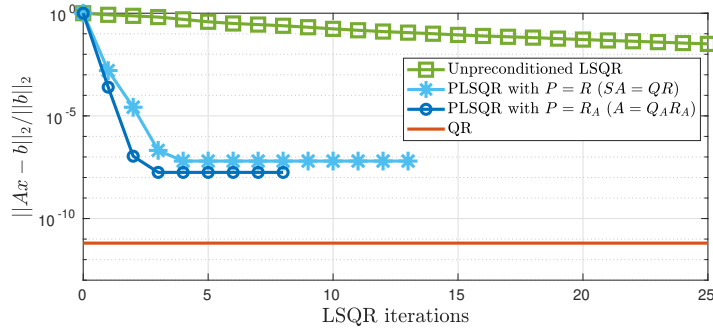


Figure 4.7: The residual attained by unpreconditioned LSQR, sketch-and-precondition, preconditioned LSQR (PLSQR) with the exact R factor of the QR decomposition of A , and the QR method for LS problems. In exact arithmetic $\kappa_2(AR_A^{-1}) = 1$, so LSQR should converge in 1 iteration. We still see rapid convergence at the start, but then stagnation occurs around a residual $\|Ax - b\| \approx 10^{-8}$, whereas QR attains a residual $\|Ax - b\| \approx 10^{-11}$. The problem is **Maragal_8** as a regularized overdetermined LS problem.

While we have not determined the precise source of instability in the standard preconditioned LSQR routines, we believe that it lies in the fact that each application of the preconditioner P – which involves solving an ill-conditioned linear system $Rv = w$ – incurs a relative error proportional to $u\kappa_2(R)$. Such errors are present every time we apply P or P^T . Moreover, each application behaves somewhat differently, in that in the i th iterate we have $(R + \Delta R_i)\hat{v} = w + \delta b w_i$, where $\Delta R_i, \delta b_i$ are small but different for each i . In other words, one can view the preconditioner as having a $u\kappa_2(R)$ nonlinear effect. The fact that the convergence of iterative methods can be impaired by nonlinear preconditioners has been observed in [271].

4.8 Summary and outlook

We have discussed perturbation theory for least squares problems, different algorithms for LS problems, and their behaviour in finite precision. Most notably, we showed that the sketch-and-precondition framework without initialization suffers from numerical instabilities. These instabilities significantly limit the minimal attainable residual and backward error, especially for moderately to severely ill-conditioned problems. The residual of iterates seem to be lower bounded by a minimal

attain accuracy proportional to $\|Ax - b\| \lesssim \kappa_2(A)u$. This means that particularly for consistent problems, iterations stagnate potentially long before the optimal residual is reached. In terms of the backward error, the loss of accuracy is largest for problems with $\kappa_2(A) \approx 10^{10}$ in double precision. We observe the same qualitative behaviour regardless of specific choices for sketching matrix, sketching dimension, decomposition, and Krylov subspace solver.

In fact, we observe similar instabilities in preconditioned LSQR with a deterministic preconditioner; in particular R_A from the QR decomposition $A = Q_A R_A$. Figure 4.7 shows that even with this ideal preconditioner, which clusters all singular values at 1 in exact arithmetic, cannot converge to a good and stable solution.

We discussed how LSQR, LSMR, and their preconditioned variants are implemented in finite-precision, but can only conjecture towards the source of the instability. The finite-precision behaviour of any (short-recurrence) iterative Krylov subspace solver is difficult to grasp despite various seminal works [117, 120, 188, 203, 239]. The finite-precision behaviours of preconditioned Krylov subspace solvers appear to be completely uncharted territory in the literature. One reason for this might be the wide range of available preconditioners of varying quality and varying finite-precision behaviour. However, the excellent theoretical properties of randomized preconditioners ensuring $\kappa_2(AP) = \mathcal{O}(1)$ combined with thorough understanding of, for instance, a triangular solve in finite precision, can offer a good environment to pursue better understanding of this class of algorithms.

Randomized preconditioners were revolutionary when they were introduced because they allowed practitioners to precondition dense (highly-overdetermined) linear systems, for which there previously existed no viable method. The low computational cost and excellent theoretical convergence properties accompanying the sketch-and-precondition framework make it a popular choice with immense potential. It is therefore extremely worthwhile to investigate strategies that improve the numerical stability of the framework.

One solution has already been mentioned throughout this chapter: the use of sketch-and-solve initialization. In the next chapter, we show that this indeed greatly improves the convergence properties of the sketch-and-precondition framework and should be a standard element of this algorithm (as is also argued in [47]). However, initialization does not resolve all instabilities. We will show that for certain problems, even sketch-and-precondition with sketch-and-solve initialization cannot obtain backward stable solutions. Resolving this further is the focus of the next chapter.

Chapter 5

Stabilizing the sketch-and-precondition framework

5.1 Introduction

Sketch-and-precondition algorithms have been discussed in detail in the previous chapter. It is a class of randomized algorithms for solving overdetermined least squares (LS) problems of the form

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2, \quad A \in \mathbb{R}^{m \times n}, \quad b \in \mathbb{R}^m, \quad (5.1)$$

where $m \gg n$. In Chapter 4, we showed that sketch-and-precondition techniques without initialization are numerically unstable. For moderately ill-conditioned problems ($1 \ll \kappa_2(A) \ll u^{-1}$, where $\kappa_2(A) = \sigma_{\max}(A)/\sigma_{\min}(A)$ is the condition number of A and u is the unit round-off), sketch-and-precondition iterations stagnate in terms of residual and backward error, potentially before optimal levels are reached (see Figure 4.4). The main focus of this chapter is deriving a provably stable method to resolve these numerical instabilities. We suggest a modification to the sketch-and-precondition framework to obtain a new algorithm, which we coin sketch-and-apply.

The algorithm is based on directly *applying* the randomized preconditioner to the data matrix, as opposed to applying it to vectors within an iterative solver. Essentially, having computed a preconditioner P (usually $P = R^{-1}$ for $SA = QR$), we explicitly compute $Y = AP$ and solve the well-conditioned system

$$z^* = \operatorname{argmin}_{z \in \mathbb{R}^n} \|Yz - b\|_2$$

using unpreconditioned LSQR. The solution to (5.1) is then found by the change of coordinates $x^* = Pz^*$.

We show using classical stability analysis as well as experimentally that sketch-and-apply attains backward stable solutions under modest conditions, making it one of the first randomized algorithms shown to be backward stable in a classical sense. The caveat on the backward stability is that it relies on the backward stability of unpreconditioned LSQR on a very well-conditioned matrix, which we discussed in the previous chapter. However, one of the main downsides of sketch-and-apply is that its computational complexity is $\mathcal{O}(mn^2)$ which is on par with backward stable direct solvers.

A much more computationally inexpensive, yet partial fix, was readily available in the literature. As discussed in the previous chapter, various authors include sketch-and-solve initialization as a standard part of the sketch-and-precondition framework. It was originally suggested in the seminal paper by Rokhlin and Tygert [222], as well as included in the general framework proposed in [47]. We empirically show that this significantly improves the performance of the framework on the different metrics discussed in the previous chapter. However, we must also conclude that it is still not entirely backward stable.

In the next section, we introduce these two different approaches to stabilizing the sketch-and-precondition framework. The focus of the remainder of the chapter is proving the backward stability of sketch-and-apply. In Section 5.3, we introduce some background material on numerical stability analysis and smoothed analysis. In Section 5.4, we consider sketch-and-apply (see Algorithm 10) in finite precision. In Section 5.5, we consider extremely ill-conditioned LS problems and look at the numerical stability of smoothed sketch-and-apply algorithms (see Algorithm 11). We introduce a master algorithm and display numerical experiments in Section 5.6. Finally, in Section 5.7, we conclude by noting the practical significance of the sketch-and-solve initialization.

Throughout the chapter, we drop the assumption that $A \in \mathbb{R}^{m \times n}$ is of full column rank. We deal with rank-deficiency in a different way than in [47] and [187] (see Section 4.5.1). Instead of truncating the decomposition of SA , we smoothe A before sketching so we ensure $S\tilde{A}$ is of full column rank. This is discussed in more detail in Sections 5.2.3 and 5.5. We focus on overdetermined LS problems in this chapter, but we expect much of the stability results to carry over to solving underdetermined linear systems using sketching, as is done in LSRN [187].

5.2 Stabilizing approaches to the sketch-and precondition framework

We discuss two different attempts at stabilizing the sketch-and-precondition framework. The first is to include an excellent initial guess in the iterative solver, which is not an original idea. The particular initialization was suggested in Rokhlin and Tygert (2008), but not explicitly included in Blendenpik [14] and LSRN [187]. The very general sketch-and-precondition framework and implementation Ski-LLS presented in [47] does include the initialization as standard practice. Cartis et al. (2021) also provide convergence guarantees (in exact arithmetic) that assume initialization. Through

experimental evidence in this section, we note that these improvements in convergence behaviour are even more notable in finite-precision. Although this approach greatly improves the numerical properties of sketch-and-precondition, it is still not always numerically stable. After having introduced the initialization approach and having shown its deficiencies, we focus on sketch-and-apply.

5.2.1 Sketch-and-precondition with sketch-and-solve initialization

The sketch-and-precondition framework as presented in Blendenpik [14] and LSRN [187] (see Algorithm 2) has $x_0 = 0$ as the initial guess in LSQR. However, as suggested in Rokhlin and Tygert’s paper [222] and amplified in [47], a more natural guess is readily available; the solution to the sketched LS problem, i.e.,

$$x_0 = \operatorname{argmin}_{x \in \mathbb{R}^n} \|SAx - Sb\|_2. \quad (5.2)$$

This can be computed directly with the QR decomposition of SA ; the resulting algorithm is displayed in pseudocode in Algorithm 9. Although originally proposed in [222] as part of the sketch-and-precondition framework, some notable implementations (e.g. [14, 187]) do not mention this choice of initial guess as part of their algorithms.¹ Notable, Ski-LLS does include the sketch-and-solve initialization as standard [47].

Algorithm 9 A sketch-and-precondition LS solver for (5.1) with a sketch-and-solve solution as initial guess. Here, HHQR refers to the Householder QR algorithm.

- 1: Draw a random sketching matrix $S \in \mathbb{R}^{s \times m}$, where $m \gg s > n$
 - 2: Compute $B = SA$ and $c = Sb$
 - 3: Compute both Q and R of the QR factorization of B using HHQR
 - 4: Compute initial guess $x_0 = R^{-1}Q^T c$
 - 5: Solve $\min \|Ax - b\|$ with LSQR using right preconditioner $P = R^{-1}$ and initial guess x_0
-

The performance of Algorithm 9 is displayed in Figure 5.1. We test the algorithm on the same set of problems as in the previous chapter, where the condition number $\kappa_2(A)$ and the optimal residual $\|Ax^* - b\|$ are varies. See Section 4.2.5 for details on how these problems are constructed.

The sketch-and-solve solution typically attains an accuracy within a small multiple of the optimal accuracy (see Section 2.3.1). Although the attainable accuracy of sketch-and-precondition algorithms with a zero initial guess will stagnate before a desired accuracy when dealing with ill-conditioned problems (see Figure 4.4(b)), our experiments show Algorithm 9 attains optimal residuals in all cases, and is therefore significantly better than the standard, trivial initial guess $x_0 = 0$. This can be observed in the left panel of Figure 5.1. Furthermore, as the QR decomposition (or SVD) of the sketch SA is necessary for any sketch-and-precondition solver, this initial guess is obtained

¹In the C implementation of Blendenpik [14], sketch-and-solve initialization is available as an option, but the user needs to append ‘improve_start_point’ to the parameters.

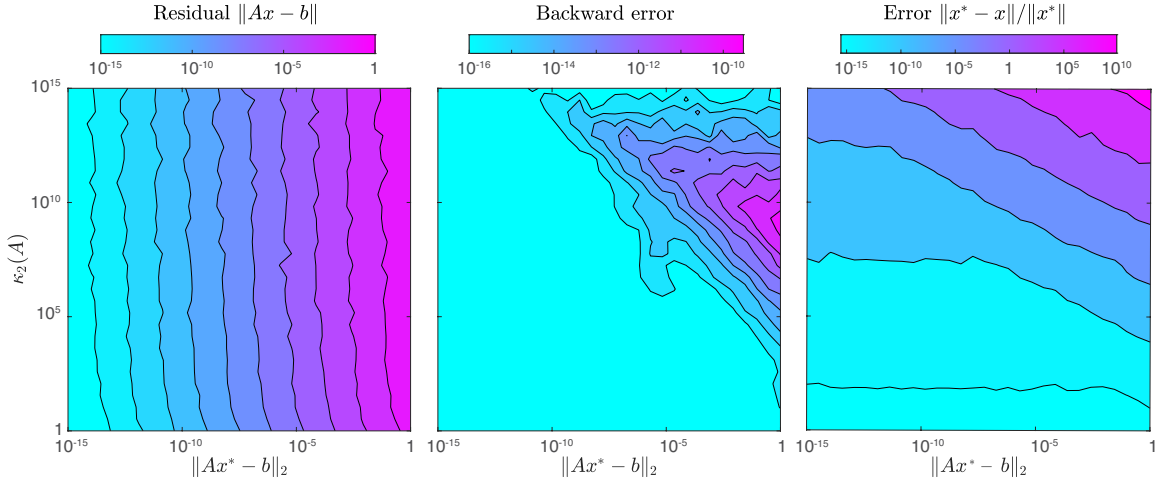


Figure 5.1: The sketch-and-precondition algorithm with sketch-and-solve initialization attains optimal residuals in all considered problems, but is not backward stable for moderately ill-conditioned problems with large optimal residual. Note that the algorithm does appear to be forward stable, in the sense that the (forward) errors are of the same order as for the stable QR algorithm in Figure 4.2.

practically for free. We join Cartis et al. [47] in urging practitioners to adopt this as standard practice for these types of algorithms.

It should be noted, however, that there are instances where the solution found by sketch-and-precondition with sketch-and-solve initialization does not attain backward stable solutions. This is demonstrated in the middle panel of Figure 5.1. In particular, problems where the optimal residual is large (inconsistent) and the matrix A is moderately ill-conditioned cannot be solved in a stable way. There is a clear intuition to describe this phenomenon. The sketch-and-solve solution has a residual that is close to optimal. For a problem with, say, $\|Ax^* - b\| = 10^{-10}$ or smaller, this means the solution is already very good. It resolves directions corresponding to quite small singular values. On the other hand, for an optimal residual of one or two digits, only a few singular vector directions in the solution need to be accurate. As for the conditioning of A : if A is extremely ill-conditioned, then in some sense many solutions are backward stable as small perturbations to the smallest singular vector directions can drastically change the computed solution, while maintaining backward stability.

To further resolve the numerical instabilities, we introduce the *sketch-and-apply* algorithm.

5.2.2 Sketch-and-apply

The sketch-and-apply algorithm is a modification of the sketch-and-precondition algorithm for which we can ensure that the computed residual is close to optimal and the backward error is approximately machine precision regardless of the initial guess. This is demonstrated experimentally in Figure 5.2. The top row corresponds to sketch-and-apply with trivial initialization, and the bottom row to sketch-and-apply with sketch-and-solve initialization. Both variants are backward stable for all the

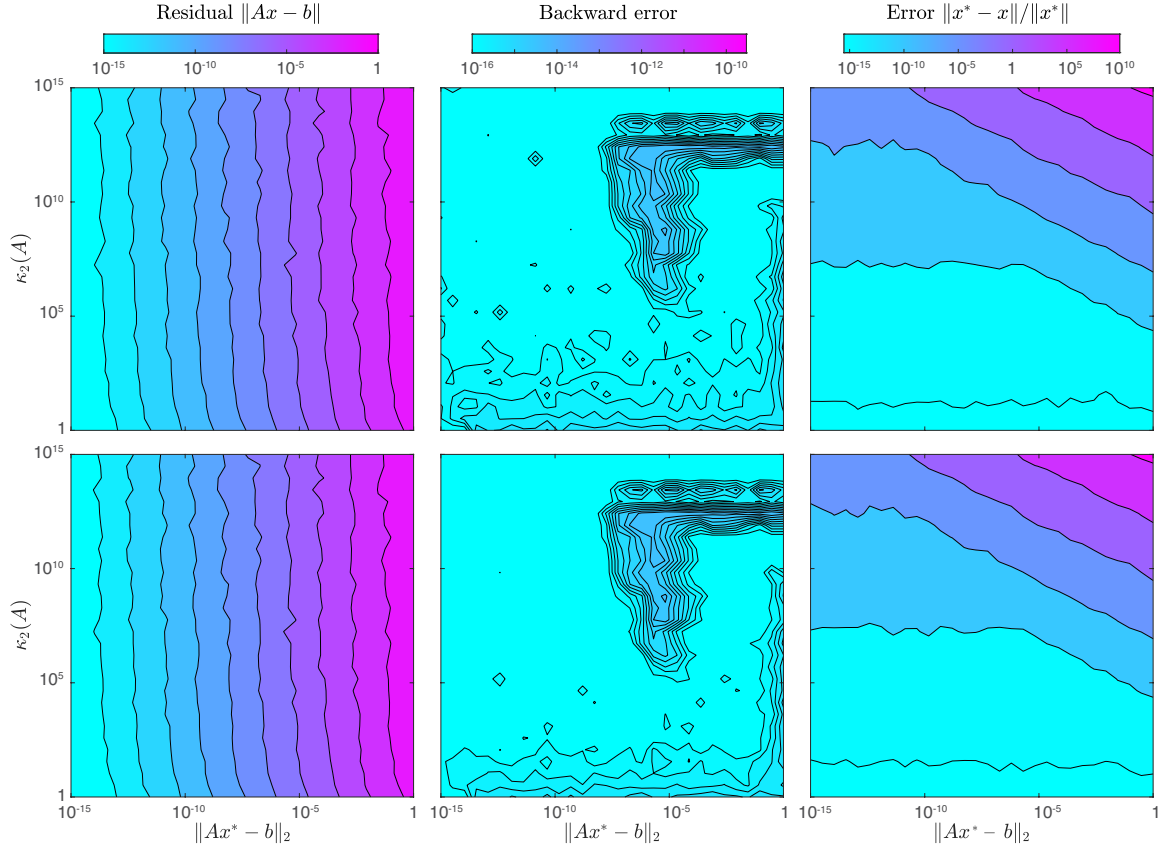


Figure 5.2: Top row: sketch-and-apply without initialization, bottom row: sketch-and-apply with sketch-and-solve initialization (see Algorithm 10). The sketch-and-apply algorithm attains optimal residuals and is backward stable for all considered problems. The error is of similar levels as the error in the QR algorithm.

considered problems. However, we must note that this approach to initialization comes at significant cost. The complexity of sketch-and-apply is $\mathcal{O}(mn^2)$, which bring it on equal footing with direct solvers as QR. This contribution is primarily of a theoretical nature.

The modification to sketch-and-precondition is simple. Instead of using $P = R^{-1}$ as a preconditioner, we explicitly apply P by computing AP and then employ an unpreconditioned iterative LS solver on $(AP)z = b$ with $x = Pz$. We therefore call this a *sketch-and-apply* technique. Of course, in exact arithmetic sketch-and-precondition and sketch-and-apply compute the same solution; however, for ill-conditioned LS problems in floating-point arithmetic, we find a significant difference.

By computing AP explicitly, we remove all ill-conditioning from the iterative solver and instead use an unpreconditioned solver on a well-conditioned system. This results in accurate and backward stable solutions. In particular, we prove that if $\kappa_2(A) \ll u^{-1}$, where u is machine epsilon, our sketch-and-apply technique computes a backward stable solution provided that LSQR on a well-conditioned matrix computes a backward stable solution (see Theorem 5.6).

Unfortunately, while sketch-and-precondition techniques cost $\mathcal{O}(mn \log m)$ operations, sketch-and-apply costs $\mathcal{O}(mn^2)$ operations as AP must be computed. Therefore, sketch-and-apply tech-

Algorithm 10 A sketch-and-apply LS solver for (5.1). Here, HHQR refers to the Householder QR algorithm. One can include a sketch-and-solve initial guess by computing the QR decomposition $QR = SA$ and $z_0 = Q^T Sb$, and setting z_0 as an initial guess in step 5.

- 1: Draw a random sketching matrix $S \in \mathbb{R}^{s \times m}$, where $m \gg s > n$
 - 2: Compute $B = SA$
 - 3: Compute the triangular factor R of a QR factorization of B using HHQR
 - 4: Compute $Y = AR^{-1}$ with forward substitution
 - 5: Solve $\min \|Yz - b\|$ with LSQR and no preconditioner
 - 6: Compute $x = R^{-1}z$ with back substitution
-

niques have the same computational complexity as the classical QR-based LS solver. We note, however, that LSRN, another popular sketch-and-precondition algorithm, also uses $\mathcal{O}(mn^2)$ operations but is still competitive as the expensive computation is in matrix multiplication, which is highly parallelizable. The same applies to sketch-and-apply.

5.2.3 Smoothed sketch-and-apply

When $\kappa_2(A) \gtrsim u^{-1}$, our analysis does not guarantee that Algorithm 10 leads to an accurate solution. However, in practice, we often get accurate final LS residuals. To explain why this is the case, we model floating-point rounding errors using smoothed analysis. That is, we consider “smoothing” A to $A + \sigma G/\sqrt{m}$, where the entries of G are independent and identically distributed (i.i.d.) standard Gaussian random variables and σ is a scaling factor that we select as $\sigma = 10\|A\|_2 u$. The idea is that $A + \sigma G/\sqrt{m}$ is significantly better conditioned than A itself, assuming that A is extremely ill-conditioned. In fact, for sufficiently small σ , one can show that $\kappa_2(A + \sigma G/\sqrt{m}) \lesssim 1/\sigma$ with high probability [41] (see Corollary 5.1). The additive perturbation of $\sigma G/\sqrt{m}$ to A ensures that sketch-and-apply techniques can also compute solutions with good backward error, even when A is extremely ill-conditioned (see Algorithm 11). For extremely ill-conditioned LS problems, one could explicitly add an additive random perturbation or hope that floating point rounding errors deliver the same effect, as it often does in practice.

Algorithm 11 A smoothed sketch-and-apply LS solver for (5.1) when $\kappa_2(A) \gtrsim u^{-1}$.

- 1: Draw a random standard Gaussian matrix $G \in \mathbb{R}^{m \times n}$ with i.i.d. entries
 - 2: Compute $\tilde{A} = A + \sigma G/\sqrt{m}$ for $\sigma = 10\|A\|_2 u$
 - 3: Perform Algorithm 10 on $\tilde{A}$
-

The remainder of this chapter focuses on analyzing the sketch-and-apply algorithm. The numerical stability of sketch-and-apply (without smoothing) is considered in Section 5.4, and the stability of its smoothed variant in Section 5.5.

5.3 Finite-precision preliminaries

We now introduce some background material for the numerical stability analysis (see Section 5.3.1) and smoothed analysis (see Section 5.3.2). Background material on sketch-and-precondition techniques and sketching matrices can be found in Sections 2.1.3 and 2.3.2.

5.3.1 Numerical stability analysis

Basic arithmetic operations (e.g., $+$, $-$, $\cdot$, and $/$) on a computer are performed with rounding errors due to the finite precision of numbers in floating-point representation. The stability of a numerical algorithm refers to the property to compute solutions with a small backward error [140] in the presence of rounding errors. We are particularly interested in proving that sketch-and-apply computes backward stable solutions. The backward error for LS problems is defined in (4.6). A backward stable algorithm computes solutions with a backward error of the same order as the precision.

We assume that computations are performed with a precision of $u \ll 1$ and that numbers and arithmetic operations are exact up to this precision. We now consider three example algorithms: (1) The standard algorithm for matrix-matrix multiplication, (2) Householder QR, and (3) Triangular solve with substitution. We need these results to understand sketch-and-apply LS solvers.

5.3.1.1 Matrix-matrix multiplication

Consider matrix-matrix multiplication of two matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$. It is shown that the standard algorithm for computing AB satisfies [140, Chapt. 3]

$$|AB - \text{fl}(AB)| \leq \frac{nu}{1 - nu} |A| |B| = \gamma_n |A| |B|, \quad \gamma_n = \frac{nu}{1 - nu}, \quad (5.3)$$

where $\text{fl}(AB)$ means that AB is computed with precision u and $|A|$ denotes the entry-wise absolute value of A . The inequality in (5.3) is understood as holding for each entry. It can be shown that (5.3) leads to the bound

$$\|AB - \text{fl}(AB)\|_2 \leq \gamma_n \min(\sqrt{m}, \sqrt{n}) \min(\sqrt{n}, \sqrt{p}) \|A\|_2 \|B\|_2, \quad (5.4)$$

where $\|\cdot\|_2$ is the spectral norm. The inequality in (5.4) is a forward error in the sense that it shows that $\text{fl}(AB)$ is close to AB .

5.3.1.2 Householder QR factorization

Next, we consider the Householder QR of a matrix $A \in \mathbb{R}^{m \times n}$, where $m \geq n$. We denote the triangular factor computed by the Householder QR algorithm as $\hat{R} \in \mathbb{R}^{n \times n}$, and we are interested in the accuracy of $\hat{R}$. By [140, Thm. 19.4], there exists a matrix $Q \in \mathbb{R}^{m \times n}$ with orthonormal columns such that

$$A + E = Q\hat{R}, \quad \|e_j\|_2 \leq \tilde{\gamma}_{mn}\|a_j\|_2, \quad 1 \leq j \leq n,$$

where e_j and a_j denote the j th columns of E and A respectively, and $\tilde{\gamma}_{mn} = cmnu/(1 - cmnu)$ for a small integer constant c . A direct consequence is that

$$\|E\|_F \leq \tilde{\gamma}_{mn}\|A\|_F, \quad (5.5)$$

where $\|\cdot\|_F$ is the matrix Frobenius norm. The inequality in (5.5) has the interpretation that the QR factorization computed by Householder QR is exact for a slightly perturbed matrix A .

5.3.1.3 Triangular system solving with substitution

Finally, we consider the error arising from solving a triangular system with substitution. Let $R \in \mathbb{R}^{n \times n}$ be a nonsingular upper-triangular matrix. It is known that the computed solution, $\hat{x}$, of the linear system $R^T x = b$ satisfies [140, Sec. 8.2]

$$\|x - \hat{x}\|_2 \leq \frac{\sqrt{n}\gamma_n\kappa_2(R)}{1 - \sqrt{n}\gamma_n\kappa_2(R)}\|x\|_2, \quad (5.6)$$

where γ_n is defined in (5.3). The inequality in (5.6) tells us that we expect the relative forward error of the computed solution to be on the order of $\kappa_2(R)$.

When we do any sketch-and-apply technique, we need to compute AP , where $P = R^{-1}$. We compute AP by solving $R^T x_i = a_i$ for $1 \leq i \leq m$, where a_i^T denotes the i th row of A and x_i^T is the i th row of AP . If we denote the computed solution to $R^T x_i = a_i$ by $\hat{x}_i$, then from (5.6) we find that

$$\sum_{i=1}^m \|x_i - \hat{x}_i\|_2^2 \leq \left[\frac{\sqrt{n}\gamma_n\kappa_2(R)}{1 - \sqrt{n}\gamma_n\kappa_2(R)} \right]^2 \sum_{i=1}^m \|x_i\|_2^2.$$

We conclude that for $P = R^{-1}$ we have

$$\|AP - \widehat{AP}\|_F \leq \frac{\sqrt{n}\gamma_n\kappa_2(P)}{1 - \sqrt{n}\gamma_n\kappa_2(P)}\|AP\|_F, \quad (5.7)$$

where $\widehat{AP}$ denotes the computed matrix-matrix product. Roughly speaking, $\widehat{AP}$ is close to AP when $\kappa_2(P)$ is modest. We additionally bound the backward error resulting from solving $Rx = b$

with back substitution. The computed solution $\hat{x}$ satisfies [140, Theorem 8.5]

$$(\hat{R} + \Delta\hat{R})\hat{x} = b, \quad \|\Delta\hat{R}\|_2 \leq \gamma_n \sqrt{n} \|\hat{R}\|_2. \quad (5.8)$$

5.3.2 Smoothed analysis of condition numbers

Due to rounding errors, most matrices with $\kappa_2(A) > u^{-1}$ are perturbed to a matrix with condition number less than u^{-1} once represented on a computer. To explain this, we can model the rounding errors by an additive Gaussian perturbation to A . This type of technique fits into the field of smoothed analysis and has been studied in [141]. More precisely, we suppose that $A \in \mathbb{R}^{m \times n}$ and that we would like to bound the condition number of $A + \sigma G$, where σ is a small number and G is a standard Gaussian matrix with i.i.d. entries. We have the following statement, which is a corollary of [41, Thm. 1.1].

Corollary 5.1. *Let $A \in \mathbb{R}^{m \times n}$ with $m \geq 3n$ and $r \geq 10^4$. If A is perturbed to $\hat{A} = A + \sigma G / \sqrt{m}$ with $\sigma = 8.25 \|A\|_2 / r$, where G is standard Gaussian matrix with i.i.d. entries, then*

$$\mathbb{P}[\kappa_2(\hat{A}) \geq r] < 2^{n-m}.$$

Proof. The result immediately follows by setting $z = (r(1 - \lambda))/e$ and $\sigma = 8.25/(r\sqrt{m})$ in Theorem 1.1 of [41]. This takes into account that Theorem 1.1 of [41] considers the case when $m < n$. The extra $\|A\|_2$ factor of σ in the statement of the corollary ensures that our final result continues to hold when $\|A\|_2 \neq 1$. $\square$

Corollary 5.1 shows us that a small additive Gaussian perturbation to a matrix A ensures that the condition number is small with high probability. It partially explains why matrices represented in floating-point arithmetic rarely have a condition number $> u^{-1}$. In Section 5.5, we use Corollary 5.1 to explain why sketch-and-apply techniques continue to deliver accurate LS solutions, even when we have $\kappa_2(A) > u^{-1}$.

5.4 On the numerical stability of sketch-and-apply without smoothing

We now consider the sketch-and-apply algorithm presented in Algorithm 10 and analyse its behaviour in finite precision arithmetic. We split this up in two parts: we first look at the condition number of the preconditioned system, and secondly we look at the algorithm as a whole. We will find a necessary assumption for convergence is that $\kappa_2(A) \ll u^{-1}$, which corresponds to (numerical)

full-rankedness of A . In Section 5.5, we drop this assumption and investigate the smoothed sketch-and-apply algorithm in finite precision.

We first consider the conditioned of the preconditioned matrix AP in finite precision (without smoothing). By the properties of sketching matrices and Lemma 2.5, we know that if one computes P and AP in exact arithmetic, then $\kappa_2(AP)$ is bounded by a constant independent of $\kappa_2(A)$ with high probability. In finite precision, one typically needs to be more careful. In this section, we show that when $\kappa_2(A) \ll u^{-1}$, the condition number of the computed matrix AP can also be bounded by a constant with high probability (see Theorem 5.5). Unfortunately, such a bound does not extend to the case when $\kappa_2(A) > u^{-1}$ (see Section 5.5). In Section 5.4.4 we use the result on the condition number of AP to analyse the complete sketch-and-apply algorithm.

There are three main steps to compute AP . In finite precision, the computed matrices satisfy the following:

1. $\hat{B} = SA + E_1$,
2. $\tilde{Q}\hat{R} = \hat{B} + E_2$,
3. $\hat{Y} = A\hat{R}^{-1} + \Delta Y$,

where hats denote that we are accounting for rounding errors. The matrix-matrix multiplication error E_1 can be bounded using (5.4). We have a bound on the error arising from the Householder QR algorithm E_2 in (5.5). Note that $\tilde{Q}$ has orthonormal columns but is not equal to Q , where Q is the orthonormal factor of the exact QR factorization of $B = SA$. Finally, the error term ΔY in the step resulting from solving the triangular matrix system can be bounded using (5.7). After all these errors, we want to derive a bound on $\kappa_2(\hat{Y})$, as $\hat{Y}$ is the computed preconditioned matrix in finite precision.

5.4.1 Bounding $\kappa_2(\hat{Y})$ in terms of numerical errors

First, let $E = E_1 + E_2$ be the sum of the additive errors from the matrix-matrix multiplication and the QR factorization, such that $\tilde{Q}\hat{R} = QR + E$. We show in Lemma 5.2 that the quantity $\varepsilon_1 := \|E\|_2 \|R^{-1}\|_2$ controls the numerical error in the first two steps.

Lemma 5.2. *Let $A = Q_A R_A$ and $SA = QR$ be economized QR factorizations of A and $B = SA$, respectively. Let $\hat{B}$ be the computed SA and $\hat{R}$ be the computed upper triangular factor from the Householder QR algorithm of $\hat{B}$, both computed in finite precision. Let $\tilde{Q}$ be a matrix with orthonormal columns such that $E = \tilde{Q}\hat{R} - QR$ is the numerical error associated with $\hat{R}$. Define $\varepsilon_1 := \|E\|_2 \|R^{-1}\|_2$. Assuming that $\varepsilon_1 < 1$, we find that $\hat{R}$ is invertible and $A\hat{R}^{-1}$ is full rank. Furthermore,*

$$\kappa_2(\hat{R}) \leq \frac{\kappa_2(R) + \varepsilon_1}{1 - \varepsilon_1} \leq \frac{\kappa_2(SQ_A)\kappa_2(A) + \varepsilon_1}{1 - \varepsilon_1},$$

and

$$\kappa_2(A\hat{R}^{-1}) \leq \kappa_2(SQ_A) \frac{1 + \varepsilon_1}{1 - \varepsilon_1}.$$

Proof. We have

$$\|\hat{R}\|_2 = \|\tilde{Q}\hat{R}\|_2 = \|QR + E\|_2 \leq \|R\|_2 + \|E\|_2, \quad (5.9)$$

and

$$\sigma_{\min}(\hat{R}) = \sigma_{\min}(QR + E) \geq \sigma_{\min}(R) - \|E\|_2. \quad (5.10)$$

We see that $\sigma_{\min}(\hat{R}) > 0$ provided that $\varepsilon_1 < 1$. We can combine (5.9) and (5.10) together to obtain an upper bound on $\kappa_2(\hat{R})$. To bound $\kappa_2(A\hat{R}^{-1})$, we use the fact that² $A\hat{R}^{-1}\tilde{Q}^T = AR^{-1}(Q + ER^{-1})^\dagger$ to find that for any $v \in \mathbb{R}^m$, we have

$$\|v^T AR^{-1}\|_2(1 + \varepsilon_1)^{-1} \leq \|v^T A\hat{R}^{-1}\|_2 \leq \|v^T AR^{-1}\|_2(1 - \varepsilon_1)^{-1}. \quad (5.11)$$

The bound follows as $\kappa_2(AR^{-1}) = \kappa_2(SQ_A)$ (see Lemma 2.5). $\square$

Lemma 5.2 shows us, assuming ε_1 is sufficiently small, that $\kappa_2(\hat{R})$ cannot be larger than about $\kappa_2(SQ_A)\kappa_2(A)$, and $\kappa_2(A\hat{R}^{-1})$ cannot be much larger than $\kappa_2(SQ_A)$.

As the following lemma shows, the error incurred by solving the triangular system depends on the quantities in Lemma 5.2.

Lemma 5.3. *Let $\hat{R}^{-1}$ be the upper triangular matrix defined in Lemma 5.2 and let $\hat{Y}$ be the matrix obtained by computing $A\hat{R}^{-1}$ with forward substitution in finite precision. Define $\Delta Y = \hat{Y} - A\hat{R}^{-1}$ and $\varepsilon_2 = \|\Delta Y\|_2 \|(A\hat{R}^{-1})^\dagger\|_2$. Assuming that $\varepsilon_2 < 1$, we find that $\hat{Y}$ is full rank and*

$$\kappa_2(\hat{Y}) \leq \frac{\kappa_2(A\hat{R}^{-1}) + \varepsilon_2}{1 - \varepsilon_2}.$$

Proof. This follows immediately from the relationship $\hat{Y} = A\hat{R}^{-1} + \Delta Y$ and similar reasoning to the proof of Lemma 5.2. $\square$

In exact arithmetic, clearly $\kappa_2(Y) = \kappa_2(AR^{-1})$. The numerical error incurred due to the triangular solve is controlled by $\|\Delta Y\|$, which depends on $\kappa_2(\hat{R})$ and $\kappa_2(A\hat{R}^{-1})$. We show that the error is small under the assumption that $\kappa_2(A)u \ll 1$. As a result, the condition number of $\hat{Y}$ will be bounded by a small constant related to the conditioning of S and SQ_A . Thus, the computed AP has a reasonable condition number provided that the numerical errors ε_1 and ε_2 are small.

²The ‘ $\dagger$ ’ superscript on a matrix denotes the matrix pseudo-inverse.

5.4.2 Bounding the numerical errors

We now bound the errors ε_1 and ε_2 .

Lemma 5.4. *Assume the same setup as in Lemma 5.2 and Lemma 5.3, where $\varepsilon_1 = \|E\|_2 \|R^{-1}\|_2$ and $\varepsilon_2 = \|\Delta Y\|_2 \|(A\hat{R}^{-1})^\dagger\|_2$, and all quantities in finite precision are computed with unit round-off u . Furthermore, define*

$$C_1 := \sqrt{sn}\gamma_m + \sqrt{n}\tilde{\gamma}_{sn}(1 + \sqrt{sn}\gamma_m), \quad k(S) := \|S\|_2 \|(SQ_A)^\dagger\|_2. \quad (5.12)$$

If $\kappa_2(A) < 1/(C_1 k(S))$, then we have

$$\varepsilon_1 \leq C_1 k(S) \kappa_2(A),$$

and

$$\varepsilon_2 \leq \frac{n\gamma_n \kappa_2(SQ_A)(\kappa_2(A)\kappa_2(SQ_A) + \varepsilon_1)(1 + \varepsilon_1)}{(1 - \varepsilon_1 - \sqrt{n}\gamma_n(\kappa_2(A)\kappa_2(SQ_A) + \varepsilon_1))(1 - \varepsilon_1)}.$$

Proof. We first note that $E = E_1 + E_2$, where $E_1 = \hat{B} - SA$ is the error arising from matrix-matrix multiplication and $E_2 = \tilde{Q}\hat{R} - \hat{B}$ is the error arising from the Householder QR algorithm. We can bound these terms using (5.4) and (5.5) to find

$$\begin{aligned} \|E\|_2 &\leq \|E_1\|_2 + \sqrt{n}\tilde{\gamma}_{sn}\|SA + E_1\|_2 \\ &\leq \sqrt{n}\tilde{\gamma}_{sn}\|S\|_2\|A\|_2 + (1 + \sqrt{n}\tilde{\gamma}_{sn})\sqrt{sn}\gamma_m\|S\|_2\|A\|_2 = C_1\|S\|_2\|A\|_2. \end{aligned}$$

This is then combined with

$$\sigma_{\min}(R) = \sigma_{\min}(QR) = \sigma_{\min}(SQ_A R_A) \geq \sigma_{\min}(SQ_A)\sigma_{\min}(A).$$

As for $\|\Delta Y\|_2$, by (5.7) we have

$$\|\Delta Y\|_2 \leq \|\Delta Y\|_F \leq \frac{\sqrt{n}\gamma_n \kappa_2(\hat{R})}{1 - \sqrt{n}\gamma_n \kappa_2(\hat{R})} \|A\hat{R}^{-1}\|_F \leq \phi(\hat{R}) \|A\hat{R}^{-1}\|_2,$$

where

$$\phi(\hat{R}) := \frac{n\gamma_n \kappa_2(\hat{R})}{1 - \sqrt{n}\gamma_n \kappa_2(\hat{R})} \leq \frac{n\gamma_n(\kappa_2(A)\kappa_2(SQ_A) + \varepsilon_1)}{1 - \varepsilon_1 - \sqrt{n}\gamma_n(\kappa_2(A)\kappa_2(SQ_A) + \varepsilon_1)}. \quad (5.13)$$

We have $\varepsilon_2 \leq \phi(\hat{R})\kappa_2(A\hat{R}^{-1})$, which can be bounded using the result on $\kappa_2(A\hat{R}^{-1})$ in Lemma 5.2. $\square$

The condition that $\kappa_2(A) < 1/(C_1 k(S))$ is closely related to the assumption $\kappa_2(A) \ll u^{-1}$ (see Section 5.4.3 for further discussion), which ensures that $\varepsilon_1 < \kappa_2(A)u \ll 1$. The bound $\varepsilon_2 \leq$

$\phi(\hat{R})\kappa_2(A\hat{R}^{-1})$, where $\phi(\hat{R})$ is defined in (5.13), tells us that the dominant term in the bound for ε_2 is $n^2\kappa_2(SQ_A)\kappa_2(A)u$, when $\varepsilon_1 < 1$.

5.4.3 Bounding $\kappa_2(\hat{Y})$

We now combine Lemmas 5.2 to 5.4 to obtain a bound on the condition number of $\hat{Y}$, which is the computed version of AR^{-1} . We find that $\kappa_2(\hat{Y}) \leq 4\kappa_2(SQ_A) + 1$ provided that $\kappa_2(A)$ is sufficiently small.

Theorem 5.5. *Let $A \in \mathbb{R}^{m \times n}$ have linearly independent columns, $S \in \mathbb{R}^{s \times n}$ have linearly independent rows, and $m > s > n$. Let $A = Q_A R_A$ and $SA = QR$ be the economized QR factorizations of A and $B = SA$, respectively. Assume that*

$$\kappa_2(SQ_A) < 49, \quad n^2 u < 1/201, \quad \frac{m}{n} + \sqrt{n} > 200, \quad (5.14)$$

and that $\kappa_2(A)$ is sufficiently small so that

$$\kappa_2(A) < \frac{1}{3C_1 k(S)}, \quad (5.15)$$

where $C_1 = \mathcal{O}((\sqrt{sn}m + sn^{3/2})u)$ and $k(S)$ are defined in (5.12). Compute the preconditioned matrix $\hat{Y}$ as detailed in Lemma 5.2 and Lemma 5.3.³ Then,

$$\kappa_2(\hat{Y}) \leq 4\kappa_2(SQ_A) + 1.$$

Proof. Throughout the proof, we assume the notation introduced in Lemmas 5.2 to 5.4. By assumption (5.15) we immediately see that $\varepsilon_1 < 1/3$, defined in Lemma 5.2, and hence $\hat{R}$ and $A\hat{R}^{-1}$ are full rank. It follows that $\kappa_2(A\hat{R}^{-1}) \leq 2\kappa_2(SQ_A)$. Furthermore, also by Lemma 5.2, we find

$$\kappa_2(\hat{R}) \leq \frac{\kappa_2(R) + \varepsilon_1}{1 - \varepsilon_1} \leq \frac{1}{2}(3\kappa_2(SQ_A)\kappa_2(A) + 1) < \frac{1}{2}\left(\frac{1}{C_1} + 1\right). \quad (5.16)$$

By using the fact that $m > s > n$, we can show that $C_1 > (m + n^{3/2})\gamma_n$, which can be substituted into (5.16). One can use the bound on $\kappa_2(\hat{R})$ to then bound $\phi(\hat{R})$, defined in (5.13), as

$$\phi(\hat{R}) = \frac{n\gamma_n\kappa_2(\hat{R})}{1 - \sqrt{n}\gamma_n\kappa_2(\hat{R})} < \frac{\frac{1}{2}n\left(\frac{1}{m+n^{3/2}} + \gamma_n\right)}{1 - \frac{1}{2}\sqrt{n}\left(\frac{1}{m+n^{3/2}} + \gamma_n\right)} = \frac{\sqrt{n}c(m, n)}{1 - c(m, n)},$$

³That is, compute the matrix-matrix product SA , compute its R factor by the Householder QR algorithm, and finally compute the preconditioned matrix by solving AR^{-1} with forward substitution.

where

$$c(m, n) = \frac{1}{2}\sqrt{n} \left(\frac{1}{m + n^{3/2}} + \gamma_n \right).$$

The assumptions in (5.14) imply that $c(m, n) < (4\sqrt{n}\kappa_2(SQ_A) + 1)^{-1}$, where we used that $n^2u < 1/201$, i.e. $n\gamma_n < 1/200$. Now we have that $\phi(\hat{R}) < (4\kappa_2(SQ_A))^{-1}$ and so

$$\varepsilon_2 \leq \phi(\hat{R})\kappa_2(A\hat{R}^{-1}) \leq \kappa_2(SQ_A)\phi(\hat{R})\frac{1 + \varepsilon_1}{1 - \varepsilon_1} < \frac{1}{2}.$$

By Lemma 5.3, $\hat{Y}$ is full rank and its condition number is bounded by

$$\kappa_2(\hat{Y}) \leq \frac{\kappa_2(A\hat{R}^{-1}) + \varepsilon_2}{1 - \varepsilon_2} < 4\kappa_2(SQ_A) + 1.$$

□

It is worth discussing the assumptions of Theorem 5.5. Consider the assumption in (5.15). The $k(S)$ term (see (5.12)) is a small constant depending on the particulars of the embedding matrix. The constant C_1 (see (5.12)) can be bounded by the product of a low-degree polynomial in m , n , and s and the unit-round-off u , say $C_1 \leq p(m, n, s)u$. We specifically have $C_1 = \mathcal{O}(\sqrt{sn}(m + \sqrt{sn})u)$, as $\gamma_m = \mathcal{O}(mu)$ and $\tilde{\gamma}_{sn} = \mathcal{O}(snu)$. In classical stability analysis, this polynomial term $p(m, n, s)$ can be more or less ignored. As a result, a condition of the form $\kappa_2(A) < (p(m, n, s)u)^{-1}$ is often loosely restated as $\kappa_2(A) \ll u^{-1}$.

The rationale behind ignoring these m , n , and s terms is that classic stability analysis, as we performed in this section, provides us with worst-case bounds on numerical errors. These are generally pessimistic [140]; composing n operations would only result in an error of order nu if each rounding error is of the same sign and of maximum magnitude [141]. We could improve these bounds by using probabilistic backward error analysis, which would give us results proportional to $\sqrt{nu}$ instead of nu [141, 142]. However, it must be noted that even probabilistic stability analysis would theoretically result in a large factor multiplied by u .

The other assumptions of Theorem 5.5, in (5.14), are reasonable in practice. The condition number of SQ_A is generally observed to be bounded by a modest number, such as 5 or 10, with high probability. Furthermore, we assume that the problem dimensions are small compared to the unit round-off. Most importantly, Theorem 5.5 informs us that, provided that $\kappa_2(A) \ll u^{-1}$, the condition number of the preconditioned matrix is independent of $\kappa_2(A)$ (with high probability).

5.4.4 The backward error of sketch-and-apply

Until now, we have only considered the condition number of the preconditioned matrix AP . The analysis in this section related rounding errors incurred in the final steps — solving $(AR^{-1})z = b$

with LSQR and computing $x = R^{-1}z$ with back substitution — to the backward error of sketch-and-apply. It is not immediate that the last step in the algorithm preserves the numerical stability of the algorithm. After all, solving linear systems with R in LSQR iterations is one potential reason for the numerical instability of sketch-and-precondition algorithms. Nonetheless, we show that sketch-and-apply performs as well as LSQR on very well-conditioned matrices and that the numerical error in the last step is $\mathcal{O}(u)$ instead of $\mathcal{O}(u\kappa_2(A))$ (see Theorem 5.6).

Throughout this analysis, we will not consider a sketch-and-solve initialization and consider stability properties from any starting point. We reiterate that the following theorem relates to sketch-and-apply without any smoothing. Smoothed sketch-and-apply is investigated in Section 5.5.

We show that the backward error in sketch-and-apply is of the same order as the backward error from unpreconditioned LSQR with $\text{fl}(AR^{-1})$, under the assumptions of Theorem 5.5.

Theorem 5.6. *Assume the embedding matrix $S \in \mathbb{R}^{s \times m}$ and $A \in \mathbb{R}^{m \times n}$ satisfy assumptions (5.14) and (5.15) in Theorem 5.5. Apply Algorithm 10 and assume that LSQR terminates at an iterate $\hat{z}$ that satisfies the backward error⁴*

$$\hat{z} = (\hat{Y} + \Delta\hat{Y})^\dagger(b + \delta b). \quad (5.17)$$

Then, the computed solution $\hat{x}$ to $Ax = b$ has the backward error $(A + \Delta A)\hat{x} = b + \delta b$ satisfying

$$\|\Delta A\|_2 < \|S\|_2 \|A\|_2 \left(6.04n\gamma_n \|(SQ_A)^\dagger\|_2 + 2.01\|\Delta\hat{Y}\|_2 \right).$$

Proof. The final step of Algorithm 10 involves solving the triangular system $\hat{R}\hat{x} = \hat{z}$. We know from (5.8) that the computed solution $\hat{x}$ satisfies

$$(\hat{R} + \Delta\hat{R})\hat{x} = \hat{z}, \quad \|\Delta\hat{R}\|_2 \leq \gamma_n \sqrt{n} \|\hat{R}\|_2.$$

We now have

$$\hat{x} = (\hat{R} + \Delta\hat{R})^{-1}\hat{z} = (\hat{R} + \Delta\hat{R})^{-1}(\hat{Y} + \Delta\hat{Y})^\dagger(b + \delta b) = (A + \Delta A)^\dagger(b + \delta b),$$

where $\Delta A = (\hat{Y}\hat{R} - A) + \Delta\hat{Y}\hat{R} + \hat{Y}\Delta\hat{R} + \Delta\hat{Y}\Delta\hat{R}$. For the $\hat{Y}\hat{R} - A$ term we have that each row satisfies

$$\hat{y}_i^T \hat{R} - a_i^T = -\hat{y}_i^T \Delta_i \hat{R}, \quad |\Delta_i \hat{R}| \leq \gamma_n |\hat{R}|,$$

so that $\|\hat{Y}\hat{R} - A\|_2 \leq n\gamma_n \|\hat{R}\|_2 \|\hat{Y}\|_2$. It follows that

$$\|\Delta A\|_2 \leq \|\hat{R}\|_2 \left((n + \sqrt{n})\gamma_n \|\hat{Y}\|_2 + (1 + \sqrt{n}\gamma_n) \|\Delta\hat{Y}\|_2 \right),$$

⁴Note that the backward error term $\Delta\hat{Y}$ is not the same as the numerical error ΔY resulting from solving the triangular system $Y = A\hat{R}^{-1}$ in finite precision.

where $\|\hat{R}\|_2 \leq (1 + C_1)\|S\|_2\|A\|_2 < 2\|S\|_2\|A\|_2$, and

$$\|\hat{Y}\|_2 \leq \|A\hat{R}^{-1}\|_2 + \|\Delta Y\|_2 \leq (1 + \phi(\hat{R})) \frac{\|(SQ_A)^\dagger\|_2}{1 - \varepsilon_1} < 1.51\|(SQ_A)^\dagger\|_2,$$

where the last inequality follows from the assumptions (5.14) and (5.15). We obtain

$$\|\Delta A\| \leq 2\|S\|_2\|A\|_2 \left(3.02n\gamma_n\|(SQ_A)^\dagger\|_2 + 1.005\|\Delta\hat{Y}\|_2 \right),$$

where we used that $n + \sqrt{n} \leq 2n$ and $\sqrt{n}\gamma_n \leq n\gamma_n \leq 1/200$ by (5.14). $\square$

The moral of Theorem 5.6 is that sketch-and-apply is backward stable as long as LSQR on $\hat{Y}$ is backward stable, that is, $\|\Delta\hat{Y}\|_2$ and $\|\delta b\|_2/\|b\|_2$ are both $\mathcal{O}(u)$ in (5.17). Under the conditions of Theorem 5.6, the matrix $\hat{Y}$ is well-conditioned with high probability. As discussed in the previous chapter, we cannot provide rigorous theoretical backing that this means LSQR will be stable. However, numerical experiments provide strong evidence.

Remarkably, the final step of the algorithm does not influence the accuracy of the solution, i.e., solving $Rx = y$ with back substitution. This is despite the fact that R is ill-conditioned when A is. The mechanism with which stability is established is similar to the backward stability of an algorithm based on repeated CholeskyQR [281].

5.4.5 Ill-conditioning caused by poor column scaling

If the ill-conditioning in an LS problem is caused by poor column scaling in A , we strongly recommend using QR-based sketch-and-apply techniques. The reason is that computing AR^{-1} is invariant to column scaling, while the SVD is not. In fact, the assumption that $\kappa_2(A) \ll u^{-1}$ in Section 5.4 can be replaced by

$$\min_D \{\kappa_2(AD) : D = \text{diag}(b^{k_i}), k_i \in \mathbb{Z}, i = 1, \dots, n\} \ll u^{-1},$$

where b is the machine base (usually $b = 2$). Of course, one can also pre-process an LS problem by scaling the columns of A by powers of the machine base so that each column has a norm that is close to 1.

5.5 On the numerical stability of smoothed sketch-and-apply

We now analyze smoothed sketch-and-apply (see Algorithm 11). When $\kappa_2(A) \ll 1/u$, Theorem 5.5 states that sketch-and-apply computes a preconditioned matrix with a condition number independent of $\kappa_2(A)$ with high probability. Moreover, we can reliably estimate $\kappa_2(A)$ by $\kappa_2(SA)$. We do not

have a guarantee on the stability of sketch-and-apply if $\kappa_2(A) \gtrsim 1/u$. It is in this context that we consider smoothing.

We note that smoothing is one of various ways in which rank-deficiency can be tackled. One may add explicit regularisation that transforms the problem to be full-rank; we consider this in Chapter 6. This is also employed in the iterative LS solvers presented in [101]. Alternatively, Cartis et al. suggest computing a rank-revealing decomposition of the sketch SA and truncating the preconditioner to be numerically full-rank [47], similarly to using a truncated SVD as in [187].

One could consider Algorithm 11 from two points of view. Firstly, a practitioner may add an $\mathcal{O}(u\|A\|_2)$ Gaussian perturbation to a severely ill-conditioned problem. As a result, the perturbed problem can be accurately solved with sketch-and-apply. Secondly, floating point errors arising from the representation of any matrix in finite precision could be modeled as a Gaussian perturbation and then smoothing partially explains the success of sketch-and-apply to highly ill-conditioned LS problems.

5.5.1 Bounding $\kappa_2(\hat{Y})$

Similar to the stability analysis in Section 5.4, we have the following steps in finite precision after we have smoothed A :

1. $\tilde{A} = A + \sigma G / \sqrt{m}$,
2. $\hat{B} = S\tilde{A} + E_1$,
3. $\tilde{Q}\hat{R} = \hat{B} + E_2$,
4. $\hat{Y} = \tilde{A}\hat{R}^{-1} + \Delta Y$,

where the hats denote that we are accounting for rounding errors and the scaling parameter σ controls how much A is perturbed. Assuming σ is sufficiently large to ensure that $\tilde{A}$ satisfies (5.15), we then show that the stability analysis from Section 5.4 carries over to $\tilde{A}$.

The following theorem tells us how large σ needs to be.

Theorem 5.7. *Let $A \in \mathbb{R}^{m \times n}$ with $m \geq 3n$, $S \in \mathbb{R}^{s \times n}$ have linearly independent rows, and let $m > s > n$. Set*

$$\tilde{A} = A + \sigma G / \sqrt{m}, \quad \sigma = 52\|A\|_2 k(S) s m \sqrt{n} u, \quad (5.18)$$

where $G \in \mathbb{R}^{m \times n}$ is a standard Gaussian matrix with i.i.d. entries, $k(S)$ is defined in (5.12) and u is the unit round-off. Let $\tilde{A} = Q_{\tilde{A}} R_{\tilde{A}}$ and $S\tilde{A} = QR$ be economized QR factorizations of $\tilde{A}$ and $B = S\tilde{A}$, respectively. Lastly, assume that⁵

$$\kappa_2(SQ_{\tilde{A}}) < 49, \quad \frac{m}{n} + \sqrt{n} > 200, \quad smu < 1/67. \quad (5.19)$$

⁵We also assume that $cn < m$ for the same integer c defined in $\tilde{\gamma}_{sn}$ to simplify the notation.

Now compute first the product $S\tilde{A}$, then the upper triangular factor R by the Householder QR algorithm on $S\tilde{A}$, and finally compute $\hat{Y} = \tilde{A}R^{-1}$ with forward substitution. Assume all computations are performed in finite precision. Then,

$$\mathbb{P}\left(\kappa_2(\hat{Y}) > 4\kappa_2(SQ_{\tilde{A}}) + 1\right) < 2^{n-m},$$

which completes the proof.

Proof. We start by showing that σ in (5.18) is sufficiently large to ensure that $\tilde{A}$ satisfies (5.15) in Theorem 5.5. To this end, note the assumptions in (5.19) imply that $mu < 1/64$ and $csnu < 1/64$ so that we have $\gamma_m < 1.02mu$ and $\tilde{\gamma}_{sn} < 1.02csnu < 1.02smu$. We can then bound C_1 defined in (5.12) as

$$C_1 < 1.02 \times 65sm(1 + \sqrt{n})u/64 < 2.1sm\sqrt{nu}.$$

As a result

$$\mathbb{P}\left(\kappa_2(\tilde{A}) > \frac{1}{3C_1k(S)}\right) < \mathbb{P}\left(\kappa_2(\tilde{A}) > \frac{8.25}{52k(S)sm\sqrt{nu}}\right) < 2^{n-m}.$$

The final result on $\kappa_2(\hat{Y})$ follows by applying Theorem 5.5 to $\tilde{A}$, which is possible since the last assumption in (5.19) is stronger than $nu < 1/201$. $\square$

Theorem 5.7 shows that the magnitude of the perturbation $\sigma G/\sqrt{m}$ needs to be approximately $\mathcal{O}(sm\sqrt{nu})$ to ensure the condition number of $\tilde{A}$ is sufficiently small, that is, $\kappa_2(A) < 1/u$. The low-degree polynomial term $sm\sqrt{n}$ arises from (5.15), which includes the constant $C_1 = \mathcal{O}(\sqrt{sn}(m + \sqrt{sn})u)$. As discussed in Section 5.4.4, these are often pessimistic bounds. Probabilistic error analysis provides a theoretical justification to select σ proportional to $\sqrt{smn}^{1/4}u$. This may still be a relatively large perturbation if the dimensions of the problem are enormous, enlarging the backward error to an unacceptable level (see Section 5.5.2).

5.5.2 The backward error of smoothed sketch-and-apply

In Section 5.5.1, we showed that a small additive random perturbation to a small-skinny matrix ensures that the condition number of the perturbed matrix is sufficiently small. This allows us to conclude that $\kappa_2(\hat{Y})$ is small with high probability. We now show that smoothed sketch-and-apply (see Algorithm 11) computes a backward error similar to Theorem 5.6 with an additional term from the additive perturbation.

Theorem 5.8. *Assume the same setup as in Theorem 5.7. Apply Algorithm 10 to $\tilde{A}x = b$ and assume LSQR terminates at an iterate $\hat{z}$ that satisfies the backward error*

$$\hat{z} = (\hat{Y} + \Delta\hat{Y})^\dagger(b + \delta b).$$

Then, the computed solution $\hat{x}$ to $Ax = b$ has the backward error $(A + \Delta A)\hat{x} = b + \delta b$ satisfying

$$\|\Delta A\|_2 \leq \|A\|_2 \left[\|S\|_2 (1 + 197k(S)sm\sqrt{nu}) \right. \\ \left. \left(6.04n\gamma_n \|(SQ_{\tilde{A}})^\dagger\|_2 + 2.01\|\Delta \hat{Y}\|_2 \right) + 197k(S)sm\sqrt{nu} \right] \quad (5.20)$$

with probability at least $1 - 2^{n-m} - 10^{-3}$.

Proof. We combine Theorem 5.6 and Theorem 5.7 to obtain the result. Additionally, we use the following classic result in the analysis of Gaussian matrices [72]:

$$\mathbb{P} \left(\|G\|_2 / \sqrt{m} > 1 + \sqrt{\frac{n}{m}} + \frac{t}{\sqrt{m}} \right) \leq e^{-t^2/2},$$

applied with $t = 3.8$. This shows $\|\tilde{A}\|_2 \leq \|A\|_2 + \sigma\|G\|_2/\sqrt{m} \leq \|A\|_2 + 3.78\sigma$ with probability at least 10^{-3} , where $\sigma = 52\|A\|_2 k(S)sm\sqrt{nu}$. $\square$

Theorem 5.8 allows us to conclude that under mild assumptions, smoothed sketch-and-apply results in a backward error of the same order as the backward error of unpreconditioned LSQR on $\hat{Y}z = b$. Under the conditions of the theorem, $\hat{Y}$ is indeed a well-conditioned matrix and LSQR is expected to obtain backward stable solutions. We removed the assumptions on $\kappa_2(A)$.

Theorem 5.8 suggests that the perturbation needs to be of size $\mathcal{O}(sm\sqrt{nu})$. This would be impermissible for many problems, especially if the dimensions are large. We recommend a perturbation of magnitude approximately $10\|A\|_2 u$. Although we lack the theoretical evidence for this choice, it works in our experiments (see Section 5.6.2).

Smoothing an LS problem should be done with much caution. There are many contexts where $\kappa_2(A) > u^{-1}$, yet perturbing the problem is unnecessary. For instance, if the ill-conditioning is caused by poor column scaling, sketch-and-apply will still compute accurate solutions without smoothing (see Section 5.7). In this sense, $\kappa_2(A)$ is not an effective predictor of whether smoothing will be beneficial. We suggest one should smooth only if sketch-and-apply does not converge (see Algorithm 12).

5.6 Master sketch-and-apply algorithm and numerical experiments

We now present the implementation of the sketch-and-apply algorithms (see Algorithm 12) in a similar fashion to the overall solver in Blendenpik [14]. We perform the standard initial steps, i.e., drawing an embedding matrix S , computing the sketch SA , and computing the QR factorization $QR = SA$ using the Householder QR algorithm. Next, we compute the preconditioned matrix $Y = AR^{-1}$ explicitly with forward substitution and use (unpreconditioned) LSQR to solve $\min \|Yz - b\|$.

We include sketch-and-solve initialization in this algorithm to speed up convergence. This step consists of computing $z_0 = Q^T S b$. If LSQR converges to the desired tolerance, we compute $x = R^{-1} z$ with back substitution. In this case, we assume our algorithm computed a backward stable solution.

In the case where LSQR does not converge, we perturb/smooth A . This will only happen when A is numerically rank-deficient, as we've theoretically shown in this chapter. In this case, we resolve the rank-deficiency by smoothing A .⁶ In particular, we compute $\tilde{A} = A + \sigma G / \sqrt{m}$ for a Gaussian matrix G with $\sigma = 10 \|A\|_2 u$ and perform the sketch-and-apply steps including a sketch-and-solve initial guess on $\tilde{A} x = b$. We recommend doing this because there are examples, although rare, where sketch-and-apply converges to an accurate solution only after smoothing (see Section 5.6.2). The spectral norm of A can be estimated, for instance, with the sketch SA or using the power method.

Algorithm 12 Master algorithm for sketch-and-apply to solve (5.1). Here, HHQR refers to the Householder QR algorithm.

- 1: Draw a random sketching matrix $S \in \mathbb{R}^{s \times m}$, where $m \gg s > n$
 - 2: Compute $B = SA$ and $c = Sb$
 - 3: Compute both Q and R of the QR factorization of B using HHQR
 - 4: Compute $Y = AR^{-1}$ with forward substitution
 - 5: Compute initial guess $z_0 = Q^T c$
 - 6: Solve $\min \|Yz - b\|$ with LSQR and no preconditioner and initial guess z_0
 - 7: **if** LSQR converges to the desired tolerance **then**
 - 8: Compute $x = R^{-1} z$ with back substitution
 - 9: **else**
 - 10: Draw a random standard Gaussian matrix $G \in \mathbb{R}^{m \times n}$ with i.i.d. entries
 - 11: Compute $\tilde{A} = A + \sigma G / \sqrt{m}$ for $\sigma = 10 \|A\|_2 u$
 - 12: Compute $B = S\tilde{A}$
 - 13: Compute both Q and R of the QR factorization of B using HHQR
 - 14: Compute $Y = \tilde{A}R^{-1}$ with forward substitution
 - 15: Compute initial guess $z_0 = Q^T c$
 - 16: Solve $\min \|Yz - b\|$ with LSQR and no preconditioner and initial guess z_0
 - 17: Compute $x = R^{-1} z$ with back substitution
 - 18: **end if**
-

It should be noted that without smoothing, LSQR iteration can converge to a solution with a sub-optimal residual (see Figure 5.6 (right)). However, it is infeasible to compute the backward error when m is large as it requires computing the smallest singular value of an $m \times (m + n)$ matrix (although it is possible to reduce the cost [129]). If the residual obtained from sketch-and-apply is less accurate than desired, one could estimate the condition number of A via the condition number of R . If the condition number estimate of R is of the same order of magnitude as u^{-1} , then the problem is severely ill-conditioned and one could try smoothing to potentially improve the accuracy of the computed solution. Smoothing can improve the convergence rate of LSQR, even if sketch-and-apply obtains a backward stable solution without smoothing (see Figure 5.5). Nonetheless one

⁶As mentioned throughout, this is certainly not the only resolution to rank-deficiency. Primarily, one can truncate the (low-rank) decomposition of SA before computing a preconditioner (as in [47, 187], see Section 4.5.1). Alternatively, one can apply a different type of regularisation, such as Tikhonov regularisation discussed in Chapter 6.

should be careful with smoothing because examples exist where $\kappa_2(A)$ is large but smoothing remains unnecessary (see Figure 5.5). Exactly when and when not to smooth remains unclear.

5.6.1 Numerical experiments with sketch-and-apply

Now we examine the performance of sketch-and-precondition with sketch-and-solve initialization, sketch-and-apply with and without sketch-and-solve initialization, and the QR-based direct solver. The experiment considers ill-conditioned LS problems that are randomly generated (see Section 4.2.5 for details on how A and b are formed) and can be stored in local cache. The QR solver is implemented with the `qr` and backslash `\` (with R) commands in MATLAB. The execution times are shown in Figure 5.3.

We find that on this scale, sketch-and-apply, especially with SAS, is competitive relative to QR in terms of computational time. A sketch-and-solve initialization improves the computing time as fewer iterations are needed. Sketch-and-precondition with SAS is the computationally most efficient, and will in almost all cases converge to a solution with optimal residual and backward error. For most practical applications, this should remain the preferred option.

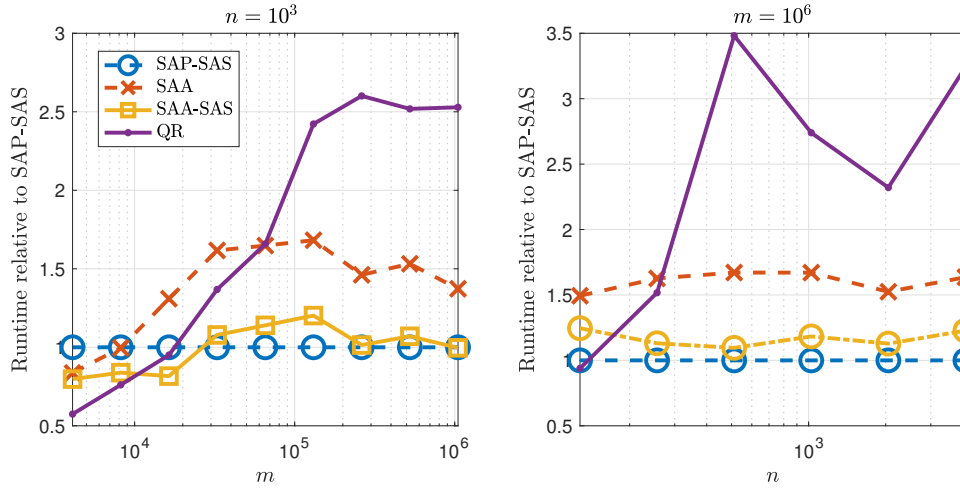


Figure 5.3: The relative execution time to solve large LS problems using sketch-and-precondition with sketch-and-solve initialization (SAS), sketch-and-apply, sketch-and-apply with SAS, and Householder QR. For each problem $\kappa(A) = 10^{10}$ and the noise level $\|e\|_2 = 10^{-10}$. The tolerance is set to 10^{-12} and the maximum number of iterations to 100 (this is never reached). Left: Timings for $n = 4000$ and $2^{12} \leq m \leq 2^{20}$. Right: Timings for $m = 10^6$ and $2^7 \leq n \leq 2^{12}$.

We furthermore note that sketch-and-apply (and sketch-and-precondition) algorithms could outperform the direct QR method by a larger factor if it is expensive to communicate with the matrix A . This context occurs, for example, when A is too large to be stored in local cache and is instead stored on disk. Provided we obtain a good sketch of A , the number of iterations in LSQR before convergence will be modest, requiring limited streaming. The QR method, however, requires n views for an $m \times n$ matrix.

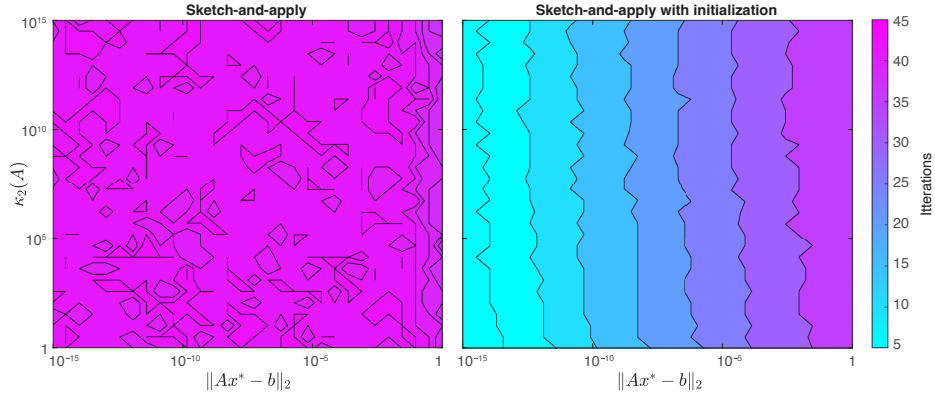


Figure 5.4: The iteration count necessary for convergence when using sketch-and-apply with or without sketch-and-solve initialization.

Figure 5.2 and the analysis throughout the chapter show that sketch-and-apply always obtains backward stable solutions, regardless of whether initialized or not. Figure 5.4 displays that the difference in the two approaches is the number of iterations necessary to get there. The quality of the sketch-and-solve solution depends on the optimal residual, as previously noted in Section 5.2.1. For small optimal residuals, we start closer to a backward stable solution.

5.6.2 Numerical experiments with smoothed sketch-and-apply

Sketch-and-precondition with initialization, sketch-and-apply, its smoothed version, and Householder QR can compute accurate solutions to LS problems, even when $\kappa_2(A) > 1/u$ (see Figure 5.5, the problems are generated as explained in Section 4.2.5). Therefore, smoothing is only sometimes required for extremely ill-conditioned least squares problems. However, even when smoothing is unnecessary, there is some potential benefit to smoothing because the perturbed LS problem can lead to rapid LSQR convergence. Of course, there is a trade-off here as one converges rapidly to an accurate solution of the perturbed problem, and the computed solution may be less accurate for the original LS problem of interest.

However, there are also examples for which smoothing provides a more accurate solution to the original problem of $\min_x \|Ax - b\|$, not $\min_x \|\tilde{A}x - b\|$ (see Figure 5.6). For example, we take an LS problem involving a 1000×100 Kahan matrix with $\theta = 1.1$ from the MATLAB `gallery` collection and another one involving a column-scaled 1000×10 Vandermonde matrix involving equally spaced points between -1 and 1 . The Kahan and Vandermonde matrices are designed so that their condition numbers are $> 1/u$. Without smoothing, even the QR-based algorithm fails to compute an accurate solution along with all the other methods. This is because the problem is so ill-conditioned that even a backward stable solution can behave wildly. However, after smoothing, sketch-and-apply computes an accurate solution to the original LS problems. It should be noted that one could also smooth before solving the problems with QR to obtain more accurate solutions.

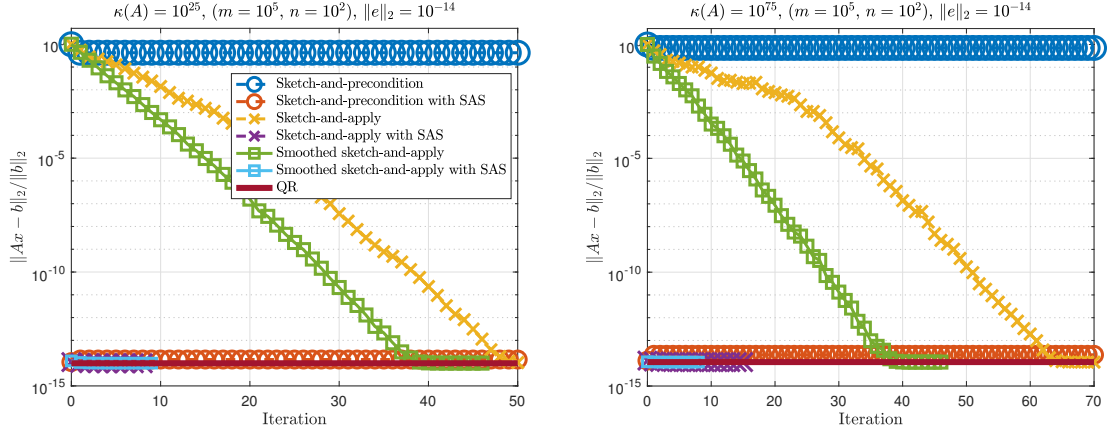


Figure 5.5: Smoothing is not always needed for LS problems with $\kappa_2(A) > 1/u$. These matrices are formed by drawing singular vectors from the Haar distribution and letting the singular values decay exponentially from 1 to $1/\kappa_2(A)$. For comparison, we show the residual error computed by Householder QR. These problems are so ill-conditioned that the backward error is not an informative measure, i.e., most computed solutions (e.g., with very few LSQR iterations) give a backward error $\mathcal{O}(u)$. Note that the reported residual is always the residual in the original problem with A , not the residual of the smoothed problem.

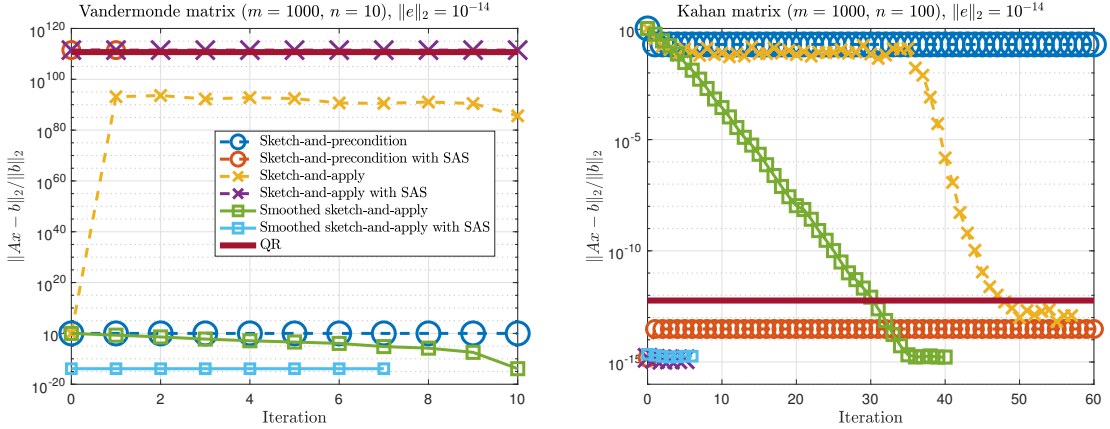


Figure 5.6: Smoothing can solve extremely ill-conditioned LS problems, even when sketch-and-apply and Householder QR cannot. For the Vandermonde matrix, we can only obtain accurate solutions using a smoothed algorithm (note the y-axis is the residual $\|Ax - b\|_2 / \|b\|_2$, not the residual in the smoothed problem). The problem with the Kahan matrix can only be solved optimally with certain sketch-and-apply variations; the initialized sketch-and-precondition algorithm and standard sketch-and-apply attain sub-optimal residuals. Again, these problems are too ill-conditioned for the backward error to be informative.

5.6.3 Sparse matrices

We briefly discuss the performance of the various algorithms in question on sparse matrices. Firstly, it should be noted that any sketch-and-apply variant will not respect the sparsity: the matrix $Y = AR^{-1}$ is generally dense. As a result, convergence will be slower and computational cost will be unnecessarily high. Remarkably, Figure 5.7 shows that standard sketch-and-precondition can be numerically stable for sparse matrices with few non-zero entries. It appears that if $\text{nnz}(A)$ is sufficiently small, the rounding errors compound less and the algorithm finds accurate solutions.

A precise explanation is left for future work. Unsurprisingly, all algorithms with sketch-and-solve initialization converge rapidly to accurate solutions.

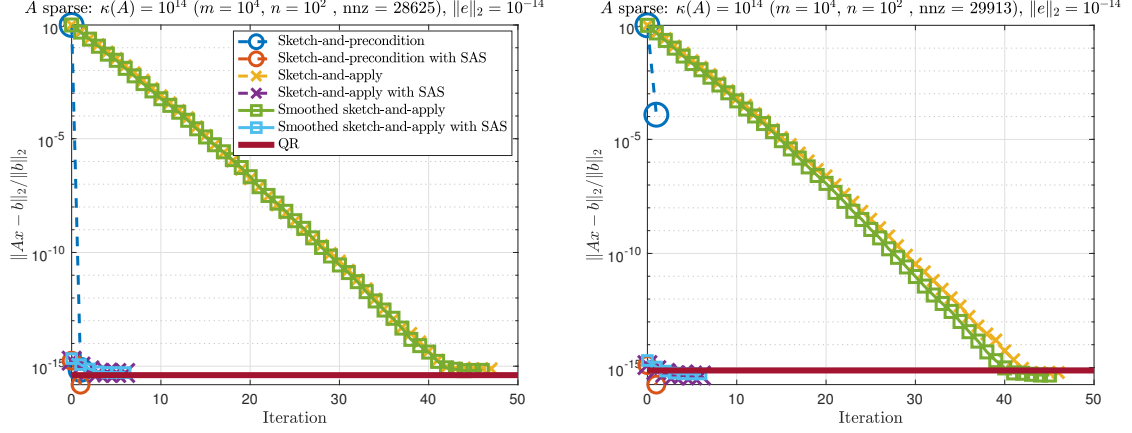


Figure 5.7: Sketch-and-apply algorithms cannot take advantage of sparsity of A . Note that sketch-and-precondition without sketch-and-solve initialization can lead to accurate solutions for sparse matrices with few non-zero entries (left figure). This behavior depends nontrivially on the sparsity etc; in the right figure, we observe the numerical instabilities we have seen throughout this work.

5.7 Summary and outlook

We have proposed a stable iterative LS solver sketch-and-apply, derived from the sketch-and-precondition framework by explicitly computing the preconditioned matrix AP and using an unpreconditioned iterative solver on AP . We furthermore demonstrated that using sketch-and-solve initialization greatly improves convergence properties as well as the maximal attainable accuracy of sketch-and-precondition. Practically, the sketch-and-apply algorithm requires the same computational cost as direct solvers, i.e. $\mathcal{O}(mn^2)$, although it can be faster in terms of running time (see Figure 5.3). It may therefore be seen as a theoretical contribution more so than a practical one. The radical improvement in performance of sketch-and-precondition when combined with sketch-and-solve initialization is the main take-away for practitioners. This was previously argued in [47, 187].

As to the numerical stability of sketch-and-apply, we were not able to state that the backward error $\|\Delta A\|_2$ is $\mathcal{O}(u\|A\|_2)$ (see Theorems 5.6 and 5.8). Instead, we have shown that $\|\Delta A\|_2/\|A\|_2$ is of the same order as the backward error $\|\Delta \hat{Y}\|$ incurred when $\hat{Y}z = b$ is solved with unpreconditioned LSQR, where $\hat{Y}$ is well-conditioned. The finite-precision behaviour of Krylov subspace solvers is a difficult topic, strongly dependent on the particular implementation (as discussed in Section 4.4.2). Empirically, however, we find that sketch-and-apply is always stable when $\kappa_2(A) \ll u^{-1}$.

We have assumed specific choices for how the randomized preconditioner is constructed. In LSRN [187], for instance, the preconditioner is chosen to be $P = V\Sigma^{-1}$, where $SA = U\Sigma V^T$ is the

SVD of the sketch. The sketch-and-apply technique can also be used with LSRN ideas where the preconditioned matrix is computed as $Y = AV\Sigma^{-1}$. We suspect one can also prove similar results to our sketch-and-apply analysis for sketch-and-apply LSRN. Versions of Lemmas 5.2 and 5.3 hold almost identically for SVD-based sketch-and-apply, with R replaced by ΣV . The key difference is the numerical errors ε_1 and ε_2 , which we have not investigated. Since computing the SVD is typically more expensive than a QR factorization, we recommend sketch-and-apply. One notable exception is when solving a sequence of Tikhonov regularized LS problems for various regularization parameters (see Chapter 6 and [187]).

Backward stability is important to ensure that errors incurred throughout the computational process are as small as one could hope. These errors are the smallest perturbations to the data so that the computed solution exactly solves the perturbed system. However, in any real problems the data A , b , or both will already be perturbed by measurement errors. In this case, we only desire the backward error to be smaller than the size of the measurement error, which are usually not of order machine precision. Backward stability would then not be strictly necessary; all we would need is a reasonably small backward error. One argument for backward stability is if a least squares problem is solved as part of a larger computational routine in which backward errors might accumulate.

There is an important question that has been left unresolved in this chapter: is there an iterative LS solver based on randomized preconditioning that is both stable and has smaller computational cost than direct solvers? In recent work by Epperly, Meier, and Nakatsukasa (2024), such an algorithm is proposed. It is shown that combining sketch-and-solve initialization with two rounds of sketch-and-precondition resolves the numerical instabilities of sketch-and-precondition [101]. The algorithm, which is called SPIR (Sketch-and-Precondition with Iterative Refinement) works as follows

1. Draw a random embedding S , sketch A and b , compute a randomized preconditioner P from the sketch SA .
2. Set $x_0 = \operatorname{argmin} \|S(Ax - b)\|$;
3. Use an iterative solver with initial guess x_0 and preconditioner P to solve $\min \|Ax - b\|$ and find solution x_1 ;
4. Use an iterative solver with initial guess x_1 and preconditioner P to solve $\min \|Ax - b\|$ and find solution x_2 .

We have shown throughout this chapter that x_0 is far from backward stable, but leads to good residual errors. Similarly, x_1 is not always a backward stable solution, but gives optimal residuals and forward errors of similar accuracy as the QR method (see Figure 5.1). Finally, in [101], we show x_2 is a backward stable solution. A significant reason of the good performance of this algorithm

is the careful stopping criteria (also based on sketching), monitoring when to refine the solution and when backward stability is attained. This work not only highlights the potential of randomized preconditioning as a default solver for highly-overdetermined least squares problems, but also emphasizes the importance of leveraging classical NLA knowledge (finite-precision analysis, numerical stability, stopping criteria for iterative solvers, iterative refinement) to unleash the full potential of randomized algorithms.

Chapter 6

Randomized algorithms for Tikhonov regularization in linear least squares

6.1 Introduction

In the previous chapter, we set out to compute backward stable solutions to highly-overdetermined least squares problems. Now, we investigate how we can use randomization to compute solutions with small forward error. We again consider the least squares problem

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2, \quad A \in \mathbb{R}^{m \times n}, \quad b \in \mathbb{R}^m, \quad (6.1)$$

but allow for various different dimensions of the problem: $m \gg n$, $n \gg m$, or problems with rapidly decaying spectra of any dimension, including $m = n$.

As discussed in Section 4.2.3 and displayed in Figures 4.2 and 5.2, backward stable solutions of ill-conditioned problems can have extremely large forward errors. Small rounding errors are amplified in the direction of the smallest singular values and the solution norm blows up. We might be solving a nearby problem, but the solutions may differ enormously (see also Figure 1.2).

From a numerical point of view, ill-conditioning to the point of rank-deficiency can make standard LS solvers fail. In Chapter 5, we resolved this by adding random noise to the matrix A to improve its conditioning. Other works have proposed sketch-and-precondition schemes that can be applied to rank-deficient matrices through truncating decompositions of the sketch [47, 187]. In this chapter, we

discuss another option: namely Tikhonov regularization. This type of regularization will transform any linear system into an overdetermined system of full column rank.

Our treatment will be more general than previous chapters, in that we will also discuss underdetermined and square systems. Section 6.2 will mainly consider highly overdetermined problems, i.e. $m \gg n$, as in most of this thesis. All the theorems and algorithms presented in Section 6.2 may be extended to the highly underdetermined case, i.e. $n \gg m$. We discuss this in Section 6.2.5. Most notably, in this chapter we also discuss sketch-and-precondition-type algorithms for problems of any dimension, including square A , provided the matrix has rapidly decaying singular values. This case is considered in Section 6.3.

Let us discuss the need for regularization beyond numerical stability. Consider a linear inverse problem of the form $Ax_{\text{true}} + e = b$, where $e \in \mathbb{R}^m$ is some unknown noise and x_{true} is some ground truth that we wish to recover from A and b . If A is very ill-conditioned and we were to find the exact solution $x^* = \operatorname{argmin} \|Ax - b\|$, we would have

$$x^* = A^\dagger b = A^\dagger b_{\text{true}} + A^\dagger e = x_{\text{true}} + A^\dagger e.$$

The magnitude of $A^\dagger e$ might be as large as $\|A^\dagger e\| \approx \kappa_2(A)\|e\|$, rendering the solution x^* useless for very ill-conditioned matrices. The ill-conditioning results in amplification of the (e.g. measurement or approximation) error in b and (potentially) A [45, 62]. In this chapter, we set out to resolve this and compute a solution $\hat{x}$ that minimizes the error $\|\hat{x} - x_{\text{true}}\|$.

To do so, we must turn to regularizing the problem before solving it numerically [99]. Regularization roughly refers to ‘filtering out’ the effect of very small singular values, and thereby obtaining solutions with smaller error. Regularization is often proposed in the contexts of inverse problems, which are of interest in a number of domains such as (medical) imaging [31, 198, 199], scattering problems [42, 51, 65], geophysics [147, 244], and astronomy [124, 171]. In many of these applications, A is the discretization of an integral operator [121] and b are the measurements. If the operator A is (very) ill-conditioned, as it often is, then recovering x_{true} from A and b is an ill-posed problem. We demonstrate this using the example of computed tomography.

6.1.1 An example inverse problem: computed tomography

In computed tomography (CT) for radiology, a cross-section of the human body is exposed to X-rays. The goal is to reconstruct the X-ray attenuation of the tissue at different points; this is accomplished by measuring the loss in intensity. A simple physical model for this is described through an integral transform called the *Radon transform*. Following [135], let $f(x_1, x_2)$ be a function describing the X-ray attenuation at a point $(x_1, x_2) \in [-1, 1]^2$ of the image. Any line $L_{\theta, s}$ through the image can be parameterized by $\theta \in [0, 180)$ and $s \in [-1, 1]$ if we define $L_{\theta, s}$ to be the line with distance s from

the origin and orthogonal to the x_1 -axis rotated θ degrees. The Radon transform of f for a given line is simply the line integral

$$\mathcal{R}[f](\theta, s) = \int_{L_{\theta, s}} f(x_1, x_2) dl.$$

Radon showed in 1917 that the object $f(x_1, x_2)$ can be completely reconstructed from a full set of line integrals over all angles, say $g(\theta, s)$ [217]. This is the basis of CT [198].

In an ideal world, we take our measurements (the line integrals $g(\theta, s)$ collected in a sinogram) and apply the inverse Radon transform to reconstruct the image of the body part. In the real world, we only take measurements over a discrete number R angles and a discrete number P lines per angle, such that we measure data $b \in \mathbb{R}^{PR}$. The image is pixelated into a $M \times N$ grid and then reshaped as a vector $x_{\text{true}} \in \mathbb{R}^{MN}$. We can similarly discretize the integral transform $\mathcal{R}$ into a $PR \times MN$ matrix A so that $Ax_{\text{true}} = b_{\text{true}}$. The actual measurements $b = b_{\text{true}} + e$ contain some noise e .

The discretized Radon transform is available as a fast transform in MATLAB, but we choose to explicitly form the matrix A so we can investigate its spectrum. This is done by applying the `radon` function in MATLAB so MN images of size $M \times N$ with just one entry equal to one and the rest zero.

We take the modified Shepp-Logan phantom of size 110×110 (a model for the cross-section of a human head, see [144, 232, 248]) as an example image. The Radon transform is computed over $R = 90$ angles and $P = 159$ lines. Figure 6.1 shows this image and the associated sinogram which is obtained through the matrix-vector multiply Ax_{true} . We add a small amount of artificial noise to this sinogram to create the data b , and then solve the LS problem $x^* = \text{argmin} \|Ax - b\|$. The reconstruction is shown in the right-most panel in Figure 6.1. It is immediately apparent that it is completely useless.

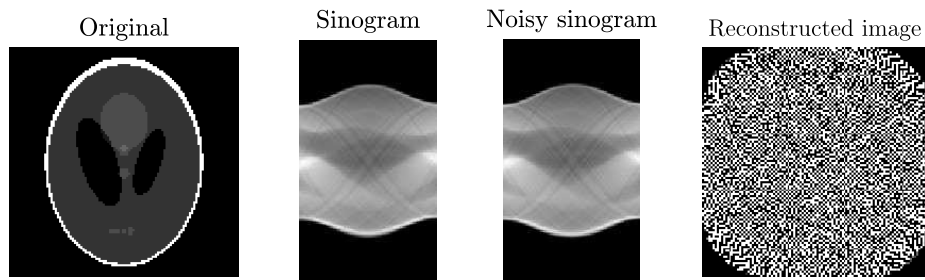


Figure 6.1: The sinogram b_{true} is the result of the Radon transform applied to the original image x_{true} to which we add some artificial noise of magnitude 10^{-6} to obtain the noisy sinogram b . The right image shows the reconstruction $x^* = A^\dagger b$.

The Radon transform, both in its continuous and its discretized form, has rapidly decaying singular values in its tail [135]. The ill-conditioning in A amplifies the error e and renders the naive solution x^* useless. To tackle this issue we must turn to regularization.

6.1.2 Tikhonov regularization

There exist various different types of regularization, and we refer to [99, 122, 130, 132] for excellent discussions. In this chapter we focus on Tikhonov regularization, which is “*probably the most successful regularization method of all times*” according to Hansen [132].

When using Tikhonov regularization [122, 246, 247] (also known as ridge regression), the LS problem (6.1) is transformed to

$$x_\lambda = \operatorname{argmin}_{x \in \mathbb{R}^n} \{ \|Ax - b\|_2^2 + \lambda \|Lx\|_2^2 \}, \quad (6.2)$$

for a regularization parameter $\lambda > 0$. In the standard form $L = I$, and in the general form $L \in \mathbb{R}^{p \times n}$ is any operator [34]. The regularization operator L is usually chosen to be some discretized differential operator with physical relevance to the underlying problem [132]. The conditioning of the problem improves for larger λ and x tends to the solution of (6.1) as λ tends to zero. By adding the regularization term $\lambda \|Lx\|$, we force the solution to have a physically relevant norm. The regularization parameter dictates how much we penalize the solution (semi-)norm. The optimal regularization parameter λ^* minimizes the error [130]

$$\lambda^* = \min_{\lambda > 0} \|x_\lambda - x_{\text{true}}\|.$$

6.1.2.1 Selecting a regularization parameter

The optimal regularization parameter is usually unknown a priori and is to be determined in an ad hoc manner, for instance by considering an L-curve [132, 133], using the discrepancy principle [99, 132, 190, 214], or generalized cross-validation [113, 262]. Many such methods depend on the quantities

$$\|Ax_\lambda - b\|_2 \quad \text{and} \quad \|Lx_\lambda\|_2,$$

illustrating that we are balancing goodness-of-fit (small $\|Ax_\lambda - b\|_2$) with regularity (small $\|Lx_\lambda\|_2$). This balancing act is displayed graphically in an L-curve, which plots $\|Lx_\lambda\|_2$ versus $\|Ax_\lambda - b\|_2$ as a continuous function of λ in a log-log scale [133]. This graph often has a characteristic L-shape. We plot the L-curve and the singular value spectrum of the example CT problem in Figure 6.2.

The vertical part in the left part of the figure corresponds to solutions that are ‘undersmoothed’: the solution norm blows up without the fit improving. In the horizontal part solutions are ‘over-smoothed’, the solution no longer captures the trend in the data since the residual is large. Critically, the optimal regularization parameter is close to the regularization parameter in the corner of the L-curve [131]. In this particular example, there is a small range of regularization parameters that

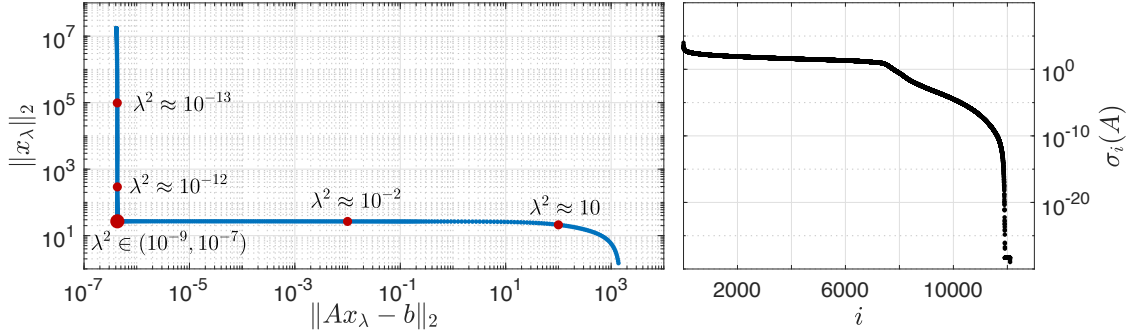


Figure 6.2: The L-curve associated to a discretized Radon transform A and its singular value spectrum. The left panel shows a log-log plot of the norm of the residual and the norm of the solution as a continuous function of the regularization parameter λ . Different values for λ are marked on the curve. The right panel shows the spectrum of A . The rapidly decaying singular value spectrum without clear gaps is typical for inverse problems [132]. The image and its (regularized) reconstructions are shown in Figures 6.1 and 6.3.

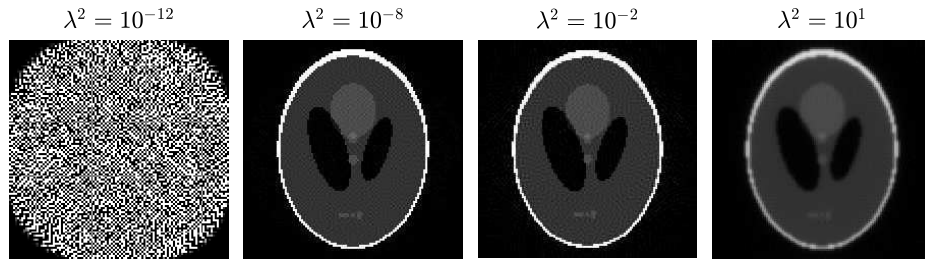


Figure 6.3: We try to reconstruct the original image using Tikhonov regularization (in its standard form). A small regularization parameter $\lambda^2 = 10^{-12}$ corresponds to undersmoothing, and if the regularization value is too large, we have oversmoothing and the lines become blurry. Note the correspondence to the values on the L-curve in Figure 6.2. The optimal regularization parameter is approximately $\lambda^2 = 10^{-7}$.

result in corner solutions. The reconstructions corresponding to this particular example and a few regularization parameters can be seen in Figure 6.3.

Naturally, the L-curve defines a criterion for regularization parameter selection. Choosing the ‘corner’ of the curve

$$(\log \|Ax_\lambda - b\|_2, \log \|Lx_\lambda\|_2)$$

corresponds to selecting the point that has maximum curvature [70, 136]. Computation of this point, however, is not computationally straightforward. An SVD of A (for $L = I$) or generalized SVD of (A, L) (for $L \neq I$) is necessary to determine it precisely, although one may approximate it using a bidiagonalization procedure [44]. Alternatively, with access to a multiple points on the curve we can interpolate it to find an appropriate parameter [133]. In this case we need to compute multiple solutions x_{λ_i} for regularization parameters $\lambda_1, \dots, \lambda_N$.

The Regińska criterion [220] is related to the L-curve; instead of maximising curvature, we aim to minimize $\|Ax_\lambda - b\| \|x_\lambda\|^\xi$ for some $\xi > 0$. This is still related to finding the corner of the L-curve. If multiple solutions x_{λ_i} or the SVD of A are not available, [26] proposed a fixed-point algorithm for selecting the minimizer of his function (see also [27, 28]).

If we have knowledge on the magnitude of the error present in the data, we can use the (Morozov)

discrepancy principle to select a regularization parameter [99, 190, 214]. Following [132], say $Ax_{\text{true}} = b_{\text{true}}$ holds for some exact solution x_{true} and exact data b_{true} . If we have data b with (measurement) errors but with some upper bound $\|b - b_{\text{true}}\|_2 \leq \delta_b$, then we should pick λ so that $\|Ax_\lambda - b\| = \delta_b$. Note that this would again involve computing solutions x_{λ_i} for various parameters λ_i , if a decomposition of A is not available.

Hansen (2010) notes that these types of methods to select a good regularization parameter “*are designed for problems where it is feasible to compute the SVD or compute Tikhonov solutions via the least squares formulation for several values of the regularization parameter. Many real problems, however, lead to large matrices where these factorization methods are infeasible with respect to computing time and/or storage*” [130]. We argue that owing to the randomized approach presented in this chapter, this argument no longer holds and we are able to apply sophisticated criteria to large-scale problems. This is the context of this work. The aim is to efficiently solve the same LS problem for multiple regularization parameters:

$$\text{find } x_{\lambda_i}^* = \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \{ \|Ax - b\|_2^2 + \lambda_i \|x\|_2^2 \}, \quad \text{for } i = 1, \dots, N. \quad (6.3)$$

6.1.2.2 Other regularization methods

There are many other methods for determining a regularization parameter. We distinguish between direct, iterative, and hybrid regularization approaches [108, 134]. The methods we discussed before are all part of the direct regularization approach. Tikhonov regularization is one class of direct regularization: fundamentally the problem (6.2) is formulated explicitly and, through some mechanisms (e.g. maximizing the curvature of the L-curve, the discrepancy principle), a regularization parameter λ is determined and the corresponding solution x_λ is given. Another class of direct regularization is to use the truncated SVD of A , where the rank of the truncation determines the level of regularization. Different ways to select the regularization parameter include generalized cross-validation (GCV) [262], the unbiased predictive risk estimator (UPRE) [261], and the quasi-optimality criterion [98, 131]. See [1, 5, 24, 25, 122, 206, 245], the work by Hämarik, Hanke, and Lukas, and [132, Ch. 7] for overviews of different methods to select a suitable parameter in direct regularization.

Another class of algorithms are iterative regularization methods. There is no explicit regularization parameter λ present in these methods, rather regularization occurs through early termination of an iterative method. Common choices include Landweber iterations, Kaczmarz’s method [155], and iterative Krylov subspace methods [132]. In iterative regularization, the solution x_k at any iteration k before convergence is considered to be regularized. The parameter that controls the amount of regularization is not the number of iterations. The phenomenon on which this is based is sometimes called semi-convergence [198]. The idea is that solutions first converge in the direction

of dominant singular values towards a solution close to x_{true} and only in later iterations converge to the naive solution $x^* = A^\dagger b$. One of the reasons iterative methods are preferred to methods that require full factorizations is that they often require only matrix-vector operations with A , instead of access to the entire matrix. This makes them suitable for large-scale problems where matrices are not explicitly stored [130].

There also exists a class of regularization algorithms that combine direct and iterative regularization. In these methods the small projected problem in each iteration is explicitly regularized. Recall that, at the k th iteration of, for instance LSQR, we solve the small LS problem (4.11). In a hybrid method, rather than solving (4.11), we solve

$$\min_{y \in \mathbb{R}^k} \{ \|\beta_1 e_1 - B_k y\|^2 + \lambda_k \|y\|^2 \},$$

where the regularization parameter λ_k is determined on-the-fly using criteria we previously discussed (but based on the small, projected problem). See [60, 108] for excellent reviews of hybrid Krylov methods for inverse problems.

In this chapter, we focus on direct regularization. We aim to show that, using randomization, it becomes feasible to apply Tikhonov regularization to large-scale problems.

6.1.3 Scope and focus of the algorithms

We set out to solve (6.3) for a sequence of parameters $\lambda_1, \dots, \lambda_N$. Our approach is to solve each sub-problem with an iterative sketch-and-precondition type solver, that is suitable for a variety of different problems. We discuss the sketch-and-precondition approach tailored for the problem in the next section. First, here, we discuss the range of problems for which this approach is useful.

We distinguish between two types of spectra of the data A . Firstly, a general context in which A is close to full-rank. That is, the singular values of A are not rapidly decreasing, or, more likely for inverse problems, just a few singular values are small. Secondly, we consider a context where A is approximately low-rank and has rapidly decaying singular values. In the context of a Tikhonov regularized LS problem, we then say that (6.2) has small statistical dimension (see Definition 6.1). We are able to exploit a small statistical dimension to reduce the total computational cost. In fact, if the statistical dimension is very small the problem need not be highly over- or underdetermined (which is the usual setting for randomized algorithms for LS). If the statistical dimension is much smaller than $\min(m, n)$, we can even solve square problems.

Any algorithm we present here for the highly overdetermined case ($m \gg n$) is trivially extendable to the highly underdetermined case ($n \ll m$, see Section 6.2.5). For full generality, we consider $L = I$ and $L \neq I$ separately using SVD- and GSVD-based techniques, respectively. The case $L \neq I$ is discussed in Section 6.4.1.

Although we show in Chapter 4 and Chapter 5 that this framework is not numerically stable, even with the inclusion of the sketch-and-solve initialization, we argue that it is sufficiently stable for practical applications in the regularized context. By including a regularization parameter, we are implicitly filtering out singular vectors corresponding to small singular values. As a result, considering the discussion in Section 4.2.4, it is not necessary to have a backward error smaller than the errors in the data.

6.1.4 Sketch-and-precondition for Tikhonov regularization

Building on the vast body of previous work on randomized numerical linear algebra for least squares problems, we leverage existing knowledge on sketch-and-precondition techniques (see Section 2.3.2) to design efficient solvers for (6.3). The algorithms we present aim to exploit the structure present in the regularized problem to efficiently update preconditioners. The overarching idea is that a decomposition of a sketch of A allows us to decompose, in some sense, the entire problem. As a result we can recycle certain computed quantities and easily update them for new regularization parameters.

We first briefly discuss the sketch-and-precondition framework and how (6.2) must be framed to allow for this randomized approach. We focus on the standard case with $L = I$ for now, but return to the general formulation in Section 6.4.1. Recall that in the standard LS problem (6.1), a sketch-and-precondition algorithm does the following. One draws a random embedding matrix $S \in \mathbb{R}^{s \times m}$, where $n < s \ll m$ and applies it to A to obtain the sketch SA . Next, this smaller matrix SA is decomposed, usually into a QR decomposition $SA = QR$. The inverse of the upper triangular factor is then used as a preconditioner in an iterative solver ($P = R^{-1}$), such as LSQR [205]. Additionally, through the QR decomposition of SA and the sketch Sb one obtains a good initial guess for the iterative solver with the sketch-and-solve solution [222], as discussed in detail in Chapter 5. If S is a sufficiently good embedding for the column space of A , then AR^{-1} will be well-conditioned in the sense that $\kappa_2(AR^{-1}) = \mathcal{O}(1)$. As a result, the iterative solver that is used will converge rapidly.

We transform the regularized problem (6.2) to standard LS formulation to be able to extend the sketch-and-precondition framework to this context. Regardless of the dimensions of A , (6.2) is equivalent to

$$\min_{x \in \mathbb{R}^n} \|Bx - \hat{b}\|_2^2, \quad B = \begin{bmatrix} A \\ \sqrt{\lambda}I_n \end{bmatrix}, \quad \hat{b} = \begin{bmatrix} b \\ 0 \end{bmatrix}, \quad (6.4)$$

where $B \in \mathbb{R}^{(m+n) \times n}$ and $\hat{b} \in \mathbb{R}^{m+n}$. For the underdetermined problem we can equivalently to (6.2)

find the minimum-norm solution to the problem [62]

$$\min_{x \in \mathbb{R}^n, r \in \mathbb{R}^m} \left\| \begin{bmatrix} x \\ r \end{bmatrix} \right\| \quad \text{s.t.} \quad D \begin{bmatrix} x \\ r \end{bmatrix} = b, \quad D = \begin{bmatrix} \lambda^{-1/2} A & I_m \end{bmatrix}. \quad (6.5)$$

The solution x_λ^* to (6.2) is then given by the top n coordinates x of the solution to (6.5).¹

A naive randomized approach to solving (6.2) for multiple regularization parameters would be to solve the problem from scratch for each λ_i . The algorithms we present in this work use ‘partly-exact’ sketches instead. The idea is to sketch A only once and reuse this sketch for each of the λ_i , while leaving the scaled identity part ‘exact’. Reusing sketches in Tikhonov regularization was similarly suggested in LSRN [187], and follows naturally in various works [54, 59, 268]. Avron et al. [13] introduced the name ‘partly-exact’ sketching. In our context, a partly-exact sketch of the matrix in the overdetermined problem (6.4), would be

$$\hat{S}B = \begin{bmatrix} S & 0 \\ 0 & I_n \end{bmatrix} \begin{bmatrix} A \\ \sqrt{\lambda} I_n \end{bmatrix} = \begin{bmatrix} SA \\ \sqrt{\lambda} I_n \end{bmatrix}, \quad (6.6)$$

for an embedding matrix $S \in \mathbb{R}^{s \times m}$, $s \ll m$ (as opposed to sketching B with an $s \times (m + n)$ embedding matrix). We can equivalently define this for the underdetermined case.

As explained before, if we were to compute the QR factorization $\hat{S}B = QR$, BR^{-1} would be a well-conditioned matrix. A fundamental idea throughout this work is that we can equivalently state R^{-1} is a good preconditioner provided that

$$R^T R = (SA)^T (SA) + \lambda I_n. \quad (6.7)$$

This point of view allows us to rephrase the sketch-and-precondition problem to finding any matrix R that satisfies (6.7) and may be cheaply inverted. In the next section we discuss broadly our approach to this.

6.1.5 Our contributions

We introduce a variety of algorithms within the sketch-and-precondition framework to solve (6.3). Each algorithm uses partly-exact sketching, combined with a particular way of choosing an R that satisfies (6.7). The overarching idea is to choose a smart decomposition that allows for a computationally inexpensive update of the preconditioner. As a result, the work involved in solving (6.3)

¹This formulation for $L \neq I$ requires L to be non-singular. We do not consider the weighted underdetermined case for a singular or non-square matrix L , as it does not lend itself to the sketch-and-precondition framework. This is further discussed in Section 6.4.1.

will (for our best case complexity algorithms) be the cost of a single sketch, the cost of a single decomposition, and then the cost of N instances of the iterative solver.

6.1.5.1 General context

For the general context, we propose three algorithms based on different decompositions: the SVD, the eigendecomposition and the Cholesky decomposition. The SVD-based algorithm has excellent complexity and allows for an easy update of an initial guess for LSQR, which is essential for rapid and stable convergence (see Section 5.2.1). Its complexity is upper bounded by $\mathcal{O}(mn \log m + n^3 \log(n) + Nmn \log(1/\varepsilon))$ to solve (6.3) for N regularization parameters to relative accuracy ε . The term $\mathcal{O}(mn \log(m))$ is an upper bound for the cost to sketch A to SA . Computing the SVD of SA costs $\mathcal{O}(n^3 \log(n))$ operations assuming an embedding dimension of $n \log(n)$, and finally we have $\mathcal{O}(mn \log(1/\varepsilon))$ cost N times to iteratively solve for all parameters.

The SVD- and eigendecomposition-based algorithms are similar in nature and have the same computational complexity. The difference is that we compute the eigendecomposition of $(SA)^T(SA)$ instead of the SVD of SA , yet both procedures cost $\mathcal{O}(n^3 \log(n))$. The advantage of the SVD-based algorithm is that a sketch-and-solve initialization naturally follows, whereas a stable initialization is not readily available for the eigendecomposition-based approach. It can still be competitive, because our experiments show the computation of the eigendecomposition of the Gram matrix of the sketch can be quicker than computation of the SVD of SA .

The Cholesky decomposition-based algorithm has a much worse complexity than the two previous algorithms. Its complexity is upper bounded by $\mathcal{O}(mn \log m + N(n^3 + mn \log(1/\varepsilon)))$. The Nn^3 complexity comes from having to compute N Cholesky decompositions of $n \times n$ matrices.

However, if there are just a few regularization parameters, it may still outperform alternatives with better complexity due to its low start-up cost and the speed with which the Cholesky decomposition is computed.

6.1.5.2 A context with low-statistical dimension

We furthermore introduce an algorithm that can efficiently tackle the situation where the matrix A has rapidly decaying singular values, regardless of the dimension of the problem. Specifically, even if A is square or nearly-square, then the proposed algorithms might be used; provided that the spectrum of A decays sufficiently rapidly. The quantity of interest in this case, in the context of Tikhonov regularization, is the *statistical dimension* (also called degrees of freedom).

Definition 6.1 (Statistical dimension). *For $\lambda \geq 0$ and a rank- r matrix A with singular values $\sigma_1(A), \dots, \sigma_r(A)$, the quantity*

$$sd_\lambda(A) = \sum_{i=1}^r \frac{1}{1 + \frac{\lambda}{\sigma_i(A)^2}} \quad (6.8)$$

is the statistical dimension.

One can loosely interpret the statistical dimension as the number of singular values of A that are larger than $\sqrt{\lambda}$. The quantity is bounded above by the rank of A . The statistical dimension has received significant attention in the randomized NLA literature as recent work [12, 13, 59, 63] shows the sketch size can be of the same order as the statistical dimension. The theory on embedding matrices presented in Section 2.1.3 suggests that the optimal embedding dimension is $\mathcal{O}(r)$, where r is the rank of A . In the context of Tikhonov regularization, and thus in the presence of the parameter λ , we can instead consider an optimal dimension proportional to $\text{sd}_\lambda(A)$. For a matrix A with rapidly decaying singular values, we could form a sketch with sketch dimension smaller than the rank of A .

Inspired by the Kernel Ridge Regression (KRR) solver [12], we propose a scheme that results in a preconditioner based on a small sketch $SA \in \mathbb{R}^{s \times n}$, where $s < n$. In at most $\mathcal{O}(mn \log(n) + \min(m, n)s^2)$ operations we find a preconditioner that is suitable for any value of λ . This is possible by computing the SVD of SA and then using the Woodbury matrix identity and the (truncated) SVD factors of SA to compute a preconditioner with low-rank structure that can be applied cheaply. Notably, this sketch-and-precondition-based algorithm is suitable for problems of any dimension, even square. The only requirement is a very rapidly decaying singular value spectrum. This makes the scheme particularly useful for inverse problems.

Although our work in (approximately) low-rank LS problems is close in spirit to [12] (translated to a LS context) and [59], these works are concerned with solving the normal equations, thus involving the Gram matrix $A^T A$ (or AA^T). This results in squaring the condition number (as $\lambda \rightarrow 0$). It is well-known for LS problems that solutions obtained via the normal equation are unstable, whereas better stability can be recovered if one works directly with the matrix A [140, Ch. 20] (see Section 4.2.5). Our approach does this, and finds a preconditioner for A rather than its Gram matrix. Furthermore, we avoid $\mathcal{O}(\min(m, n)^3)$ work and instead only perform $\mathcal{O}(\min(m, n)\text{sd}_\lambda(A)^2)$ operations.

6.1.5.3 Convergence results

We finally present convergence results for all algorithms. The theoretical results are based on the structural conditions presented in Chowdhury et al. (2018) [59]. In particular, for the overdetermined case in a general context, we would be interested in bounding the singular values of SU_A from either side, where S is an embedding matrix and U_A are the left singular vectors of A . Similarly, for the low statistical dimension context we would like our embedding matrix S to satisfy

$$\|\Sigma_\lambda U_A^T S^T S U_A \Sigma_\lambda - \Sigma_\lambda^2\|_2 \leq \epsilon, \quad \Sigma_\lambda = \Sigma_A(\Sigma_A^2 + \lambda I)^{-1/2},$$

for some $0 < \epsilon < 1$, where Σ_A is the diagonal matrix of singular values of A . This is further expanded upon in Sections 6.2.4 and 6.3.4.

6.1.6 Related work

The regularized system (6.2) can be solved with direct methods based on the SVD in $\mathcal{O}(mn \min(m, n))$ operations. For very large scale systems, this cost is prohibitively large and we must turn to iterative solvers and/or randomized methods. We have discussed various approaches in Section 6.1.2.2.

Recently, a new class of randomized algorithms for LS that does not require any decompositions was introduced. These are iterative methods in which sketching is used in each iteration, most notably Iterative Hessian Sketching (IHS), proposed in [216], which uses a sketched Hessian and inexact inversion. Extensions of this work include Accelerated IHS [267], Momentum-IHS [200, 201], and Polyak-IHS [158]. Wang et al. discuss more general iterative sketching methods for regularized methods (that might require decompositions) in [268], in particular from a statistical perspective. Of works in this category, [59] is closest in spirit to our work. The authors design an iterative algorithm for the underdetermined problem which only requires a sketch (based on leverage scores) of size proportional to the statistical dimension. A difference is that it does require a $\min(m, n) \times \min(m, n)$ exact inversion.

There are also works in Kernel Ridge Regression (KRR) that are closely related, most notably [12] which we discussed above. Chen et al. [56] discuss a general framework for sketch-and-solve in KRR and [13] analyses errors in this context. In [96], the authors propose a sketch-and-solve algorithm based on leverage score sampling and in [283] a sketch-and-solve algorithm based on a different definition of the statistical dimension is introduced. An iterative sketching algorithm is proposed in [106].

There are various randomized approaches to regularized problems that use the randomized SVD [150, 172, 277] or randomized generalized SVD [226, 258, 273, 278]. Other methods have a sketch-and-solve flavour [170].

6.2 Randomized sketch-and-precondition algorithms for Tikhonov regularization in general settings

We start with the exposition of a scheme that fits a general spectral regime: the singular values of A do not decay very rapidly and as a result, the statistical dimension $\text{sd}_\lambda(A)$ of the problem is not much smaller than $\min(m, n)$. The algorithmic consequence is that the necessary sketch dimension s is proportional to $\min(m, n)$: that is, $s > \min(m, n)$ and SA is a matrix of full column rank.

We discuss three algorithms: 1) SVD-based, 2) eigendecomposition-based, and 3) Cholesky decomposition-based. In this section we treat the overdetermined case for standard Tikhonov regularization ($L = I$) and discuss the problem setting, the algorithms, and the convergence analysis. All algorithms and theory can be trivially extended to the underdetermined setting.

Let us start with specifying the problem. Let $A \in \mathbb{R}^{m \times n}$ with $m \gg n$. We aim to find preconditioners for the system (6.4) for multiple regularization parameters $\lambda_1, \dots, \lambda_N$. In particular, by denoting

$$B_i = \begin{bmatrix} A \\ \sqrt{\lambda_i} I_n \end{bmatrix}, \quad i = 1, \dots, N,$$

we find preconditioners $P_1, \dots, P_N$ such that $\kappa(B_i P_i) = \mathcal{O}(1)$. Additionally, we ideally compute initial guesses $x_{\lambda_i}^{(0)}$, $i = 1, \dots, N$ such that

$$\left\| \begin{bmatrix} A \\ \sqrt{\lambda_i} I_n \end{bmatrix} x_0^{(i)} - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|_2 \leq (1 + \delta) \left\| \begin{bmatrix} A \\ \sqrt{\lambda_i} I_n \end{bmatrix} x_{\lambda_i}^* - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|_2,$$

for a small constant $\delta > 0$ with high probability. Here, $x_{\lambda_i}^*$ denotes the optimal residual solution to the regularized problem with λ_i . One choice that satisfies this condition is the sketch-and-solve solution, as suggested in [222] and discussed in Sections 2.3.1 and 5.2.1. Our eigendecomposition-based and Cholesky-based algorithms will include different initializations.

Having obtained a preconditioner P_i and an initial guess $x_0^{(i)}$, the solution to (6.4) is obtained with a preconditioned iterative solver. As a default we use LSQR [205].

6.2.1 An SVD-based approach

Let $S \in \mathbb{R}^{s \times m}$, $n < s \ll m$, be an embedding matrix and let SA be the sketch of A . Suppose we have an SVD of $SA = U \Sigma V^T$, where $U \in \mathbb{R}^{s \times n}$ and $V \in \mathbb{R}^{n \times n}$ have orthogonal columns and $\Sigma \in \mathbb{R}^{s \times s}$ is diagonal. For the ease of notation, fix a regularization parameter $\lambda > 0$. Under the assumption that S is a good embedding matrix for A , we know that R^{-1} is a good preconditioner provided that

$$R^T R = (SA)^T (SA) + \lambda I_n.$$

Note that this is equivalent to

$$R^T R = V \Sigma^2 V^T + \lambda I_n = V (\Sigma^2 + \lambda I_n) V^T,$$

which is satisfied for $R = (\Sigma^2 + \lambda I_n)^{1/2} V^T$ with inverse

$$R^{-1} = V(\Sigma^2 + \lambda I_n)^{-1/2} =: P.$$

The preconditioner P can be applied to a vector $x \in \mathbb{R}^n$ in $2n(n+1)$ operations. More importantly, once V and Σ are computed, one can easily update the preconditioner for new values of λ .

6.2.1.1 A sketch-and-solve initial guess

The sketch-and-solve solution to (6.4) given a partly-exact sketch is the solution to

$$x_0 = \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \left\| \begin{bmatrix} SA \\ \sqrt{\lambda} I_n \end{bmatrix} x - \begin{bmatrix} Sb \\ 0 \end{bmatrix} \right\|_2. \quad (6.9)$$

Given the SVD $SA = U\Sigma V^T$, one can show that the SVD of the partly-exact sketch is given by

$$\begin{bmatrix} SA \\ \sqrt{\lambda} I_n \end{bmatrix} = \begin{bmatrix} U\Sigma(\Sigma^2 + \lambda I_n)^{-1/2} \\ \sqrt{\lambda} V(\Sigma^2 + \lambda I_n)^{-1/2} \end{bmatrix} (\Sigma^2 + \lambda I_n)^{1/2} V^T.$$

As a result, the sketch-and-solve initial guess is

$$x_0 = \begin{bmatrix} SA \\ \sqrt{\lambda} I_n \end{bmatrix}^\dagger \begin{bmatrix} Sb \\ 0 \end{bmatrix} = V\Sigma(\Sigma^2 + \lambda I_n)^{-1} U^T Sb. \quad (6.10)$$

6.2.1.2 Algorithm

We display the final algorithm in pseudocode in Algorithm 13. The full complexity is $\mathcal{O}(mn \log m + sn^2 + Nmn \log(1/\varepsilon))$ to solve (6.3) to relative accuracy ε . We note that solving the N problems in the for loop can trivially be parallelized.

6.2.2 An eigendecomposition-based approach

Note that for the actual preconditioners, only the right singular vectors and the singular values of the sketch are necessary. As a result we, can alternatively compute the eigendecomposition of the Gram matrix $(SA)^T(SA) = V\Lambda V^T$, where $\Lambda = \Sigma^2$. This approach might sometimes be preferred to an SVD-approach as computing the Gram matrix and its eigendecomposition can be quicker than computing the SVD, despite both routines having the same complexity. The preconditioner is then of the form

$$P_i = V(\Lambda + \lambda_i I_n)^{-1/2}.$$

Algorithm 13 An SVD-based sketch-and-precondition algorithm for Tikhonov regularized over-terminated LS. Given an $m \times n$ matrix A , $m > n$, a set of regularization parameters $\lambda_1, \lambda_2, \dots, \lambda_N$, and a sampling parameter s ($n < s < m$), this algorithm computes approximate solutions $x_{\lambda_i}^*$ to (6.3).

- 1: Draw an $s \times m$ random embedding matrix S .
- 2: Compute $Y = SA$ and $\tilde{b} = Sb$.
- 3: Compute the economy-sized SVD $[U, \Sigma, V] = \text{svd}(Y, \text{'econ'})$.
- 4: Compute $y_0 = U^T \tilde{b}$.
- 5: **for** $i = 1, 2, \dots, N$ **do**
- 6: Compute $x_{\lambda_i}^{(0)} = V \Sigma (\Sigma^2 + \lambda_i I_n)^{-1} y_0$.
- 7: Solve the following system

$$x_{\lambda_i}^* = \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \left\| \begin{bmatrix} A \\ \sqrt{\lambda_i} I_n \end{bmatrix} x - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|^2 \quad (6.11)$$

using LSQR with preconditioner

$$P_i = V(\Sigma^2 + \lambda_i I_n)^{-1/2},$$

and with initial guess $x_{\lambda_i}^{(0)}$.

8: **end for**

We might consider an initial guess x_0 of the form

$$(A^T S^T S A) x_0 = A^T b,$$

which can be seen as a sketched version of the normal equations. The solution to this problem using the eigendecomposition of $(SA)^T(SA)$ is

$$x_0 = V(\Lambda + \lambda I)^{-1} V^T A^T b.$$

We find that for small values of λ and highly ill-conditioned problems, the diagonal matrix $\Lambda + \lambda I$ is so ill-conditioned that the norm of the solution x_0 blows up. These instabilities can harm the convergence of the iterative solver. As a work-around, we suggest to first compute the solution corresponding to the largest λ , i.e. the best-conditioned problem, and using this solution as an initial guess for the next largest value of λ , and continuing this process, initializing a new sub-problem with the previously found solution.

6.2.3 A Cholesky-based approach

We finally propose an algorithm that is most useful when there are just a few regularization parameters we would like to compute solutions for. We propose to compute the Cholesky factorization of the Gram matrix combined with the regularization term:

$$R^T R = (SA)^T(SA) + \lambda I.$$

The preconditioner is then given by $P = R^{-1}$. The only preprocessing step is computing the Gram matrix of the sketch. For each λ_i we will consequently have the cost of computing the Cholesky factor of an $n \times n$ matrix, which has complexity $\mathcal{O}(n^3)$ with a small constant.

Even though this approach has $\mathcal{O}(Nn^3)$ complexity, as opposed to $\mathcal{O}(n^3)$ in the previous sections, it can still be competitive because the preprocessing steps are much cheaper and the Cholesky decomposition is very fast to compute. Furthermore, computing the decomposition is possible for almost all regularization parameters λ , as the requirement is that the matrix $(SA)^T(SA) + \lambda I$ is numerically full rank, so R can be computed without breakdown [140, Ch. 10].

A natural initial guess is again available from the equation $(AS^TSA + \lambda I)x_0 = A^Tb$. We then need to complete two triangular solves to compute x_0 : firstly, $R^Ty_0 = A^Tb$ and secondly, $Rx_0 = y_0$. This approach faces similar instabilities as the initial guess arising from the eigendecomposition, discussed in the previous section. We suggest it is only used for large values of λ .

6.2.4 Convergence analysis

Throughout this chapter, we propose convergence proofs based on the structural conditions presented in [59]. These conditions are on the quality of the embedding matrix, in particular how well the relevant information in A is preserved. For the case $s \geq n$, the relevant condition is

$$\|U_A^T S^T S U_A - I_n\|_2 \leq \tilde{\epsilon},$$

for some $0 < \tilde{\epsilon} < 1$. Here, U_A is the matrix consisting of the left singular vectors of A . We use the closely related condition on the singular values of SU_A :

$$1 - \epsilon \leq \sigma_{\min}(SU_A) \leq \sigma_{\max}(SU_A) \leq 1 + \epsilon.$$

It results in the following lemma on the condition number of BR^{-1} . It is important to note that Algorithm 13, and the other algorithms proposed in this section, output preconditioners $P_i = R_i^{-1}$ that satisfy (in exact arithmetic)

$$R_i^T R_i = A^T S^T S A + \lambda_i I_n.$$

Lemma 6.2. *Let $A \in \mathbb{R}^{m \times n}$, $m > n$ have SVD $A = U_A \Sigma_A V_A^T$, where $U_A \in \mathbb{R}^{m \times n}$, and let $\lambda > 0$. Let $S \in \mathbb{R}^{s \times m}$, $m > s \geq n$, be a matrix such that $1 - \epsilon < \sigma_{\min}(SU_A) < \sigma_{\max}(SU_A) < 1 + \epsilon$ for some $0 < \epsilon < 1$. Suppose $R \in \mathbb{R}^{n \times n}$ is an invertible matrix such that*

$$R^T R = A^T S^T S A + \lambda I_n.$$

Then

$$\kappa_2(BR^{-1}) \leq \frac{1 + \epsilon}{1 - \epsilon},$$

where

$$B = \begin{bmatrix} A \\ \sqrt{\lambda} I_n \end{bmatrix}. \quad (6.12)$$

Proof. The proof is inspired by that of [59, Lem. 15]. We write $S \preceq T$ for Hermitian matrices S and T to imply that $S - T$ is negative semi-definite. We start from the fact that $\sigma_i(SU_A)^2 = \lambda_i(U_A^T S^T S U_A)$, where λ_i denotes the i th greatest eigenvalue. Then

$$\sigma_{\min}(SU_A)^2 I \preceq U_A^T S^T S U_A \preceq \sigma_{\max}(SU_A)^2 I.$$

Multiply this expression by $V_A \Sigma_A$ from the left and $(V_A \Sigma_A)^T$ from the right to find

$$\sigma_{\min}(SU_A)^2 A^T A \preceq A^T S^T S A \preceq \sigma_{\max}(SU_A)^2 A^T A, \quad (6.13)$$

which is equivalent to

$$\sigma_{\min}(SU_A)^2 A^T A + \lambda I \preceq A^T S^T S A + \lambda I \preceq \sigma_{\max}(SU_A)^2 A^T A + \lambda I.$$

Under the assumption that $\sigma_{\min}(SU_A) < 1 < \sigma_{\max}(SU_A)$ this implies

$$\sigma_{\min}(SU_A)^2 (A^T A + \lambda I) \preceq A^T S^T S A + \lambda I \preceq \sigma_{\max}(SU_A)^2 (A^T A + \lambda I).$$

We can rephrase this as

$$\sigma_{\min}(SU_A)^2 B^T B \preceq R^T R \preceq \sigma_{\max}(SU_A)^2 B^T B.$$

It follows that

$$\sigma_{\max}(SU_A)^{-2} I \preceq (BR^{-1})^T (BR^{-1}) \preceq \sigma_{\min}(SU_A)^{-2} I.$$

This finally implies

$$\sigma_{\max}(SU_A)^{-1} \leq \sigma_{\min}(BR^{-1}) \leq \sigma_{\max}(BR^{-1}) \leq \sigma_{\min}(SU_A)^{-1},$$

so we have

$$\kappa_2(BR^{-1}) \leq \kappa_2(SU_A),$$

as required. $\square$

It is well-known that LSQR applied to a well-conditioned LS problem converges geometrically with respect to the number of iterations. It follows from the above result that we obtain an ε -accuracy solution for (6.11) with $\mathcal{O}(\log(1/\varepsilon))$ iterations in LSQR. Note that ε is different from ϵ in the condition of Lemma 6.2. In practise, ϵ is usually not much smaller than 1, say 0.5, whereas ε can be much smaller.

6.2.5 Adaptation to underdetermined problems

We demonstrate how the SVD-based algorithm can be extended to the underdetermined case. The eigendecomposition- and Cholesky-based algorithms follow trivially, as does the convergence result Lemma 6.2. Essentially, the previously proposed algorithms will be applied to A^T as opposed to A . The main difference is that we now find the minimum-norm solution to a consistent system, as opposed to a solution to an (inconsistent) overdetermined problem.

Let us first summarise the problem set-up: we aim to find preconditioners R_i for

$$D_i = \begin{bmatrix} \lambda_i^{-1/2} A & I_m \end{bmatrix}, \quad i = 1, \dots, N$$

such that $\kappa(R_i^{-T}D_i) = \mathcal{O}(1)$. We then use an iterative solver to find solution to [187]

$$\begin{bmatrix} x_{\lambda_i}^* \\ y^* \end{bmatrix} = \underset{x \in \mathbb{R}^n, y \in \mathbb{R}^m}{\operatorname{argmin}} \left\| \begin{bmatrix} x \\ y \end{bmatrix} \right\|_2^2 \text{ such that } R_i^{-T}D_i \begin{bmatrix} x \\ y \end{bmatrix} = R_i^{-T}b. \quad (6.14)$$

Luckily, LSQR converges to the minimum-norm solution when presented with the linear system $R_i^{-1}D_i z = R_i^{-T}b$ naturally [187].

We drop the subscripts and define $D = [\lambda^{-1/2}A \ I_m]$. Since A is now a short matrix, we sketch it from the right instead of the left with an embedding matrix $S \in \mathbb{R}^{n \times s}$, where $m \leq s \ll n$. Translating (6.7) to this context, we find that R^{-1} is a good preconditioner to D if

$$R^T R = \lambda^{-1}(AS)(AS)^T + I_m.$$

Suppose we have the SVD of the sketch $SA = U\Sigma V^T$. We can repeat a similar argument as before to observe

$$R^T R = U(\lambda^{-1}\Sigma^2 + I_m)U^T,$$

and thus $R^{-1} = U(\lambda^{-1}\Sigma + I_m)^{-1/2}$ is a good preconditioner.

We can prove a result analogous to Lemma 6.2, but with the condition on the sketching matrix replaced by $(1 - \varepsilon) < \sigma_{\min}(S^T V_A) \leq \sigma_{\max}(S^T V_A) < 1 + \varepsilon$, where V_A are the right singular vectors of A .

6.3 Randomized Tikhonov regularization for problems with small statistical dimension

A regularized linear least squares problem with an approximately low-rank data matrix, i.e. one for which A has rapidly decreasing singular values, may be solved in a way that takes advantage of this particular structure. Specifically, we may sketch the matrix with an embedding matrix that is smaller than either of the original dimensions. In this section we discuss an algorithm that computes a preconditioner for this context. Recall that we no longer desire $m \ll n$ or $m \gg n$, but rather just that the statistical dimension is much smaller than $\min(m, n)$. This means the approach also works for (nearly) square systems.

We take inspiration from Avron et al. (2017) [12], which discusses a sketch-and-precondition algorithm for kernel ridge regression. The authors use the Woodbury matrix identity combined with a Cholesky decomposition of $(SA)(SA)^T + \lambda I_s$ to obtain a preconditioner for the normal equations mathematically equivalent to $P = (SA)^T(SA) + \lambda I_n$. That is, the matrix they are interested in is P whereas we are interested in obtaining a matrix R such that $P = R^T R$. This is an important difference that avoids instabilities caused by invoking the normal equations (and thus computing the Gram matrix of A), as opposed to preconditioning B (in (6.28)) directly and solving the LS problem. Our experiments illustrate the improved stability.

In this section we assume A to be (approximately) low-rank, with decaying singular values such that $\operatorname{sd}_\lambda(A)$ as defined in Definition 6.1, is (much) smaller than $\min(m, n)$. Again, we consider a sequence of regularization parameters $\lambda_1 \geq \dots \geq \lambda_N$ for which we aim to solve (6.3) with an iterative solver. We propose an algorithm that finds a sequence of preconditioners for either B_i and

D_i , where

$$B_i = \begin{bmatrix} A \\ \sqrt{\lambda_i} I_n \end{bmatrix}, \quad D_i = \begin{bmatrix} A & \sqrt{\lambda_i} I_m \end{bmatrix}, \quad i = 1, \dots, N,$$

in $\mathcal{O}(mn \log(\max(m, n)) + \min(m, n) \text{sd}_\lambda(A)^2)$ operations. The preconditioners can be applied to a vector in $\mathcal{O}(\min(m, n) \text{sd}_\lambda(A))$ operations. We furthermore discuss an easy update of the sketch-and-solve initial guess.

Throughout this section we assume $m \geq n$, but all algorithms and theory may trivially be extended to the underdetermined case similar to Section 6.2.5.

6.3.1 A preconditioner with low-rank structure

Let $A \in \mathbb{R}^{m \times n}$ where $m \geq n$, and fix a λ (for the sake of the argument) such that $\text{sd}_\lambda(A) \ll n$. We look for a preconditioner for $B = [A^T \sqrt{\lambda} I_n]^T$. Let $S \in \mathbb{R}^{s \times m}$ be an embedding with $s \ll n \leq m$ and $s = \mathcal{O}(\text{sd}_\lambda(A))$. We use the (short and fat) sketch $Y = SA \in \mathbb{R}^{s \times n}$. Our aim is to find R such that

$$R^T R = Y^T Y + \lambda I_n, \quad (6.15)$$

as we will show this results in a good preconditioner for B . First take the SVD of Y :

$$Y = U \Sigma V^T, \quad U \in \mathbb{R}^{s \times s}, \quad \Sigma \in \mathbb{R}^{s \times s}, \quad V \in \mathbb{R}^{n \times s}.$$

Then

$$Y^T Y + \lambda I_n = V \Sigma^2 V^T + \lambda I_n = \begin{bmatrix} V & V_\perp \end{bmatrix} \begin{bmatrix} \Sigma^2 + \lambda I_s & 0 \\ 0 & \lambda I_{n-s} \end{bmatrix} \begin{bmatrix} V^T \\ V_\perp^T \end{bmatrix},$$

where $V_\perp \in \mathbb{R}^{n \times (n-s)}$ is such that $\begin{bmatrix} V & V_\perp \end{bmatrix}$ is a square orthogonal matrix. If we let

$$R = \begin{bmatrix} V & V_\perp \end{bmatrix} \begin{bmatrix} (\Sigma^2 + \lambda I_n)^{1/2} & 0 \\ 0 & \sqrt{\lambda} I_{n-s} \end{bmatrix} \begin{bmatrix} V^T \\ V_\perp^T \end{bmatrix}, \quad (6.16)$$

then $R^T R = Y^T Y + \lambda I_n$. Note we can reformulate R to not include $V_\perp$ as follows

$$\begin{aligned} R &= V(\Sigma^2 + \lambda I_n)^{1/2} V^T + \sqrt{\lambda} V_\perp V_\perp^T \\ &= V(\Sigma^2 + \lambda I_n)^{1/2} V^T + \sqrt{\lambda}(I_n - V V^T) \\ &= \sqrt{\lambda}(V F V^T + I_n), \end{aligned}$$

where

$$F = (\lambda^{-1/2} \Sigma^2 + I_s)^{1/2} - I_s$$

is a diagonal matrix. We are interested in the inverse of R . By using the Sherman-Morrison-Woodbury [233, 274] formula we have

$$(V F V^T + I_n)^{-1} = I_n - V(F^{-1} + V^T V)^{-1} V^T = I_n - V G V^T,$$

where $G = (F^{-1} + I_n)^{-1}$ is a diagonal matrix with elements

$$G_{ii} = 1 - \sqrt{\frac{1}{1 + \frac{\sigma_i(Y)^2}{\lambda}}}.$$

Finally,

$$R^{-1} = \lambda^{-1/2}(I_n - VGV^T), \quad (6.17)$$

where $V \in \mathbb{R}^{n \times s}$ and $G \in \mathbb{R}^{s \times s}$. This matrix, although not triangular, has a low-rank structure that allows it to be applied quickly. Most importantly, it is such that (6.15) holds while we avoided the decomposition of an $n \times n$ matrix.

6.3.2 A sketch-and-solve initial guess

The initial guess advised by [222] and shown to be important for numerical stability in Chapter 5 is the sketch-and-solve solution to (6.2). That is,

$$x_0 = \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \left\| \begin{bmatrix} SA \\ \sqrt{\lambda}I_n \end{bmatrix} x - \begin{bmatrix} Sb \\ 0 \end{bmatrix} \right\|. \quad (6.18)$$

We would like to express the solution to (6.18) in terms of the SVD of only the sketch $SA = U\Sigma V^T$ as before. One can show, using similar manipulations as above, that the SVD of the partly-exact sketch is given by

$$\begin{bmatrix} SA \\ \sqrt{\lambda}I_n \end{bmatrix} = \begin{bmatrix} U\Sigma(\Sigma^2 + \lambda I_s)^{-1/2} & 0 \\ \sqrt{\lambda}V(\Sigma^2 + \lambda I_s)^{-1/2} & V_{\perp} \end{bmatrix} \begin{bmatrix} (\Sigma^2 + \lambda I_s)^{1/2} & 0 \\ 0 & \sqrt{\lambda}I_{n-s} \end{bmatrix} \begin{bmatrix} V & V_{\perp} \end{bmatrix}^T.$$

As a result, the solution to (6.18) is given by

$$x_0 = V\Sigma(\Sigma^2 + \lambda I_s)^{-1}U^T Sb, \quad (6.19)$$

and can be easily updated for various λ given the SVD of SA .

6.3.3 An algorithm exploiting small statistical dimension

We aim to solve (6.2) for a sequence of regularization parameters $\lambda_1 \geq \dots \geq \lambda_N$. The way the algorithms in Section 6.2 were designed allowed us to sketch the matrix A only once, and reuse the sketch for different values of λ_i . For problems with low statistical dimension it is slightly more difficult to do this, because we want our sketch size to depend on $\operatorname{sd}_{\lambda_i}(A)$ which in turn depends on λ_i . We propose an algorithm to bypass this problem.

As the statistical dimension increases as λ decreases, the largest statistical dimension corresponds to the smallest value of λ — which we assume is λ_N . Our algorithm sketches A once, with dimension corresponding to $\operatorname{sd}_{\lambda_N}(A)$, and computes the SVD of this sketch. For all other values of λ , we truncate the factors of the SVD and so use lower-rank matrices that correspond to $\operatorname{sd}_{\lambda_i}(A)$.

The resulting algorithm is presented in Algorithm 14. We are able to find a sequence of preconditioners in $\mathcal{O}(mn \log m + n \operatorname{sd}_{\lambda_i}(A)^2)$ operations. It can be applied to a vector $x \in \mathbb{R}^n$ by

computing

$$R^{-1}x = \lambda^{-1/2}(x - VSVx),$$

which requires $2(2ns + s^2 + n)$ operations — compared to n^2 for an upper triangular preconditioner. One of its main advantages is that we only need to compute a preconditioner once, and for any (sufficiently large) λ , one can truncate this without any additional computational work.

An important algorithmic consideration is the estimation of the statistical dimension for the various values of λ . This is discussed in the next section.

Algorithm 14 An SVD-based sketch-and-precondition algorithm for Tikhonov regularized overdetermined or square LS problems with small statistical dimension. Given an $m \times n$ matrix A , $m > n$, a set of regularization parameters $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_N$, estimates of $\text{sd}_{\lambda_i}(A)$ for $i = 1, \dots, N$, and an oversampling parameter α such that $\alpha \text{sd}_{\lambda_N}(A) \leq n$, this scheme computes solutions $x_{\lambda_i}^*$ to (6.3).

- 1: Set $s = \alpha \lceil \text{sd}_{\lambda_N}(A) \rceil$.
- 2: Draw an $s \times m$ random embedding matrix S .
- 3: Compute $Y = SA$ and Sb .
- 4: Compute the SVD $[U, \Sigma, V] = \text{svd}(Y, \text{'econ'})$.
- 5: **for** $i = 1, 2, \dots, N$ **do**
- 6: Set $s_i = \alpha \lceil \text{sd}_{\lambda_i}(A) \rceil$.
- 7: Truncate $U_i = U(:, 1 : s_i)$, $V_i = V(:, 1 : s_i)$, and $\Sigma_i = \Sigma(1 : s_i, 1 : s_i)$.
- 8: Compute $x_0^{(i)} = V_i(\Sigma_i^2 + \lambda_i I)^{-1} \Sigma_i U_i^T S b$
- 9: Form a diagonal matrix G_i with elements

$$(G_i)_{jj} = 1 - \sqrt{\frac{1}{1 + \frac{(\Sigma_i)_{jj}^2}{\lambda}}}$$

- 10: Solve the system

$$x_{\lambda_i}^* = \underset{x \in \mathbb{R}^n}{\text{argmin}} \left\| \begin{bmatrix} A \\ \sqrt{\lambda_i} I_n \end{bmatrix} x - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|^2,$$

using LSQR with preconditioner

$$P_i = \lambda_i^{-1/2}(I_n - V_i G_i V_i^T),$$

and with initial guess $x_0^{(i)}$.

- 11: **end for**
-

6.3.3.1 Estimating the statistical dimension

In Algorithm 14 it is necessary to obtain estimates of the statistical dimension for various values of λ . One approach is to sketch A with an embedding matrix of size s considered to be an upper bound for the statistical dimension. This could be n if no information is known, as the sketching step is not the dominant part of the algorithm. The singular values of SA can then be used as estimates for the leading singular values of A , as described in Chapter 3. In particular, for each value of λ_i we can find an estimate $\hat{\text{sd}}_{\lambda_i}(A)$ for the statistical dimension with

$$\hat{\text{sd}}_{\lambda_i}(A) = \sum_{j=1}^s \frac{1}{1 + \frac{\lambda_i}{\sigma_j(SA)^2}}.$$

Although we may need to compute the SVD of a matrix with greater dimensions than necessary

once, a good estimate of $\text{sd}_{\lambda_i}(A)$ for each value of λ_i will allow us to truncate the low-rank structure in the preconditioner, making it very cheap to apply. Note that these estimates come without any additional cost, as computing the SVD of SA is a necessary step in Algorithm 14.

Other methods to estimate the statistical dimension use a randomized trace estimator such as described in [15]. However, this will involve solving a linear system, possibly approximately, as in [201]. Avron et al. propose a new method in [13] with which the statistical dimension can be estimated in $\mathcal{O}(\text{nnz}(A))$ time up to a constant factor, yet as is noted in [201], this is exclusively applicable in a context of very rapid decay of $\sigma_i(A)$.

6.3.4 Convergence analysis

Throughout this analysis, let $A = U_A \Sigma_A V_A^T$ be the economy-sized SVD of A . We again employ the structural conditions proposed in [59]. The proofs on the condition numbers of the preconditioned matrices are again inspired by the proofs in [59]. We first show these conditions are equivalent to the conditions used in [13], as also follows from Lemma 12 in [13].

The following lemma relates the condition in [59] to the condition in [13], and their relation to the statistical dimension.

Lemma 6.3. *Let U_1 consist of the first m rows of the left singular vectors of B such that*

$$B = \begin{bmatrix} A \\ \sqrt{\lambda}I \end{bmatrix} = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \Sigma_B V_B.$$

Define a diagonal matrix Σ_λ by

$$\Sigma_\lambda = \Sigma_A(\Sigma_A^2 + \lambda I)^{-1/2} = (\Sigma_A^2 + \lambda I)^{-1/2} \Sigma_A.$$

For an embedding matrix $S \in \mathbb{R}^{s \times m}$, $s < n < m$, we have

$$\|U_1^T S^T S U_1 - U_1^T U_1\| = \|\Sigma_\lambda U_A^T S^T S U_A \Sigma_\lambda - \Sigma_\lambda^2\|.$$

Furthermore

$$\|U_1\|_F^2 = \|U_A \Sigma_\lambda\|_F^2 = \text{sd}_\lambda(A).$$

Proof. First note that we have the following for the right singular vectors and values of B

$$B^T B = A^T A + \lambda I = V_A(\Sigma_A^2 + \lambda I)V_A^T = V_B \Sigma_B^2 V_B^T,$$

so $V_B = V_A$ and $\Sigma_B = (\Sigma_A^2 + \lambda I)^{1/2}$. Then

$$\begin{bmatrix} A \\ \sqrt{\lambda}I \end{bmatrix} = \begin{bmatrix} U_1(\Sigma_A^2 + \lambda I)^{1/2}V_A^T \\ U_2(\Sigma_A^2 + \lambda I)^{1/2}V_A^T \end{bmatrix},$$

so that

$$U_1 = AV_A(\Sigma_A^2 + \lambda I)^{-1/2} = U_A \Sigma_A(\Sigma_A^2 + \lambda I)^{-1/2} = U_A \Sigma_\lambda.$$

The first result follows; as for the second result, note that

$$\|U_1\|_F^2 = \|\Sigma_\lambda\|_F^2 = \|\Sigma_A(\Sigma_A^2 + \lambda I)^{-1/2}\|_F^2 = \sum_{i=1}^n \frac{\sigma_i^2}{\sigma_i^2 + \lambda} = \text{sd}_\lambda(A).$$

□

We use one of these equivalent conditions in the convergence theorems for Algorithm 14.

Theorem 6.4. *Assume the notation of Lemma 6.3 and suppose the following condition holds for some $\epsilon > 0$*

$$\|U_1^T S^T S U_1 - U_1^T U_1\| \leq \epsilon. \quad (6.20)$$

Let R be such that

$$R^T R = (SA)^T SA + \lambda I_n.$$

Then

$$\kappa(BR^{-1}) \leq \sqrt{\frac{1+\epsilon}{1-\epsilon}}.$$

Proof. We start from condition (6.20) to find

$$-\epsilon I \preceq \Sigma_\lambda U_A^T S^T S U_A \Sigma_\lambda - \Sigma_\lambda^2 \preceq \epsilon I.$$

Multiply these inequalities by V_A from the left and V_A^T from the right and use the fact that

$$V_A \Sigma_\lambda U_A^T = V_A (\Sigma_A^2 + \lambda I)^{-1/2} V_A V_A^T \Sigma_A = (A^T A + \lambda I)^{-1/2} A^T,$$

to obtain

$$\begin{aligned} -\epsilon I &\preceq (A^T A + \lambda I)^{-1/2} A^T S^T S A (A^T A + \lambda I)^{-1/2} - \\ &\quad (A^T A + \lambda I)^{-1/2} A^T A (A^T A + \lambda I)^{-1/2} \preceq \epsilon I. \end{aligned}$$

By multiplying with $(A^T A + \lambda I)^{1/2}$ on either side, we find

$$-\epsilon B^T B \preceq A^T S^T S A - A^T A \preceq \epsilon B^T B,$$

which is equivalent to

$$-\epsilon B^T B \preceq A^T S^T S A + \lambda I - (A^T A + \lambda I) \preceq \epsilon B^T B.$$

We then have

$$(1 - \epsilon) B^T B \preceq R^T R \preceq (1 + \epsilon) B^T B,$$

which results in the following inequalities

$$(1 + \epsilon)^{-1/2} \leq \sigma_{\min}(BR^{-1}) \leq \sigma_{\max}(BR^{-1}) \leq (1 - \epsilon)^{-1/2}.$$

□

6.3.5 Sketching matrices

The relationship between condition (6.20) and the statistical dimension of the problem can be explained by random matrix multiplication bounds in terms of the stable rank [63]. See for instance Corollary 15 and its proof in [13]. The following theorem describes the size that is required for various oblivious subspace embeddings to obtain a desired level of accuracy.

Theorem 6.5 (Ozaslan et al. (2023) [201]). *Let $A \in \mathbb{R}^{m \times n}$, $m \geq n$, and $\lambda > 0$ have statistical dimension $\text{sd}_\lambda(A)$. Let U_1 consist of the first m rows of the left singular vectors of $B = [A^T \sqrt{\lambda} I_n]^T$ such that*

$$B = \begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix} = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \Sigma_B V_B.$$

The condition

$$\|U_1^T S^T S U_1 - U_1^T U_1\| \leq \epsilon$$

for an embedding matrix $S \in \mathbb{R}^{s \times m}$ is satisfied with probability at least $1 - \delta$ in the following cases:

- *S is a sparse subspace embedding [275] with one non-zero element in each column and*

$$s = \Omega(\text{sd}_\lambda(A)^2 / (\epsilon^2 \delta)).$$

- *S is a subsampled randomized trigonometric transform [181] and*

$$s = \Omega((\text{sd}_\lambda(A) + \log(1/\epsilon\delta) \log(\text{sd}_\lambda(A)/\delta)) / \epsilon^2).$$

- *S is a sub-Gaussian embedding [259] and*

$$s = \Omega((\text{sd}_\lambda(A) / \epsilon^2)).$$

Here, the $\Omega(\cdot)$ notation is defined as $a(n) = \Omega(b(n))$, if there exist two integers k and n_0 such that for all $n > n_0$ we have $a(n) \geq kb(n)$.

One could also use a subsampling matrix based on leverage scores to satisfy the condition of the theorem, see for instance [59]. Note that the sketching dimensions in Theorem 6.5 scale very similarly in $\text{sd}_\lambda(A)$ as regular embedding dimensions in n (see Section 2.1.3).

6.4 A weighted regularization function

We now consider the set of problems

$$\min_x \|Ax - b\|_2^2 + \lambda_i \|Lx\|_2^2, \quad i = 1, \dots, N, \quad (6.21)$$

where $A \in \mathbb{R}^{m \times n}$, $L \in \mathbb{R}^{p \times n}$, and $\lambda_1, \dots, \lambda_N > 0$ are regularization parameters.

In Sections 6.2 and 6.3 we discussed the case $L = I_n$. If this is not the case, we often have that L is a diagonal matrix with entries not all equal to zero, an upper triangular matrix that spans some

differential operator, or rectangular differential operators [130]

$$L_1 = \begin{bmatrix} 1 & -1 & & \\ & \ddots & \ddots & \\ & & 1 & -1 \end{bmatrix} \in \mathbb{R}^{(n-1) \times n}, \quad L_2 = \begin{bmatrix} 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 2 & -1 \end{bmatrix} \in \mathbb{R}^{(n-2) \times n}.$$

It is generally assumed that L is chosen such that

$$\mathcal{N}(A) \cap \mathcal{N}(L) = \emptyset. \quad (6.22)$$

In the overdetermined case ($m > n$) we can reformulate the problem in two ways. Firstly, we may aim to solve

$$\min_x \|B_i x - \hat{b}\|_2^2, B_i = \begin{bmatrix} A \\ \sqrt{\lambda_i} L \end{bmatrix}, \hat{b} = \begin{bmatrix} b \\ 0 \end{bmatrix}, \quad (6.23)$$

and to find preconditioners R_i such that $\kappa(B_i R_i^{-1}) = \mathcal{O}(1)$. Note that B_i is full column rank if (6.22) is assumed.

Secondly, if L is invertible, we can perform a change of variable and solve the system

$$\min_y \|\hat{B}_i y - \hat{b}\|_2^2, \hat{B}_i = \begin{bmatrix} AL^{-1} \\ \sqrt{\lambda_i} I_n \end{bmatrix}, \hat{b} = \begin{bmatrix} b \\ 0 \end{bmatrix}, \quad (6.24)$$

The minimizer x_λ is given by $x_\lambda = L^{-1} y_\lambda$ and we then wish to find preconditioners R_i such that $\hat{B}_i R_i^{-1}$ is well-conditioned.

In the underdetermined case ($m < n$) we may consider the problem [138, 187]

$$\min_{y \in \mathbb{R}^n, w \in \mathbb{R}^m} \left\| \begin{bmatrix} y \\ w \end{bmatrix} \right\|_2^2 \quad \text{s.t.} \quad D_i \begin{bmatrix} y \\ w \end{bmatrix} = b, \quad D_i = \begin{bmatrix} \lambda^{-1/2} AL^{-1} & I_m \end{bmatrix}, \quad (6.25)$$

and we wish to precondition D_i with R_i such that $\kappa(R_i^{-T} D_i) = \mathcal{O}(1)$. The condition that L is invertible to transform (6.21) to LS standard form in the underdetermined case makes it less suitable for our sketch-and-precondition algorithms, and we do not discuss it further.

We consider how our work for $L = I$ may be extended to this setting in the overdetermined case.

6.4.1 GSVD-based approach to the overdetermined problem

Consider first (6.23), and a sketch dimension $s > n$. Our goal is to find a good preconditioner R_i for B_i as in (6.23). Suppose we have R_i such that

$$R_i^T R_i = A^T S^T S A + \lambda_i L^T L,$$

then we are able to show that $\kappa(B_i R_i^{-1}) \leq \kappa(S U_A)$ as before, where $U_A \in \mathbb{R}^{m \times s}$ are left singular vectors of A . We propose a method that requires a single generalized SVD (GSVD) [95, 204] of size $(s + p) \times n$ and a single LU factorization of an $n \times n$ matrix.

We extend the idea of performing a single decomposition to solve for a sequence of values of λ_i to this setting. Consider the generalized singular value decomposition of the following matrices, where

$SA \in \mathbb{R}^{s \times n}$, $L \in \mathbb{R}^{p \times n}$, and $s \geq n \geq p$,

$$\begin{bmatrix} SA \\ \sqrt{\lambda}L \end{bmatrix} = \begin{bmatrix} U_\lambda C_\lambda \\ V_\lambda S_\lambda \end{bmatrix} H_\lambda, \quad \begin{bmatrix} SA \\ L \end{bmatrix} = \begin{bmatrix} U_1 C_1 \\ V_1 S_1 \end{bmatrix} H_1. \quad (6.26)$$

In these decomposition, U_λ and U_1 are $s \times n$ with orthonormal columns, V_λ and V_1 are $p \times p$ square orthogonal, C_λ and C_1 are $n \times n$ square diagonal, S_λ and S_1 are $p \times n$ diagonal, and H_λ and H_1 are $n \times n$ matrices without structure. The diagonal matrices are such that

$$C_\lambda^T C_\lambda + S_\lambda^T S_\lambda = C_1^T C_1 + S_1^T S_1 = I_n.$$

We have the following identities relating the GSVDs of the matrices in (6.26):

$$\begin{aligned} U_\lambda &= U_1 := U, \quad V_\lambda = V_1 := V, \\ C_\lambda &= C_1 D_\lambda^{-1}, \quad S_\lambda = \sqrt{\lambda} S_1 D_\lambda^{-1} \\ H_\lambda &= D_\lambda H_1 \end{aligned}$$

where

$$D_\lambda = \sqrt{C_1^T C_1 + \lambda S_1^T S_1}$$

is a diagonal matrix. Furthermore, we have

$$\begin{aligned} (AS)^T(AS) + \lambda L^T L &= H_\lambda^T C_\lambda^T U^T U C_\lambda H_\lambda + H_\lambda^T S_\lambda^T V^T V S_\lambda H_\lambda \\ &= H_\lambda^T [C_\lambda^T C_\lambda + S_\lambda^T S_\lambda] H_\lambda \\ &= H_\lambda^T H_\lambda \\ &= H_1^T D_\lambda^T D_\lambda H_1 \\ &= R^T R, \end{aligned}$$

for $R = D_\lambda H_1$. We wish to quickly apply the inverse of R . For that reason, suppose we have an LU factorization of $L_H U_H = H_1$, then

$$R^{-1} = U^{-1} L^{-1} D_\lambda^{-1},$$

which may be applied to a vector $x \in \mathbb{R}^n$ in $\mathcal{O}(n^2)$ operations. The main advantage is we only need to compute the GSVD of the matrix $B = [A^T L^T]^T$, that is, for $\lambda = 1$.

We also compute the sketch-and-solve solution to be used as an initial guess:

$$x_0^{(i)} = \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \left\| \begin{bmatrix} SA \\ \sqrt{\lambda}L \end{bmatrix} x - \begin{bmatrix} Sb \\ 0 \end{bmatrix} \right\|. \quad (6.27)$$

It immediately follows that

$$\begin{bmatrix} SA \\ \sqrt{\lambda}L \end{bmatrix}^\dagger = \left(\begin{bmatrix} U_\lambda C_\lambda \\ V_\lambda S_\lambda \end{bmatrix} H_\lambda \right)^\dagger = H_\lambda^{-1} \begin{bmatrix} C_\lambda U_\lambda^T & S_\lambda V_\lambda^T \end{bmatrix}.$$

As a result,

$$x_0^{(i)} = \begin{bmatrix} SA \\ \sqrt{\lambda}L \end{bmatrix}^\dagger \begin{bmatrix} Sb \\ 0 \end{bmatrix} = H_\lambda^{-1} C_\lambda U_\lambda^T S b = H_1^{-1} C_1 D_\lambda^{-2} U^T S b.$$

The full algorithm is displayed in Algorithm 15. The most expensive steps are the computation of the GSVD of SA and L , and the computation of the LU decomposition of H_1 ; the first costs $\mathcal{O}(n^3 \log(n))$ operations and the second costs $\mathcal{O}(n^3)$ work [17]. The cost of computing the initial guess is $3n^2$ flops and the cost of applying the preconditioner is $2n^2$ flops. The total cost is $\mathcal{O}(mn \log(m) + n^3 \log(n) + N(mn \log(1/\varepsilon)))$ to solve for N regularization parameters.

Algorithm 15 A GSVD-based sketch-and-precondition algorithm for Tikhonov regularized problems with weighted regularization function. Given an $m \times n$ matrix A , $m \gg n$ and a set of regularization parameters $\lambda_1, \dots, \lambda_N$, this scheme computes solutions $x_{\lambda_i}^*$ to (6.23).

- 1: Draw an $s \times m$ random embedding matrix S .
- 2: Compute $Y = SA$ and Sb .
- 3: Compute $[U, \sim, H_1, C_1, S_1] = \text{gsvd}(Y, L, \text{'econ'})$.
- 4: Compute $[L_H, U_H] = \text{lu}(H_1)$.
- 5: **for** $i = 1, 2, \dots, N$ **do**
- 6: Compute the entries of the diagonal matrix $D_i = (C_1^T C_1 + \lambda S_1^T S_1)^{1/2}$.
- 7: Compute $x_0^{(i)} = U_H^{-1} L_H^{-1} C_1 D_i^{-2} U^T S b$.
- 8: Solve the system

$$x_{\lambda_i}^* = \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \left\| \begin{bmatrix} A \\ \sqrt{\lambda_i}L \end{bmatrix} x - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|^2,$$

using LSQR with preconditioner

$$P_i = U_H^{-1} L_H^{-1} D_i^{-1},$$

and with initial guess $x_0^{(i)}$.

9: **end for**

6.4.2 Convergence analysis

We obtain a convergence result very similar to Lemma 6.2

Lemma 6.6. *Let $A \in \mathbb{R}^{m \times n}$, $m > n$ have SVD $A = U_A \Sigma_A V_A^T$, where $U_A \in \mathbb{R}^{m \times n}$, let $L \in \mathbb{R}^{p \times n}$, and let $\lambda > 0$. Let $S \in \mathbb{R}^{s \times m}$, $m > s \geq n$, be a matrix such that $1 - \epsilon < \sigma_{\min}(SU_A) < \sigma_{\max}(SU_A) < 1 + \epsilon$ for some $0 < \epsilon < 1$. Suppose $R \in \mathbb{R}^{n \times n}$ is a matrix such that*

$$R^T R = A^T S^T S A + \lambda L^T L.$$

Then

$$\kappa_2(BR^{-1}) \leq \frac{1 + \epsilon}{1 - \epsilon},$$

where

$$B = \begin{bmatrix} A \\ \sqrt{\lambda}L \end{bmatrix}. \quad (6.28)$$

Proof. We start from (6.13) in the proof of Lemma 6.2, as the assumption on the random embedding

is the same. We have that

$$\sigma_{\min}(SU_A)^2 A^T A + \lambda L^T L \preceq A^T S^T S A + \lambda L^T L \preceq \sigma_{\max}(SU_A)^2 A^T A + \lambda L^T L,$$

and by the positive semi-definiteness of $L^T L$, and under the assumption $\sigma_{\min}(SU_A) \leq 1 \leq \sigma_{\max}(SU_A)$, we have that

$$\sigma_{\min}(SU_A)^2 B^T B \preceq R^T R \preceq \sigma_{\max}(SU_A)^2 B^T B.$$

The result follows in the same manner as in the proof of Lemma 6.2. $\square$

Since we use a partly-exact sketch, the conditioning of L (in addition to the conditioning of A) is not relevant for the conditioning of the final system. One may interpret the preconditioner arising from a partly-exact sketch as approximately orthogonalizing A and exactly orthogonalizing (or inverting) L .

6.5 Numerical experiments

In this section we aim to present numerical experiments to investigate the performance of the algorithms we have introduced. We use three different problems

1. A square, dense, inverse diffusion problem with very rapidly decaying singular values. This experiment portrays the performance of the algorithms suitable to the low statistical dimension context introduced in Section 6.3.
2. A highly-overdetermined and ill-conditioned sparse matrix. We display the performance of the algorithms introduced in Section 6.2. It shows the effectiveness of iterative algorithms when the matrix-vector multiplications are computed rapidly and the sketch is much smaller than the original problem.
3. A dense and moderately overdetermined inverse tomography problem as introduced in Section 6.1.1. The sketch is not much smaller than the original matrix in this case, so the algorithms in Section 6.2 are less competitive compared to computing the SVD of A directly. This example shows the efficiency of the Cholesky-based algorithm from 6.2.3 when N is small.

Throughout these experiments, we use a sparse random embedding matrix with $\zeta = 8$ and a sketching dimension of $s = 2\lceil \text{sd}_{\lambda_N}(A) \rceil$ or $s = 2n$. The operator A is stored as a matrix in each of the experiments.

6.5.1 A square ill-posed inverse problem

We start with an experiment on a type of problem that has not been studied so far in this thesis: a square linear system. We look at an inverse diffusion problem from the IRtools toolbox [107]. The forward problem is a simple diffusion problem with homogeneous Neumann boundary conditions solved with a Crank-Nicolson-Galerkin finite-element method [81], the details of which are beyond the scope of this thesis. The inverse problem is to reconstruct the initial conditions from the solution at the final time-step. We have access to the operator A corresponding to the forward problem and its inverse. For this experiment, we start by explicitly storing this operator as a matrix. The matrix

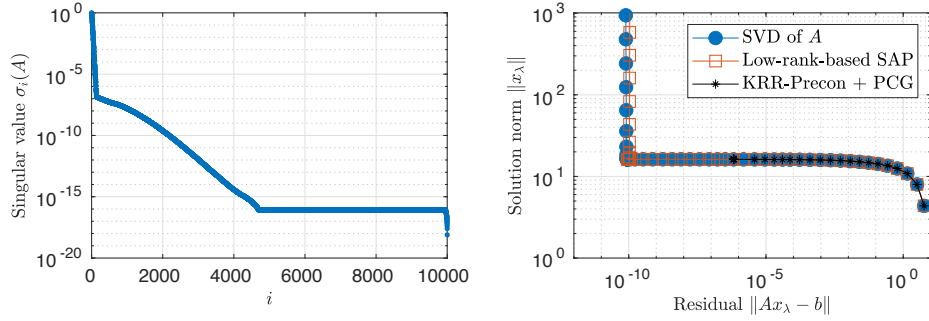


Figure 6.4: The decaying singular value spectrum of the forward diffusion operator, and the corresponding L-curve computed using three methods discussed in Section 6.5.1. The corner of the L-curve corresponds to $\lambda = 10^{-20}$.

has size $10^4 \times 10^4$, is dense, and has rapidly decaying singular values. From exact (random) initial conditions x_{true} we form a perturbed right-hand side $b = Ax_{\text{true}} + e$, where e is white noise of magnitude $\|e\| = 10^{-10}$. The singular values of A and associated L-curve to the problem are shown in Figure 6.4.

We compare three methods: computing the SVD of A , Algorithm 14, and preconditioned conjugate gradients (PCG) with the kernel ridge regression (KRR) preconditioner [12]. This final algorithm, when applied to least squares problems, preconditions the normal equations with a matrix of low-rank structure P_{KRR} given by

$$R_{KRR} = \text{chol}(SA(SA)^T + \lambda I_s), \quad U = (SA)^T R_{KRR}^{-1}, \quad P_{KRR}^{-1} = \frac{1}{\lambda}(I - UU^T), \quad (6.29)$$

where $S \in \mathbb{R}^{s \times m}$ and $s < n$. This preconditioner is applied to solve

$$(A^T A + \lambda I)x = A^T b,$$

with preconditioned conjugate gradients (PCG). LSQR and PCG are set to `tol` = 10^{-10} , and the exact singular values of A are used to compute the statistical dimension. We sketch with oversampling parameter 2; that is, $s = 2\lceil \text{sd}_{\lambda_N}(A) \rceil$. We consider $N = 50$ admissible regularization parameters ranging from $\lambda_1 = 1$ to $\lambda_N = 10^{-28}$. The results are shown in Figure 6.5.

The low-rank-based preconditioner from Algorithm 14 has the lowest computational time without sacrificing stability; it has very small start-up cost, scales well, and computes accurate solutions even for small values of λ . The KRR-based preconditioner fails there because the Cholesky decomposition cannot be computed when λ is too small. Even before it fails, by invoking the normal equations the maximal attainable residual is limited (see also Figure 4.2).

6.5.2 A highly ill-conditioned sparse matrix

We next consider a highly-overdetermined and very ill-conditioned sparse linear system. The matrix A has size $10^6 \times 5000$, condition number $\kappa_2(A) \approx 10^{-16}$, and has 4750895 non-zeros, which is about 0.001 density. It is created with MATLAB's `sprand` function. The singular values do not decay very rapidly, so we cannot use any of the low-rank-based algorithms. The solution and noise in the right-hand side are randomly generated with noise level 10^{-4} , and the tolerance in LSQR is `tol`

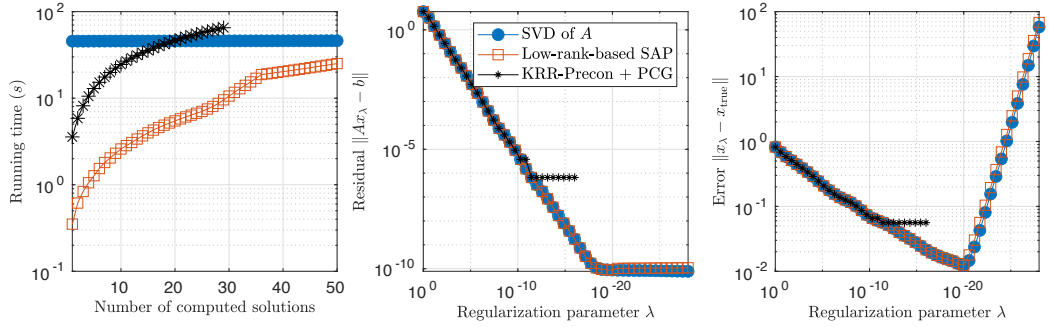


Figure 6.5: An ill-posed inverse diffusion problem solved for a sequence of regularization parameters. We compare the SVD, the low-rank-based sketch-and-precondition (SAP) method in Algorithm 14, and the KRR preconditioner in (6.29) combined with preconditioned conjugate gradients. The computation of the Cholesky decomposition in (6.29) fails for small values of λ . The left panel shows the cumulative computational time, the middle panel the residual norm for different regularization parameters, and the right panel the error with the true solution.

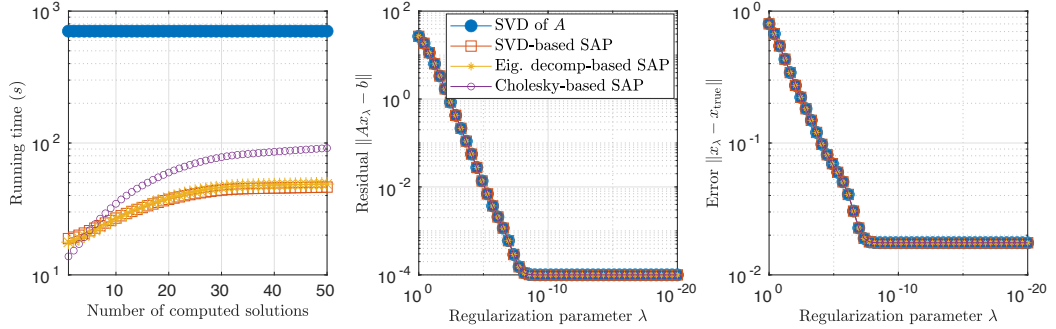


Figure 6.6: Different methods from Section 6.2 compared against computing the SVD of A . The problem is a highly-overdetermined and ill-conditioned sparse matrix. The left panel shows the cumulative running time, the middle panel the residual, and the right panel the error with x_{true} .

$= 10^{-6}$. The results are shown in Figure 6.6.

Figure 6.6 shows all sketch-and-precondition (SAP) methods are much faster than computing the SVD of the matrix. The start-up time in the Cholesky-based algorithm is smallest, but then the algorithm scales worse than the other two iterative methods. There is not much difference observed between the SVD- and eigendecomposition-based preconditioners. We note that for small values of λ , the initial guess in all cases is very close to the required residual, which mean LSQR terminates with very few iterations.

6.5.3 A dense and overdetermined problem from CT

We finally consider a problem arising in computed tomography (CT). The coefficient matrix $A \in \mathbb{R}^{259695 \times 10000}$ is a discretization of the Radon transform (see 6.1.1). The problem corresponds to 2D X-ray tomography over $R = 1791$ angles with $P = 145$ lines per angle. This problem does not have very rapidly decaying singular values, and so we only compare the computation of the SVD and the preconditioners in general settings (SVD, eigendecomposition, and Cholesky decomposition-based). The right-hand side is a noisy sinogram of the modified Shepp-Logan phantom of dimension 100×100 . The noise level is $\|e\| = 10^{-4}$ and the LSQR tolerance is set to $\text{tol} = 10^{-6}$. The regularization parameters range from $\lambda_1 = 1$ to $\lambda_N = 10^{-20}$, where $N = 50$. The results are shown

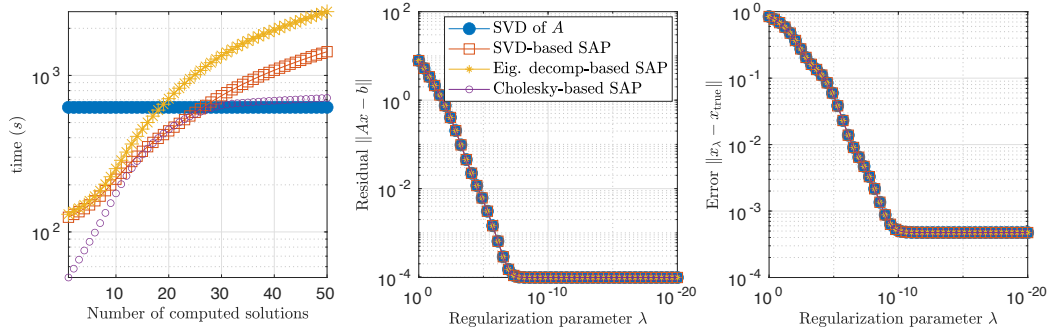


Figure 6.7: A CT problem where A is a discretized Radon transform. Note that A is applied through a regular matrix-vector transformation, not using a fast transform.

in Figure 6.7.

The iterative methods are less dominant compared to a full factorization in this example, although it must be noted that the Cholesky-based approach is very fast if one is interested in computing just a few solutions. The effectiveness of the sketch-and-precondition-based algorithms here is highly dependent on the specifics of the problem, in particular on the aspect ratio and the singular value spectrum. In this problem, the aspect ratio is approximately 26, which apparently does not lead to a significant speed up for sketch-and-precondition algorithms in this context.

6.6 Summary and outlook

We have discussed various algorithms to efficiently compute solutions to regularized problems for different regularization parameters. The algorithms allow practitioners to select an appropriate regularization parameter based on high-accuracy corresponding solutions. We discussed a range of contexts: over- and underdetermined systems, low-rank structure in the data matrix, and weighted regularization functions. All algorithms are part of the sketch-and-precondition framework, and leverage the structure of a (directly) regularized problem to efficiently compute randomized preconditioners and initial guesses. Experiments showed the effectiveness of the iterative algorithms when the sketch SA is much smaller than A (either because A has very small statistical dimension or because A is highly over-determined) and when application of the operator A is cheap.

We specifically highlight the preconditioner that can exploit low-rank structure. This scheme does not require the problem to be highly over- or underdetermined, in fact the matrix can be nearly square or even square, provided it is sufficiently low-rank. This is different from many other sketch-and-precondition algorithms and makes the scheme exceptionally well-suited for inverse problems, where A is often a discretized integral operator with a rapidly decaying spectrum. Under the assumption that A is a discretization of some operator, it would be worthwhile to investigate ways to accelerate the sketching of these matrices, for instance by considering physical interpretations of sketching these operators. Furthermore, sketch-and-solve-type methods could be of interest in problems with high noise levels where backward stability is not the main concern. A sketch-and-solve technique could be employed to estimate the optimal regularization parameter, for instance, after which a higher-accuracy is computed with that regularization parameter.

The context of the entirety of this chapter was direct regularization. In practice, however, iterative regularization methods are currently popular for large-scale problems [130]. There is no

explicit regularization parameter present in these methods, rather regularization occurs through early termination of an iterative method. Preconditioning these iterations is contradictory to this process, since semi-convergence relies on the singular value spectrum decaying. However, if there is some information known about the relevant part of the spectrum, it might be possible to develop randomized preconditioning techniques for those directions only. This could potentially speed up the iterations before semi-convergence sets in. A similar argument can be made for hybrid regularization methods.

We have not discussed the inherent regularization effect that randomization can have. We saw this before in Section 5.6.2, where random noise improved the conditioning and convergence of the problem. Mahoney [174] discusses that through implicit regularization one can quantify “*a sense in which the output of randomized matrix algorithms is even “better” than the output of the corresponding deterministic algorithms*” [174, p. 45]. This is an area we have not yet investigated in the context of regularized least squares.

Extremely large-scale regularized least squares, and inverse problems specifically, are ubiquitous in many scientific areas. These matrices come in all shapes and with varying levels of sparsity and/or fast transforms. In this chapter, we showed that direct regularization for large-scale problems can be within reach by exploiting randomized NLA across a range of matrix types. A deeper understanding of the potential of randomized algorithms in this context is therefore extremely worthwhile to pursue.

Chapter 7

Concluding remarks

In this thesis, several algorithms have been proposed and analysed for fundamental problems in numerical linear algebra: numerical rank (or singular value) estimation, least squares problems, and their regularized variants. The basis of all of these algorithms is randomization; using a random matrix allows us to project part of the problem onto a lower dimension. In this lower dimensional space, we can perform classical numerical linear algebra (NLA) routines at an acceptable cost. Randomized algorithms have already been demonstrated to provide significant improvements in computational, memory, and communication cost for a number of fundamental numerical linear-algebraic tasks, predominantly for low-rank approximation and least squares problems.

However, as we argued in the introduction, questions around implementation, parameter selection, and numerical stability require additional attention. In this thesis, we set out to contribute to randomized NLA literature with respect to these themes. In particular, we focused on parameter selection for low-rank approximation, numerical stability of randomized algorithms for least squares, and the use of randomization for parameter selection in regularized least squares.

In the context of numerical rank estimation, we have shown in Chapter 3 that sketching the dominant column space of an approximately low rank matrix preserves the order of magnitude of the singular values (provided the tail of singular values is not heavy). This sketching procedure is commonly applied in low-rank approximation algorithms to retrieve an approximate range, or for a posteriori error estimation in low-rank approximation algorithms [127, 181]. We demonstrate that we can use the full spectrum of this sketch a priori, to inform algorithmic choices. Selection of a suitable target rank in a fixed-precision low-rank approximation algorithm can be informed by this spectrum. Combining our numerical rank estimation algorithm with existing low-rank approximation algorithms allows practitioners to solve the fixed-precision problem more efficiently.

The theory of Chapter 3 leans heavily on combining linear algebra techniques with random matrix and high-dimensional statistics theory. Mahoney [174] argues that a crucial development in randomized NLA was the ‘decoupling of the randomization from the linear algebra’. This decoupling is also what allowed us to prove the results in Chapter 3. Sketching in the embedding regime (see Figure 1.5) can be viewed as sampling from a high-dimensional distribution described by the spectral properties of the original matrix. Similarly, samples from important models in high-dimensional statistics, for instance the spiked covariance model [153], can be investigated through a linear algebraic lens (e.g. as in [211]). We believe there is a wealth of knowledge in both the NLA

and statistical communities that can aid these respective fields.

A large part of this thesis was dedicated to randomized algorithms for least squares (LS) problems. The sketch-and-precondition framework has been one of the main successes of randomized NLA, particularly for large-scale, highly-overdetermined problems. Despite the framework having been the topic of much research in recent years, its finite precision behaviour was ill-understood.

We investigated this behaviour in Chapter 4 and demonstrated that sketch-and-precondition algorithms are numerically unstable without good initialization. In a number of metrics, namely the residual, error, and backward error, sketch-and-precondition algorithms perform significantly worse than stable classical alternatives. These observed instabilities raise doubt on the stability of preconditioned iterative solvers for LS problems in general, as the accumulation of rounding errors is also observed for deterministic preconditioners. Further investigation is required into the finite precision behaviour of preconditioned Krylov subspace methods to understand the source of the instabilities. A deeper understanding into the finite-precision mechanics of preconditioning might aid in the development of classes of preconditioners that do not suffer from these numerical errors.

In Chapter 5 we aimed to stabilize the sketch-and-precondition framework. We demonstrate that much of the instability can be resolved by using sketch-and-solve initialization in the iterative solver. An important take-away for practitioners from Chapter 5 is that this initialization should always be employed. However, for highly inconsistent and moderately ill-conditioned problems the resulting algorithm is still not backward stable.

To resolve the lack of backward stability completely, we introduced a new algorithm in Chapter 5 called sketch-and-apply. We show, using classical stability analysis, that it is backward stable, and support this further with experimental evidence. However, the computational complexity of sketch-and-apply is on par with direct solvers.

Chapter 5 raised the question: is there a numerically stable way to employ the randomized preconditioner that is faster than classical direct algorithms? Recent work [101] shows that this can be answered affirmatively. [101] builds heavily on traditional NLA techniques, such as iterative refinement and numerical stability analysis, and pays significant attention to parameter selection and reliable stopping criteria. The successful combination of traditional NLA themes and randomization further emphasizes that investigation of these themes improves randomized algorithms.

A fundamental fact of ill-conditioned LS problems is that backward stable solutions are not necessarily solutions with small forward errors. If a practitioner is primarily interested in recovering a solution that is close to some ground truth, they must turn to regularization. In Chapter 6, we discussed how the sketch-and-precondition framework can be extended to this context. We argue that through randomization, one can now tackle a class of large-scale problems with direct regularization, where that was previously impossible. The key contribution presented in Chapter 6 is a sketch-and-precondition-type algorithm for ill-posed inverse problems of any dimension (even square), provided the statistical dimension is small.

Chapter 6 also serves to demonstrate the potential of randomized algorithms on regularized LS problems, and particularly linear inverse problems. Various avenues for further research are suggested, in particular combining randomization with iterative or hybrid regularization methods.

As the field of randomized numerical linear algebra matures and our algorithms are adopted by other communities, it is of paramount importance to accompany them with implementation suggestions rooted in theory and computational experiments, as well as an understanding of their

finite precision behaviour. The work presented in this thesis contributes to this goal.

Bibliography

- [1] J.-F. P. Abascal, S. R. Arridge, R. H. Bayford, and D. S. Holder. Comparison of methods for optimal choice of the regularization parameter for linear electrical impedance tomography of brain function. *Physiol. Meas.*, 29(11):1319, 2008.
- [2] D. Achlioptas. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *J. Comput. Syst. Sci.*, 66(4):671–687, 2003.
- [3] S. E. Ahmed, O. San, D. Bistrian, and I. Navon. Sketching methods for dynamic mode decomposition in spherical shallow water equations. In *AIAA SciTech Forum*, page 2325, 2022.
- [4] N. Ailon and B. Chazelle. Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. In *Proc. Annu. ACM Symp. Theory Comput.*, pages 557–563, 2006.
- [5] E. O. Åkesson and K. J. Daun. Parameter selection methods for axisymmetric flame tomography through Tikhonov regularization. *Appl. Opt.*, 47(3):407–416, 2008.
- [6] A. Alaoui and M. W. Mahoney. Fast randomized kernel ridge regression with statistical guarantees. *Adv. Neural Inf. Process. Syst.*, 28:775–783, 2015.
- [7] A. Alla and J. N. Kutz. Randomized model order reduction. *Adv. Comput. Math.*, 45:1251–1271, 2019.
- [8] S. Ambikasaran, D. Foreman-Mackey, L. Greengard, D. W. Hogg, and M. O’Neil. Fast direct methods for Gaussian processes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(2):252–265, 2016.
- [9] A. Andoni and H. L. Nguyễn. Eigenvalues of a matrix in the streaming model. *Proc. Annu. ACM-SIAM Symp. Discrete Alg.*, pages 1729–1737, 2013.
- [10] R. C. Atwood, A. J. Bodey, S. W. Price, M. Basham, and M. Drakopoulos. A high-throughput system for high-quality tomographic reconstruction of large datasets at Diamond Light Source. *Phil. Trans. R. Soc. A*, 373(2043):20140398, 2015.
- [11] G. Aubrun and S. J. Szarek. *Alice and Bob meet Banach*, volume 223. Amer. Math. Soc, 2017.
- [12] H. Avron, K. L. Clarkson, and D. P. Woodruff. Faster kernel ridge regression using sketching and preconditioning. *SIAM J. Matrix Anal. Appl.*, 38(4):1116–1138, 2017.

- [13] H. Avron, K. L. Clarkson, and D. P. Woodruff. Sharper bounds for regularized data fitting. In *APPROX/RANDOM*, pages 27:1–27:22, 2017.
- [14] H. Avron, P. Maymounkov, and S. Toledo. Blendenpik: Supercharging LAPACK’s least-squares solver. *SIAM J. Sci. Comput.*, 32(3):1217–1236, 2010.
- [15] H. Avron and S. Toledo. Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *J. ACM*, 58(2):1–34, 2011.
- [16] C. Bach, D. Ceglia, L. Song, and F. Duddeck. Randomized low-rank approximation methods for projection-based model order reduction of large nonlinear dynamical problems. *Int. J. Numer. Methods Eng.*, 118(4):209–241, 2019.
- [17] Z. Bai and J. W. Demmel. Computing the generalized singular value decomposition. *SIAM J. Sci. Comput.*, 14(6):1464–1486, 1993.
- [18] Z. Bai and J. W. Silverstein. *Spectral analysis of large dimensional random matrices*. Springer, 2010.
- [19] O. Balabanov, M. Beaupère, L. Grigori, and V. Lederer. Block subsampled randomized Hadamard transform for low-rank approximation on distributed architectures. *arXiv:2210.11295*, 2022.
- [20] O. Balabanov and L. Grigori. Randomized block Gram-Schmidt process for solution of linear systems and eigenvalue problems. *arXiv:2111.14641*, 2021.
- [21] O. Balabanov and L. Grigori. Randomized Gram-Schmidt process with application to GMRES. *SIAM J. Sci. Comput.*, 44(3):A1450–A1474, 2022.
- [22] G. Ballard, J. Demmel, and I. Dumitriu. Minimizing communication for eigenproblems and the singular value decomposition. Technical Report 237, LAPACK Working Note, 2010.
- [23] Z. Bar-Yossef. Sampling lower bounds via information theory. In *Proc. Annu. ACM Symp. Theory Comput.*, pages 335–344, 2003.
- [24] F. Bauer. Some considerations concerning regularization and parameter choice algorithms. *Inverse Probl.*, 23(2):837, 2007.
- [25] F. Bauer and M. A. Lukas. Comparing parameter choice methods for regularization of ill-posed problems. *Math. Comput. Simul.*, 81(9):1795–1841, 2011.
- [26] F. S. V. Bazán. Fixed-point iterations in determining the Tikhonov regularization parameter. *Inverse Probl.*, 24(3):035001, 2008.
- [27] F. S. V. Bazán, L. S. Borges, and J. B. Francisco. On a generalization of Regińska’s parameter choice rule and its numerical realization in large-scale multi-parameter Tikhonov regularization. *Appl Math Comput.*, 219(4):2100–2113, 2012.
- [28] F. S. V. Bazán and J. B. Francisco. An improved fixed-point algorithm for determining a Tikhonov regularization parameter. *Inverse Probl.*, 25(4):045007, 2009.

- [29] B. Beckermann and A. Townsend. On the singular values of matrices with displacement structure. *SIAM J. Mat. Anal. Appl.*, 38(4):1227–1248, 2017.
- [30] M. Benzi. Preconditioning techniques for large linear systems: A survey. *J. Comput. Phys.*, 182(2):418–477, 2002.
- [31] M. Bertero, P. Boccacci, and C. De Mol. *Introduction to inverse problems in imaging*. CRC press, 2021.
- [32] Å. Björck, T. Elfving, and Z. Strakoš. Stability of conjugate gradient and Lanczos methods for linear least squares problems. *SIAM J. Mat. Anal. Appl.*, 19(3):720–736, 1998.
- [33] Å. Björck and C. C. Paige. Loss and recapture of orthogonality in the modified Gram–Schmidt algorithm. *SIAM J. Matrix Anal. Appl.*, 13(1):176–190, 1992.
- [34] Björck, Å. *Numerical Methods for Least Squares Problems*. SIAM, 1996.
- [35] N. Boullé and A. Townsend. Learning elliptic partial differential equations with randomized linear algebra. *Found. Comput. Math.*, 23(2):709–739, 2023.
- [36] J. Bourgain, S. Dirksen, and J. Nelson. Toward a unified theory of sparse dimensionality reduction in Euclidean space. In *Proc. Annu. ACM Symp. Theory Comput.*, pages 499–508, 2015.
- [37] C. Boutsidis and A. Gittens. Improved matrix algorithms via the subsampled randomized Hadamard transform. *SIAM J. Mat. Anal. Appl.*, 34(3):1301–1340, 2013.
- [38] C. Boutsidis and D. P. Woodruff. Optimal CUR matrix decompositions. In *Proc. Annu. ACM Symp. Theory Comput.*, pages 353–362, 2014.
- [39] V. Braverman, P. Drineas, C. Musco, C. Musco, J. Upadhyay, D. P. Woodruff, and S. Zhou. Near optimal linear algebra in the online and sliding window models. In *Proc. Annu. Symp. Found. Comput. Sci.*, pages 517–528. IEEE, 2020.
- [40] A. Buhr and K. Smetana. Randomized local model order reduction. *SIAM J. Sci. Comput.*, 40(4):A2120–A2151, 2018.
- [41] P. Bürgisser and F. Cucker. Smoothed analysis of Moore–Penrose inversion. *SIAM J. Mat. Anal. Appl.*, 31(5):2769–2783, 2010.
- [42] F. Cakoni and D. Colton. A survey in mathematics for industry: Open problems in the qualitative approach to inverse electromagnetic scattering theory. *Eur. J. Appl. Math.*, 16(3):411–425, 2005.
- [43] D. Calandriello, L. Carratino, A. Lazaric, M. Valko, and L. Rosasco. Gaussian process optimization with adaptive sketching: Scalable and no regret. In *Conf. Learning Theory*, pages 533–557. PMLR, 2019.
- [44] D. Calvetti, P. C. Hansen, and L. Reichel. L-curve curvature bounds via Lanczos bidiagonalization. *Electron. Trans. Numer. Anal.*, 14:20–35, 2002.

- [45] D. Calvetti and E. Somersalo. Inverse problems: From regularization to Bayesian inference. *Wiley Interdiscip. Rev.: Comput. Stat.*, 10(3):e1427, 2018.
- [46] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Found. Comput. Math.*, 9(6):717–772, 2009.
- [47] C. Cartis, J. Fiala, and Z. Shao. Hashing embeddings of optimal dimension, with applications to linear least squares. *arXiv:2105.11815*, 2021.
- [48] C. Cartis, J. Fowkes, and Z. Shao. Randomised subspace methods for non-convex optimization, with applications to nonlinear least-squares. *arXiv:2211.09873*, 2022.
- [49] C. Cartis, E. Massart, and A. Otemissov. Bound-constrained global optimization of functions with low effective dimensionality using multiple random embeddings. *Math. Program.*, 198(1):997–1058, 2023.
- [50] C. Cartis, E. Massart, and A. Otemissov. Global optimization using random embeddings. *Math. Program.*, 200(2):781–829, 2023.
- [51] K. Chadán, D. Colton, L. Päiväranta, and W. Rundell. *An introduction to inverse scattering and inverse spectral problems*. SIAM, 1997.
- [52] X.-W. Chang, C. C. Paige, and D. Titley-Péloquin. Stopping criteria for the iterative solution of linear least squares problems. *SIAM J. Mat. Anal. Appl.*, 31(2):831–852, 2009.
- [53] S. Chatterjee and A. S. Hadi. *Regression analysis by example*. John Wiley & Sons, 2015.
- [54] S. Chen, Y. Liu, M. R. Lyu, I. King, and S. Zhang. Fast relative-error approximation algorithm for ridge regression. In *Proc. Conf. Uncert. Artif. Intel.*, 2015.
- [55] Y. Chen, E. N. Epperly, J. A. Tropp, and R. J. Webber. Randomly pivoted Cholesky: Practical approximation of a kernel matrix with few entry evaluations. *arXiv:2207.06503*, 2022.
- [56] Y. Chen and Y. Yang. Accumulations of projections — A unified framework for random sketches in kernel ridge regression. In *Proc. Int. Conf. Artif. Intel. Stat.*, volume 130, 2021.
- [57] S. Chenakkod, M. Dereziński, X. Dong, and M. Rudelson. Optimal embedding dimension for sparse subspace embeddings. In *Proc. Annu. ACM Symp. Theory Comput.*, pages 1106–1117, 2024.
- [58] H. Y. Cheung, T. C. Kwok, and L. C. Lau. Fast matrix rank algorithms and applications. *J. ACM*, 60(5):1–25, 2013.
- [59] A. Chowdhury, J. Yang, and P. Drineas. An iterative, sketching-based framework for ridge regression. In *ICML*, volume 3, pages 1595–1626, 2018.
- [60] J. Chung and S. Gazzola. Computational methods for large-scale inverse problems: A survey on hybrid projection methods. *SIAM Rev.*, 66(2):205–284, 2024.
- [61] K. L. Clarkson and D. P. Woodruff. Low-rank approximation and regression in input sparsity time. *J. ACM*, 63(6):1–45, 2017.

- [62] A. Cohen, M. A. Davenport, and D. Leviatan. On the stability and accuracy of least squares approximations. *Found. Comput. Math.*, 13(5):819–834, 2013.
- [63] M. B. Cohen. Nearly tight oblivious subspace embeddings by trace inequalities. *Proc. Annu. ACM-SIAM Symp. Discrete Algorithms*, pages 278–287, 2016.
- [64] M. B. Cohen, S. Elder, C. Musco, C. Musco, and M. Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proc. Annu. ACM Symp. Theory Comput.*, pages 163–172, 2015.
- [65] D. L. Colton and R. Kress. *Inverse acoustic and electromagnetic scattering theory*, volume 93. Springer, 1998.
- [66] M. P. Connolly, N. J. Higham, and S. Pranesh. Randomized low rank matrix approximation: Rounding error analysis and a mixed precision algorithm. *MIMS EPrint:2022.10*, 2022.
- [67] V. Coppé, D. Huybrechs, R. Matthysen, and M. Webb. The AZ algorithm for least squares systems with a known incomplete generalized inverse. *SIAM J. Mat. Anal. Appl.*, 41(3):1237–1259, 2020.
- [68] A. Cortinovis and D. Kressner. On randomized trace estimates for indefinite matrices with an application to determinants. *Found. Comput. Math.*, pages 1–29, 2022.
- [69] A. Cortinovis, D. Kressner, and Y. Nakatsukasa. Speeding up Krylov subspace methods for computing via randomization. *SIAM J. Matrix Anal. Appl.*, 45(1):619–633, 2024.
- [70] A. Cultrera and L. Callegaro. A simple algorithm to find the L-curve corner in the regularization of ill-posed inverse problems. *IOP SciNotes*, 1(2):025004, 2020.
- [71] M. A. Davenport and J. Romberg. An overview of low-rank matrix recovery from incomplete observations. *IEEE J. Sel. Top. Signal Process.*, 10(4):608–622, 6 2016.
- [72] K. R. Davidson and S. J. Szarek. Local operator theory, random matrices and Banach spaces. In *Handbook of the geometry of Banach spaces*, volume 1, pages 330–360. Elsevier, 2001.
- [73] M. E. Davies and Y. C. Eldar. Rank awareness in joint sparse recovery. *IEEE Trans. Inf. Theory*, 58(2):1135–1146, 2 2012.
- [74] T. A. Davis and Y. Hu. The University of Florida sparse matrix collection. *ACM Trans. Math. Softw.*, 38(1):1–25, 2011.
- [75] A. M. DeGennaro and N. M. Urban. Scalable extended dynamic mode decomposition using random kernel approximation. *SIAM J. Sci. Comput.*, 41(3):A1482–A1499, 2019.
- [76] M. Dereziński and M. W. Mahoney. Recent and upcoming developments in randomized numerical linear algebra for machine learning. *arXiv:2406.11151*, 2024.
- [77] M. Dereziński, C. Musco, and J. Yang. Faster linear systems and matrix norm approximation via multi-level sketched preconditioning. *arXiv:2405.05865*, 2024.
- [78] M. Dereziński and J. Yang. Solving dense linear systems faster than via preconditioning. In *Proc. Annu. ACM Symp. Theory Comput.*, pages 1118–1129, 2024.

- [79] A. Deshpande, L. Rademacher, S. S. Vempala, and G. Wang. Matrix approximation and projective clustering via volume sampling. *Theory Comput.*, 2(1):225–247, 2006.
- [80] A. Deshpande and S. Vempala. Adaptive sampling and fast low-rank matrix approximation. In *APPROX/RANDOM*, pages 292–303. Springer, 2006.
- [81] G. Dhatt, E. Lefrançois, and G. Touzot. *Finite element method*. John Wiley & Sons, 2012.
- [82] E. Di Napoli, E. Polizzi, and Y. Saad. Efficient estimation of eigenvalue counts in an interval. *Numer. Lin. Alg. Apply.*, 23(4):674–692, 2016.
- [83] Y. Dong and P.-G. Martinsson. Simpler is better: A comparative study of randomized pivoting algorithms for CUR and interpolative decompositions. *Adv. Comput. Math.*, 49(4):66, 2023.
- [84] P. Drineas, R. Kannan, and M. W. Mahoney. Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication. *SIAM J. Comput.*, 36(1):132–157, 2006.
- [85] P. Drineas, R. Kannan, and M. W. Mahoney. Fast Monte Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix. *SIAM J. Comput.*, 36(1):158–183, 2006.
- [86] P. Drineas, R. Kannan, and M. W. Mahoney. Fast Monte Carlo algorithms for matrices III: Computing a compressed approximate matrix decomposition. *SIAM J. Comput.*, 36(1):184–206, 2006.
- [87] P. Drineas, M. Magdon-Ismail, M. W. Mahoney, and D. P. Woodruff. Fast approximation of matrix coherence and statistical leverage. *J. Mach. Learn. Res.*, 13(1):3475–3506, 2012.
- [88] P. Drineas and M. W. Mahoney. RandNLA: Randomized numerical linear algebra. *Commun. ACM*, 59(6):80–90, 2016.
- [89] P. Drineas and M. W. Mahoney. Lectures on randomized numerical linear algebra. In *The Mathematics of Data*, volume 25. AMS, 2018.
- [90] P. Drineas, M. W. Mahoney, and S. Muthukrishnan. Sampling algorithms for l_2 regression and applications. In *Proc. Annu. ACM-SIAM Symp. Discrete Algorithms*, pages 1127–1136, 2006.
- [91] P. Drineas, M. W. Mahoney, and S. Muthukrishnan. Subspace sampling and relative-error matrix approximation: Column-based methods. In *APPROX/RANDOM*, pages 316–326. Springer, 2006.
- [92] P. Drineas, M. W. Mahoney, and S. Muthukrishnan. Relative-error CUR matrix decompositions. *SIAM J. Matrix Anal. Appl.*, 30(2):844–881, 2008.
- [93] P. Drineas, M. W. Mahoney, S. Muthukrishnan, and T. Sarlós. Faster least squares approximation. *Numer. Math.*, 117(2):219–249, 2011.
- [94] J. A. Duersch and M. Gu. Randomized projection for rank-revealing matrix factorizations and low-rank approximations. *SIAM Rev.*, 62(3):661–682, 2020.
- [95] A. Edelman and Y. Wang. The GSVD: Where are the ellipses?, matrix trigonometry, and more. *SIAM J. Mat. Anal. Appl.*, 41(4):1826–1856, 2020.

- [96] A. El Alaoui, F. Krzakala, and M. Jordan. Fundamental limits of detection in the spiked Wigner model. *Ann. Stat.*, 48(2):863–885, 2020.
- [97] N. El Karoui. Spectrum estimation for large dimensional covariance matrices using random matrix theory. *Ann. Stat.*, 36(6):2757–2790, 2008.
- [98] L. Elden. A note on the computation of the generalized cross-validation function for ill-conditioned least squares problems. *BIT Num. Math.*, 24:467–472, 1984.
- [99] H. W. Engl, M. Hanke, and A. Neubauer. *Regularization of inverse problems*, volume 375. Springer Science & Business Media, 1996.
- [100] E. N. Epperly. Fast and forward stable randomized algorithms for linear least-squares problems. *SIAM J. Matrix Anal. Appl.*, to appear, 2024.
- [101] E. N. Epperly, M. Meier, and Y. Nakatsukasa. Fast randomized least-squares solvers can be just as accurate and stable as classical direct solvers. *arXiv:2406.03468*, 2024.
- [102] N. B. Erichson, L. Mathelin, J. N. Kutz, and S. L. Brunton. Randomized dynamic mode decomposition. *SIAM J. Appl. Dyn.*, 18(4):1867–1891, 2019.
- [103] D. C.-L. Fong and M. Saunders. LSMR: An iterative algorithm for sparse least-squares problems. *SIAM J. Sci. Comput.*, 33(5):2950–2971, 2011.
- [104] A. Frieze, R. Kannan, and S. Vempala. Fast Monte-Carlo algorithms for finding low-rank approximations. *J. ACM*, 51(6):1025–1041, 2004.
- [105] S. Garg, K. Tan, and M. Dereziński. Distributed least squares in small space via sketching and bias reduction. *arXiv:2405.05343*, 2024.
- [106] N. Gazagnadou, M. Ibrahim, and R. M. Gower. RidgeSketch: A fast sketching based solver for large scale ridge regression. *arXiv:2105.05565*, (1):1–33, 2021.
- [107] S. Gazzola, P. C. Hansen, and J. G. Nagy. IR Tools: a MATLAB package of iterative regularization methods and large-scale test problems. *Numer. Algorithms*, 81(3):773–811, 2019.
- [108] S. Gazzola and M. Sabaté Landman. Krylov methods for inverse problems: Surveying classical, and introducing new, algorithmic approaches. *GAMM-Mitteilungen*, 43(4):e202000017, 2020.
- [109] A. Gittens, J. Kottalam, J. Yang, M. F. Rungenburg, J. Chhugani, E. Racah, M. Singh, Y. Yao, C. Fischer, O. Ruebel, et al. A multi-platform evaluation of the randomized CX low-rank matrix factorization in Spark. In *IEEE Int. Paral. Distrib. Process. Symp. Works.*, pages 1403–1412. IEEE, 2016.
- [110] A. Gittens and M. W. Mahoney. Revisiting the Nyström method for improved large-scale machine learning. *J. Mach. Learn. Res.*, 17:1–65, 2016.
- [111] A. Gittens and J. A. Tropp. Tail bounds for all eigenvalues of a sum of random matrices. *arXiv:1104.4513*, 2011.
- [112] G. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *SIAM J. Numer. Anal.*, 2(2):205–224, 1965.

- [113] G. H. Golub, M. Heath, and G. Wahba. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2):215–223, 1979.
- [114] G. H. Golub and C. F. Van Loan. *Matrix computations*. Johns Hopkins University Press, 4th edition, 2012.
- [115] N. Gould and J. Scott. The state-of-the-art of preconditioners for sparse linear least-squares problems. *ACM Trans. Math. Softw.*, 43(4):1–35, 2017.
- [116] L. Grasedyck and W. Hackbusch. Construction and arithmetics of H-matrices. *Computing (Vienna/New York)*, 70(4):295–334, 2003.
- [117] A. Greenbaum. Behavior of slightly perturbed Lanczos and conjugate-gradient recurrences. *Lin. Alg. Appl.*, 113:7–63, 1989.
- [118] A. Greenbaum. Estimating the attainable accuracy of recursively computed residual methods. *SIAM J. Mat. Anal. Appl.*, 18(3):535–551, 1997.
- [119] A. Greenbaum. *Iterative methods for solving linear systems*. SIAM, 1997.
- [120] A. Greenbaum and Z. Strakoš. Predicting the behavior of finite precision Lanczos and conjugate gradient computations. *SIAM J. Mat. Anal. Appl.*, 13(1):121–137, 1992.
- [121] C. W. Groetsch. *The theory of Tikhonov regularization for Fredholm equations*. Boston Pitman Publication, 1984.
- [122] C. W. Groetsch and A. Neubauer. Regularization of ill-posed problems: Optimal parameter choice in finite dimensions. *J. Approx. Theory*, 58(2):184–200, 1989.
- [123] L. Guo, A. Narayan, and T. Zhou. Constructing least-squares polynomial approximations. *SIAM Rev.*, 62(2):483–508, 2020.
- [124] Y. Gürsel and M. Tinto. Near optimal solution to the inverse problem for gravitational-wave bursts. *Phys. Rev. D*, 40(12):3884, 1989.
- [125] S. Güttel and M. Schweitzer. Randomized sketching for Krylov approximations of large-scale matrix functions. *SIAM J. Mat. Anal. Appl.*, 44(3):1073–1095, 2023.
- [126] N. Halko, P.-G. Martinsson, Y. Shkolnisky, and M. Tygert. An algorithm for the principal component analysis of large data sets. *SIAM J. Sci. Comput.*, 33(5):2580–2594, 2011.
- [127] N. Halko, P.-G. Martinsson, and J. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 53(2):217–288, 2011.
- [128] E. Hallman. *Error Estimates for Least-Squares Problems*. PhD thesis, University of California, Berkeley, 2019.
- [129] E. Hallman. Estimating the backward error for the least-squares problem with multiple right-hand sides. *Lin. Alg. Appl.*, 605:227–238, 2020.

- [130] P. Hansen. *Discrete Inverse Problems: Insight and Algorithms*. Fundamentals of Algorithms. SIAM, 2010.
- [131] P. C. Hansen. Analysis of discrete ill-posed problems by means of the L-curve. *SIAM Rev.*, 34(4):561–580, 1992.
- [132] P. C. Hansen. *Rank-deficient and discrete ill-posed problems: Numerical aspects of linear inversion*. SIAM, 1998.
- [133] P. C. Hansen. The L-curve and its use in the numerical treatment of inverse problems. In *Computational Inverse Problems in Electrocardiology*, pages 119–142. WIT Press, 2001.
- [134] P. C. Hansen. Regularization tools version 4.0 for Matlab 7.3. *Numerical algorithms*, 46:189–194, 2007.
- [135] P. C. Hansen, J. Jørgensen, and W. R. Lionheart. *Computed tomography: Algorithms, insight, and just enough theory*. SIAM, 2021.
- [136] P. C. Hansen and D. P. O’Leary. The use of the L-curve in the regularization of discrete ill-posed problems. *SIAM J. Sci. Comput.*, 14(6):1487–1503, 1993.
- [137] P. C. Hansen, V. Pereyra, and G. Scherer. *Least squares data fitting with applications*. Johns Hopkins University Press, 2013.
- [138] G. T. Herman, A. Lent, and H. Hurwitz. A storage-efficient algorithm for finding the regularized solution of a large, inconsistent system of equations. *IMA J. Appl. Math.*, 25(4):361–366, 1980.
- [139] M. R. Hestenes, E. Stiefel, et al. *Methods of conjugate gradients for solving linear systems*, volume 49. NBS, 1952.
- [140] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, 2002.
- [141] N. J. Higham and T. Mary. A new approach to probabilistic rounding error analysis. *SIAM J. Sci. Comput.*, 41(5):A2815–A2835, 2019.
- [142] N. J. Higham and T. Mary. Sharper probabilistic backward error analysis for basic linear algebra kernels with random data. *SIAM J. Sci. Comput.*, 42(5):A3427–A3446, 2020.
- [143] N. J. Higham and G. Stewart. Numerical linear algebra in statistical computing. In *The state of the art in numerical analysis*, pages 41–57. Oxford University Press, 1987.
- [144] T. M. Inc. Matlab version: 9.13.0 (r2022b), 2022.
- [145] I. C. Ipsen. *Numerical matrix analysis: Linear systems and least squares*. SIAM, 2009.
- [146] I. C. F. Ipsen and T. Wentworth. The effect of coherence on sampling from matrices with orthonormal columns, and preconditioned least squares problems. *SIAM J. Mat. Anal. Appl.*, 35(4):1490–1520, 2014.
- [147] V. Isakov. On inverse problems in secondary oil recovery. *Eur. J. Appl. Math.*, 19(4):459–478, 2008.

- [148] C. Iyer, H. Avron, G. Kollias, Y. Ineichen, C. Carothers, and P. Drineas. A scalable randomized least squares solver for dense overdetermined systems. In *Proc. ScalA*, 2015.
- [149] C. Iyer, C. Carothers, and P. Drineas. Randomized sketching for large-scale sparse ridge regression problems. In *Proc. ScalA*, pages 65–72, 2017.
- [150] Z. Jia and Y. Yang. Modified truncated randomized singular value decomposition (MTRSVD) algorithms for large scale discrete ill-posed problems with general-form regularization. *Inverse Probl.*, 34(5):055013, 2018.
- [151] S. Jiang, D. Li, I. M. Li, A. V. Mahankali, and D. Woodruff. Streaming and distributed algorithms for robust column subset selection. In *ICML*, pages 4971–4981. PMLR, 2021.
- [152] W. B. Johnson, J. Lindenstrauss, and G. Schechtman. Extensions of Lipschitz maps into Banach spaces. *Isr. J. Math.*, 54(2):129–138, 1986.
- [153] I. M. Johnstone. On the distribution of the largest eigenvalue in principal components analysis. *Ann. Stat.*, pages 295–327, 2001.
- [154] D. M. Kane and J. Nelson. Sparser Johnson-Lindenstrauss transforms. *J. ACM*, 61(1):1–23, 2014.
- [155] S. Karczmarz. Angenaherte auflösung von systemen linearer glei-chungen. *Bull. Int. Acad. Pol. Sic. Let., Cl. Sci. Math. Nat.*, pages 355–357, 1937.
- [156] G. Kollias, H. Avron, Y. Ineichen, C. Bekas, A. Curioni, V. Sindhwani, and K. Clarkson. libSkylark: A framework for high-performance matrix sketching for statistical computing. In *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, 2015.
- [157] V. Koltchinskii and K. Lounici. Concentration inequalities and moment bounds for sample covariance operators. *Bernoulli*, 23(1):110–133, 2017.
- [158] J. Lacotte and M. Pilanci. Effective dimension adaptive sketching methods for faster regularized least-squares optimization. *Adv. Neural Inf. Process. Syst.*, 23(NeurIPS):1–11, 2020.
- [159] J. Lacotte and M. Pilanci. Faster least squares optimization. *arXiv:1911.02675*, Apr. 2021.
- [160] J. Lacotte, M. Pilanci, and M. Pavone. High-dimensional optimization in adaptive random subspaces. *Advances in Neural Information Processing Systems*, 32, 2019.
- [161] O. Ledoit and M. Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *J. Multivar. Anal.*, 88(2):365–411, 2004.
- [162] O. Ledoit and M. Wolf. Nonlinear shrinkage estimation of large-dimensional covariance matrices. *Ann. Stat.*, 40(2):1024–1060, 2012.
- [163] O. Ledoit and M. Wolf. Analytical nonlinear shrinkage of large-dimensional covariance matrices. *Ann. Stat.*, 48(5):3043–3065, 2020.
- [164] A. Li, S. Becker, and A. Doostan. Online randomized interpolative decomposition with a posteriori error estimator for temporal pde data reduction. *arXiv:2405.16076*, 2024.

- [165] M. Li, W. Bi, J. T. Kwok, and B.-L. Lu. Large-scale Nyström kernel matrix approximation using randomized SVD. *IEEE Trans. Neural Netw. Learn. Syst.*, 26(1):152–164, 2014.
- [166] N. Li and Y. Saad. MIQR: A multilevel incomplete QR preconditioner for large sparse least-squares problems. *SIAM J. Mat. Anal. Appl.*, 28(2):524–550, 2006.
- [167] E. Liberty, F. Woolfe, P.-G. Martinsson, V. Rokhlin, and M. Tygert. Randomized algorithms for the low-rank approximation of matrices. *Proc. Natl. Acad. Sci.*, 104(51):20167–20172, 2007.
- [168] J. Liesen and Z. Strakoš. *Krylov subspace methods principles and analysis*. Oxford University Press, 2012.
- [169] L. Lin, Y. Saad, and C. Yang. Approximating spectral densities of large matrices. *SIAM Rev.*, 58(1):34–65, 2016.
- [170] Y. Lu, P. Dhillon, D. P. Foster, and L. Ungar. Faster ridge regression via the subsampled randomized Hadamard transform. *Adv. Neural Inf. Process. Syst.*, 26, 2013.
- [171] L. Lucy. Optimum strategies for inverse problems in statistical astronomy. *A&A*, 289(3):983–994, 1994.
- [172] N. Luiken and T. van Leeuwen. Comparing RSVD and Krylov methods for linear inverse problems. *Comput. Geosci.*, 137:104427, 2020.
- [173] M. Mahoney, J. Duchi, and A. Gilbert. *The mathematics of data*. IAS/Park City Mathematics Series. American Mathematical Society, 2018.
- [174] M. W. Mahoney. Randomized algorithms for matrices and data. *Found. Trends Mach. Learn.*, 3(2):123–224, 2011.
- [175] V. A. Marčenko and L. A. Pastur. Distribution of eigenvalues for some sets of random matrices. *Mat. Sb.*, 1(4):457–483, 1967.
- [176] P.-G. Martinsson. *Fast direct solvers for elliptic PDEs*. SIAM, 2019.
- [177] P.-G. Martinsson, G. Quintana-Orti, and N. Heavner. randUTV: A blocked randomized algorithm for computing a rank-revealing UTV factorization. *ACM Trans. Math. Soft.*, 45(1):1–26, 2019.
- [178] P.-G. Martinsson, V. Rokhlin, and M. Tygert. A randomized algorithm for the approximation of matrices. *Tech. rep. YALEU/DCS/TR-1361*, 2006.
- [179] P.-G. Martinsson, V. Rokhlin, and M. Tygert. A randomized algorithm for the decomposition of matrices. *Appl. Comput. Harmon. Anal.*, 30(1):47–68, 2011.
- [180] P.-G. Martinsson, A. Szlam, and M. Tygert. Normalized power iterations for the computation of SVD. In *Proc. of NIPS Workshop on Low-Rank Methods for Large-Scale Machine Learning*, 2011.
- [181] P.-G. Martinsson and J. A. Tropp. Randomized numerical linear algebra: Foundations and algorithms. *Acta Numer.*, 29:403–572, 2020.

- [182] P.-G. Martinsson and S. Voronin. A randomized blocked algorithm for efficiently computing rank-revealing factorizations of matrices. *SIAM J. Sci. Comput.*, 38(5):S485–S507, 2016.
- [183] R. Mathias. Two theorems on singular values and eigenvalues. *Am. Math. Mon.*, 97(1):47–50, 1990.
- [184] M. Meier and Y. Nakatsukasa. Randomized algorithms for Tikhonov regularization in linear least squares. *arXiv:2203.07329*, 2022.
- [185] M. Meier and Y. Nakatsukasa. Fast randomized numerical rank estimation for numerically low-rank matrices. *Lin. Alg. Appl.*, 686:1–32, 2024.
- [186] M. Meier, Y. Nakatsukasa, A. Townsend, and M. Webb. Are sketch-and-precondition least squares solvers numerically stable? *SIAM J. Mat. Anal. Appl.*, 45(2):905–929, 2024.
- [187] X. Meng, M. A. Saunders, and M. W. Mahoney. LSRN: A parallel iterative solver for strongly over-or underdetermined systems. *SIAM J. Sci. Comput.*, 36(2):C95–C118, 2014.
- [188] G. Meurant and Z. Strakoš. The Lanczos and conjugate gradient algorithms in finite precision arithmetic. *Acta Numer.*, 15:471–542, 2006.
- [189] R. A. Meyer, C. Musco, C. Musco, and D. P. Woodruff. Hutch++: Optimal stochastic trace estimation. In *SOSA*, pages 142–155. SIAM, 2021.
- [190] V. Morozov. *Methods for solving incorrectly posed problems*. Springer Science & Business Media, 2012.
- [191] A. Müller, W. Deconinck, C. Kühnlein, G. Mengaldo, M. Lange, N. Wedi, P. Bauer, P. K. Smolarkiewicz, M. Diamantakis, S.-J. Lock, et al. The ESCAPE project: energy-efficient scalable algorithms for weather prediction at exascale. *Geosci. Model Dev.*, 12(10):4425–4441, 2019.
- [192] R. Murray, J. Demmel, M. W. Mahoney, N. B. Erichson, M. Melnichenko, O. A. Malik, L. Grigori, P. Luszczek, M. Dereziński, M. E. Lopes, et al. Randomized numerical linear algebra: A perspective on the field with an eye to software. *arXiv:2302.11474*, 2023.
- [193] C. Musco, C. Musco, and A. Sidford. Stability of the Lanczos method for matrix function approximation. In *Proc. Annu. ACM-SIAM Symp. Discrete Algorithms*, pages 1605–1624. SIAM, 2018.
- [194] R. Muthukumar, D. P. Kouri, and M. Udell. Randomized sketching algorithms for low-memory dynamic optimization. *SIAM J. Optim.*, 31(2):1242–1275, 2021.
- [195] B. Nadler. Finite sample approximation results for principal component analysis: A matrix perturbation approach. *Ann. Stat.*, 36(6):2791–2817, 2008.
- [196] Y. Nakatsukasa. Fast and stable randomized low-rank matrix approximation. *arXiv:2009.11392*, 2020.
- [197] Y. Nakatsukasa and J. A. Tropp. Fast and accurate randomized algorithms for linear systems and eigenvalue problems. *SIAM J. Mat. Anal. Appl.*, 45(2):1183–1214, 2024.

- [198] F. Natterer. *The mathematics of computerized tomography*. SIAM, 2001.
- [199] F. Natterer and F. Wübbeling. *Mathematical methods in image reconstruction*. SIAM, 2001.
- [200] I. K. Ozaslan, M. Pilanci, and O. Arikan. Iterative Hessian Sketch with Momentum. In *Proc. IEEE Int. Conf. Acoust. Speech Signal Process*, pages 7470–7474. IEEE, 5 2019.
- [201] I. K. Ozaslan, M. Pilanci, and O. Arikan. M-IHS: An accelerated randomized preconditioning method avoiding costly matrix decompositions. *Lin. Alg. Appl.*, 678:57–91, 2023.
- [202] C. C. Paige. *The computation of Eigenvalues and Eigenvectors of Very Large Sparse Matrices*. PhD thesis, University of London, 1971.
- [203] C. C. Paige. Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem. *Lin. Alg. Appl.*, 34:235–258, 1980.
- [204] C. C. Paige and M. A. Saunders. Towards a generalized singular value decomposition. *SIAM J. Numer. Anal.*, 18(3):398–405, 1981.
- [205] C. C. Paige and M. A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Soft.*, 8(1):43–71, 1982.
- [206] R. Palm. *Numerical comparison of regularization algorithms for solving ill-posed problems*. PhD thesis, Tartu University Press, 2010.
- [207] C. H. Papadimitriou, H. Tamaki, P. Raghavan, and S. Vempala. Latent semantic indexing: A probabilistic analysis. In *Proc. ACM SIGACT-SIGMOD-SIGART Symp. Princ. Database Syst.*, pages 159–168, 1998.
- [208] T. Park and Y. Nakatsukasa. Accuracy and stability of CUR decompositions with oversampling. *arXiv:2405.06375*, 2024.
- [209] P. Parker, P. J. Wolfe, and V. Tarokh. A signal processing application of randomized low-rank approximations. In *IEEE Workshop Stat. Signal Process.*, pages 345–350. IEEE, 2005.
- [210] M. M. A. Patwary, S. Byna, N. R. Satish, N. Sundaram, Z. Lukić, V. Roytershteyn, M. J. Anderson, Y. Yao, Prabhat, and P. Dubey. BD-CATS: big data clustering at trillion particle scale. In *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, pages 1–12, 2015.
- [211] D. Paul. Asymptotics of sample eigenstructure for a large dimensional spiked covariance model. *Stat. Sin.*, pages 1617–1642, 2007.
- [212] D. Persson, N. Boullé, and D. Kressner. Randomized Nyström approximation of non-negative self-adjoint operators. *arXiv:2404.00960*, 2024.
- [213] D. Persson and D. Kressner. Randomized low-rank approximation of monotone matrix functions. *SIAM J. Mat. Anal. Appl.*, 44(2):894–918, 2023.
- [214] D. L. Phillips. A technique for the numerical solution of certain integral equations of the first kind. *J. ACM*, 9(1):84–97, 1962.

- [215] M. Pilanci and M. J. Wainwright. Randomized sketches of convex programs with sharp guarantees. *IEEE Trans. Inf. Theory*, 61(9):5096–5115, 2015.
- [216] M. Pilanci and M. J. Wainwright. Iterative Hessian sketch: Fast and accurate solution approximation for constrained least-squares. *J. Mach. Learn. Res.*, 17:1–38, 2016.
- [217] J. Radon. Über die bestimmung von funktionen durch ihre integralwerte längs gewisser mannigfaltigkeiten. *Verh. Sachs. Akad. Wiss. Leipzig, Math Phys Klass*, 69, 1917.
- [218] D. Rajaram, C. Perron, T. G. Puranik, and D. N. Mavris. Randomized algorithms for non-intrusive parametric reduced order modeling. *AIAA Journal*, 58(12):5389–5407, 2020.
- [219] N. R. Rao, J. A. Mingo, R. Speicher, and A. Edelman. Statistical eigen-inference from large Wishart matrices. *Ann. Stat.*, 36(6):2850–2885, 2008.
- [220] T. Regińska. A regularization parameter in discrete ill-posed problems. *SIAM J. Sci. Comput.*, 17(3):740–749, 1996.
- [221] V. Rokhlin, A. Szlam, and M. Tygert. A randomized algorithm for principal component analysis. *SIAM J. Mat. Anal. Appl.*, 31(3):1100–1124, 2010.
- [222] V. Rokhlin and M. Tygert. A fast randomized algorithm for overdetermined linear least-squares regression. *Proc. Nat. Acad. Sci.*, 105(36):13212–13217, 2008.
- [223] M. Rudelson and R. Vershynin. Non-asymptotic theory of random matrices: Extreme singular values. In *Proc. Int. Cong. Math.*, pages 1576–1602, 2010.
- [224] Y. Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.
- [225] A. K. Saibaba, A. Alexanderian, and I. C. Ipsen. Randomized matrix-free trace and log-determinant estimators. *Numer. Math.*, 137(2):353–395, 2017.
- [226] A. K. Saibaba, J. Hart, and B. van Bloemen Waanders. Randomized algorithms for generalized singular value decomposition with application to sensitivity analysis. *Numer. Lin. Alg. Appl.*, 28(4):e2364, 2021.
- [227] A. K. Saibaba and A. Miedlar. Randomized low-rank approximations beyond Gaussian random matrices. *arXiv:2308.05814*, 2023.
- [228] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *Proc. IEEE Int. Conf. Acoust. Speech Signal Process*, pages 6655–6659, 10 2013.
- [229] T. Sarlóš. Improved approximation algorithms for large matrices via random projections. In *IEEE Symp. Found. Comp. Sci.*, pages 143–152. IEEE, 2006.
- [230] J. Scott and M. Tuma. On positive semidefinite modification schemes for incomplete Cholesky factorization. *SIAM J. Sci. Comput.*, 36(2):A609–A633, 2014.
- [231] H. Shen and J. Z. Huang. Sparse principal component analysis via regularized low rank matrix approximation. *J. Multivar. Anal.*, 99(6):1015–1034, 7 2008.

- [232] L. A. Shepp and B. F. Logan. The Fourier reconstruction of a head section. *IEEE Trans. Nucl. Sci.*, 21(3):21–43, 1974.
- [233] J. Sherman and W. J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *Ann. Math. Statist.*, 21:124–127, 1950.
- [234] A. Sobczyk and E. Gallopoulos. Estimating leverage scores via rank revealing methods and randomization. *SIAM J. Mat. Anal. Appl.*, 42(3):1199–1228, 2021.
- [235] A. Sobczyk and E. Gallopoulos. pylspack: Parallel algorithms and data structures for sketching, column subset selection, regression, and leverage scores. *ACM Trans. Math. Softw.*, 48(4):1–27, 2022.
- [236] R. S. Srinivasa, M. A. Davenport, and J. Romberg. Localized sketching for matrix multiplication and ridge regression. *arXiv:2003.09097*, 2020.
- [237] G. W. Stewart. An inverse perturbation theorem for the linear least squares problem. *ACM SIGNUM Newsletter*, 10(2-3):39–40, 1975.
- [238] G. W. Stewart. Research, development, and LINPACK. In *ACM Trans. Math. Softw.*, pages 1–14. Elsevier, 1977.
- [239] Z. Strakoš. On the real convergence rate of the conjugate gradient method. *Lin. Alg. Appl.*, 154:535–549, 1991.
- [240] T. Strohmer and R. Vershynin. A randomized Kaczmarz algorithm with exponential convergence. *J. Fourier Anal. Appl.*, 15(2):262–278, 2009.
- [241] A. N. Subrahmanya, V. Rao, and A. K. Saibaba. Randomized preconditioned solvers for strong constraint 4D-Var data assimilation. *arXiv:2401.15758*, 2024.
- [242] Y. Sun, Y. Guo, C. Luo, J. Tropp, and M. Udell. Low-rank Tucker approximation of a tensor from streaming data. *SIAM J. Math. Data. Sci.*, 2(4):1123–1150, 2020.
- [243] A. Tarantola and B. Valette. Generalized nonlinear inverse problems solved using the least squares criterion. *Rev. Geophys.*, 20(2):219–232, 1982.
- [244] A. Tarantola and B. Valette. Inverse problems = quest for information. *J. Geophys. Res.*, 50(1):159–170, 1982.
- [245] A. M. Thompson, J. C. Brown, J. W. Kay, and D. M. Titterton. A study of methods of choosing the smoothing parameter in image restoration by regularization. *IEEE Trans. Pattern Anal. Mach.*, 13(04):326–339, 1991.
- [246] A. N. Tikhonov. Regularization of incorrectly posed problems. *Soviet Mathematics Doklady*, 1963.
- [247] A. N. Tikhonov and V. Y. Arsenin. *Solutions of ill-posed problems*. V. H. Winston & Sons, 1977.
- [248] P. A. Toft. *The Radon transform — theory and implementation*. PhD thesis, Technical University of Denmark, 1996.

- [249] L. N. Trefethen. *Approximation theory and approximation practice, extended edition*. SIAM, 2019.
- [250] L. N. Trefethen and D. Bau. *Numerical linear algebra*. SIAM, 1997.
- [251] J. A. Tropp. Improved analysis of the subsampled randomized Hadamard transform. *Adv. Adapt. Data Anal.*, 3(01n02):115–126, 2011.
- [252] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher. Practical sketching algorithms for low-rank matrix approximation. *SIAM J. Mat. Anal. Appl.*, 38(4):1454–1485, 2017.
- [253] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher. Streaming low-rank matrix approximation with an application to scientific simulation. *SIAM J. Sci. Comput.*, 41(4):A2430–A2463, 2019.
- [254] S. Ubaru and Y. Saad. Fast methods for estimating the numerical rank of large matrices. In *ICML*, pages 468–477. PMLR, 2016.
- [255] S. Ubaru, Y. Saad, and A.-K. Seghouane. Fast estimation of approximate matrix ranks using spectral densities. *Neural Comput.*, 29(5):1317–1351, may 2017.
- [256] S. Ubaru, A.-K. Seghouane, and Y. Saad. Find the dimension that counts: Fast dimension estimation and Krylov PCA. In *SIAM Int. Conf. Data Min. SDM 2019*, pages 720–728. SIAM, 2019.
- [257] M. Udell and A. Townsend. Why are big data matrices approximately low rank? *SIAM J. Math. Data. Sci.*, 1(1):144–160, 2019.
- [258] S. Vatankehah, R. A. Renaut, and V. E. Ardestani. Total variation regularization of the 3-D gravity inverse problem using a randomized generalized singular value decomposition. *Geophys. J. Int.*, 213(1):695–705, 2018.
- [259] R. Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv:1011.3027*, 2010.
- [260] R. Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge University Press, 2018.
- [261] C. R. Vogel. *Computational methods for inverse problems*. SIAM, 2002.
- [262] G. Wahba. *Spline models for observational data*. SIAM, 1990.
- [263] M. J. Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*. Cambridge University Press, 2019.
- [264] B. Waldén, R. Karlson, and J.-G. Sun. Optimal backward perturbation bounds for the linear least squares problem. *Numer. Lin. Alg. Appl.*, 2(3):271–286, 1995.
- [265] D. Wang and J. Xu. Large scale constrained linear regression revisited: Faster algorithms via preconditioning. In *Proc. AAAI Conf. Artif. Intell.*, volume 32, 2018.
- [266] D. Wang and J. Xu. Faster constrained linear regression via two-step preconditioning. *Neurocomputing*, 364:280–296, 2019.

- [267] J. Wang, J. D. Lee, M. Mahdavi, M. Kolar, and N. Srebro. Sketching meets random projection in the dual: A provable recovery algorithm for big and high-dimensional data. *Electron. J. Stat.*, 11(2):4896–4944, 2017.
- [268] S. Wang, A. Gittens, and M. W. Mahoney. Sketched ridge regression: Optimization perspective, statistical perspective, and model averaging. *J. Mach. Learn. Res.*, 18:1–50, 2018.
- [269] S. Wang and Z. Zhang. Improving CUR matrix decomposition and the Nyström approximation via adaptive sampling. *J. Mach. Learn. Res. Res.*, 14(1):2729–2769, 2013.
- [270] A. Wathen. Some comments on preconditioning for normal equations and least squares. *SIAM Rev.*, 64(3):640–649, 2022.
- [271] A. J. Wathen and T. Rees. Chebyshev semi-iteration in preconditioning for problems including the mass matrix. *Electron. Trans. Numer. Anal.*, 34:125–135, 2008.
- [272] P.-Å. Wedin. Perturbation theory for pseudo-inverses. *BIT Num. Math.*, 13(2):217–232, 1973.
- [273] Y. Wei, P. Xie, and L. Zhang. Tikhonov regularization and randomized GSVD. *SIAM J. Mat. Anal. Appl.*, 37(2):649–675, 2016.
- [274] M. A. Woodbury. *Inverting modified matrices*. Department of Statistics, Princeton University, 1950.
- [275] D. P. Woodruff. Sketching as a tool for numerical linear algebra. *Found. Trends Theor. Comput. Sci.*, 10(1–2):1–157, 2014.
- [276] F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert. A fast randomized algorithm for the approximation of matrices. *Appl. Comput. Harmon. Anal.*, 25(3):335–366, 2008.
- [277] H. Xiang and J. Zou. Regularization with randomized SVD for large-scale discrete inverse problems. *Inverse Probl.*, 29(8):085008, 2013.
- [278] H. Xiang and J. Zou. Randomized algorithms for large-scale inverse problems with general Tikhonov regularizations. *Inverse Probl.*, 31(8):085008, 2015.
- [279] M. Xu, R. Jin, and Z.-H. Zhou. CUR algorithm for partially observed matrices. In *ICML*, pages 1412–1421. PMLR, 2015.
- [280] M. J. Yaffe. Emergence of “big data” and its potential and current limitations in medical imaging. In *Sem. Nuc. Med.*, volume 49, pages 94–104. Elsevier, 2019.
- [281] Y. Yamamoto, Y. Nakatsukasa, Y. Yanagisawa, and T. Fukaya. Roundoff error analysis of the Cholesky QR2 algorithm. *Electron. Trans. Numer. Anal.*, 44:306–326, 2015.
- [282] J. Yang, X. Meng, and M. W. Mahoney. Implementing randomized matrix algorithms in parallel and distributed environments. *Proc. IEEE*, 104(1):58–92, 2015.
- [283] Y. Yang, M. Pilanci, and M. J. Wainwright. Randomized sketches for kernels: Fast and optimal nonparametric regression. *Ann. Stat.*, 45(3):991–1023, 2017.

- [284] W. Yu, Y. Gu, and Y. Li. Efficient randomized algorithms for the fixed-precision low-rank matrix approximation. *SIAM J. Mat. Anal. Appl.*, 39(3):1339–1359, 2018.
- [285] Y. Zhang, M. Wainwright, and M. Jordan. Distributed estimation of generalized matrix rank: Efficient algorithms and lower bounds. In *ICML*, pages 457–465. PMLR, 2015.
- [286] Y. Zhang and Y. Zhao. Astronomy in the big data era. *Data Sci. J.*, 14:11–11, 2015.
- [287] L. Zhou, S. Pan, J. Wang, and A. V. Vasilakos. Machine learning on big data: Opportunities and challenges. *Neurocomputing*, 237:350–361, 2017.