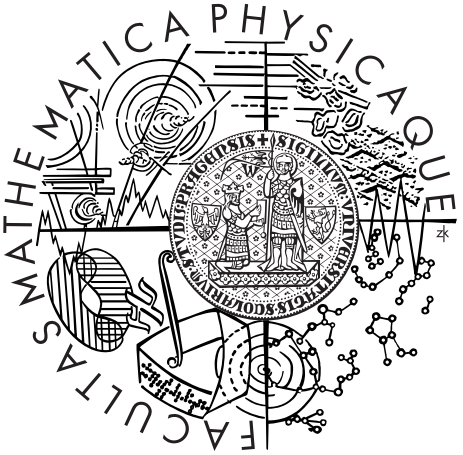
Charles University
Faculty of Mathematics and Physics

# HABILITATION THESIS




Erin Claire Carson

# Mixed Precision Matrix Computations: Analysis and Algorithms

Department of Numerical Mathematics

Prague 2023

WWW: http://www.karlin.mff.cuni.cz/~carson/
e-mail: carson@karlin.mff.cuni.cz

# Contents

# Preface

This thesis studies the topic of mixed precision matrix computations. In recent years, there has been a resurgence of interest in this topic due to the emerging commercial availability of low and mixed precision hardware, largely motivated by machine learning applications.

Computing with and moving fewer bits has obvious performance advantages. Compared to the standard IEEE double precision (64 bits), using IEEE half precision (16 bits) is up to 4× faster, and up to 16× faster using the specialized TensorCore instructions on NVIDIA V100 and A100 architectures, which perform a $4 \times 4$ matrix multiply in one clock cycle. In addition to computation time benefits, the use of lower precision can also speed up communication time, particularly in bandwidth-bound computations, as significantly fewer bits are being moved between nodes of the machine or levels of the memory hierarchy. Significant speedups are realized in practice [57].

At the time of writing, more than 1/3 of the machines on the TOP500 list [121] contain graphical processing units (GPUs) with mixed precision capabilities. This has motivated the development of a new supercomputing benchmark, HPL-MxP [71], which captures the performance gains possible when mixed precision is exploited. The HPL benchmark, traditionally used to rank the supercomputers in the TOP500 list, solves a dense linear system $Ax = b$ via Gaussian elimination with partial pivoting in double precision. In contrast, the HPL-MxP benchmark still solves a dense linear system to double precision accuracy, but instead accomplishes this by combining a low precision execution of Gaussian elimination with partial pivoting with an iterative refinement scheme. This benchmark and the underlying algorithm are based on the mixed precision algorithms developed in this thesis. In the most recent TOP500 results, the HPL-MxP benchmark obtains effective speedups of up to 9.5× versus the uniform precision HPL benchmark, demonstrating the benefits of mixed precision computation.

While low precision computation offers significant opportunities to improve performance, it must not be used blindly. One obvious drawback of using lower precision is that with fewer bits, we have greater roundoff error. Whereas a single computation in double precision gives a relative roundoff error bounded by approximately $10^{-16}$, for half precision this bound is around $10^{-4}$. Further, the range of representable numbers is smaller. This means that computations are more likely to encounter overflow and underflow.

The key idea behind the use of mixed precision in matrix computations is that we want to selectively use lower precision in the most computationally expensive parts of an algorithm while using higher precision parts in others, in order to enable both performance gains and an acceptable attainable accuracy. This venture requires rigorous mathematical analysis of finite precision matrix computations, which is the topic of

this thesis. Our goal will be to develop finite precision error analyses in a way that allows the elucidation of both potential dangers of and also opportunities for the use of mixed precision.

This thesis is comprised of 10 articles, 8 of which are published in high-impact journals, and 2 of which are currently under review, co-authored by Erin Carson in various collaborations, in particular with N. J. Higham (University of Manchester) and I. Yamazaki (Sandia National Laboratories). Other co-authors include S. Pranesh (V-Labs), B. Kelley (Sandia National Laboratories), and current and former postdoctoral researchers and Ph.D. students working under the direction of Erin Carson, including I. Daužickaitė, T. Gergelits, E. Oktay, and N. Khan.

We note that the present thesis represents a selected subset of the post-PhD work of the author. The author has also worked on other topics including the analysis and development of synchronization-reducing and communication-avoiding algorithms ([30] [28], [31], [29], and [119]), the stability of block orthogonalization methods ([24] and [25]), the mathematical and computational properties of Krylov subspace methods ([26] and [23]), as well as important survey articles [1].

We first present an overview of background material on mixed precision algorithms and numerical stability, iterative methods for solving linear systems, and least squares problems. This is followed by the technical portion of the thesis, which is divided into three parts. The first part of the thesis focuses on the development of mixed precision iterative refinement-based approaches for solving linear systems $Ax = b$. It includes the articles:

[C1] E. Carson and N. J. Higham, *A new analysis of iterative refinement and its application to accurate solution of ill-conditioned sparse linear systems*, SIAM Journal on Scientific Computing, 39 (2017), pp. A2834–A2856.
DOI: 10.1137/17M1122918

[C2] E. Carson and N. J. Higham, *Accelerating the solution of linear systems by iterative refinement in three precisions*, SIAM Journal on Scientific Computing, 40 (2018), pp. A817–A847.
DOI: 10.1137/17M1140819

[C3] E. Oktay and E. Carson, *Multistage mixed precision iterative refinement*, Numerical Linear Algebra with Applications, 29 (2022), e2434.
DOI: 10.1002/nla.2434

In iterative refinement, traditionally, an LU factorization of the matrix $A$ is used to compute the initial approximate solution $x_0$, which is then refined in each iteration by solving for a correction term; this solve is typically perfomed by reusing the already-computed LU factors. The paper [C1] develops a new two-precision iterative refinement approach based on the use of preconditioned Krylov subspace methods for the inner solve routine; the approach instead uses the already-computed LU factors as preconditioners for the GMRES method. In [C1], it is proved that this approach allows for the convergence of iterative refinement for more ill-conditioned matrices $A$ than the traditional approach.

Motivated by the inclusion of three or more hardware precisions in modern GPUs, the work [C2] developed and analyzed a general three-precision iterative refinement

scheme. The main idea is to use a potentially lower precision for the LU factorization (the most expensive part of the computation), a high precision for the residual computation, and a middle-ground precision as the working precision. This can result in substantial performance improvements while still obtaining forward and backward errors to the level of the working precision. This highly-cited work, together with [C1], significantly expanded and renewed interest in mixed precision numerical linear algebra. The works [C1] and [C2] also form the basis for the new HPL-MxP benchmark [71], which complements the TOP500 list [121].

Finally, it was observed experimentally that as an effect of the worst-case rounding error analysis in [C1], [C2], and related works (which is typical of finite precision error analyses), the resulting constraints on condition number were often too tight in practice. That is, iterative refinement often converges for more ill-conditioned matrices than guaranteed by the theory. Practically, this could result in one using a more costly variant of iterative refinement than necessary. As a solution to this problem, the work [C3] develops a multistage approach to mixed precision iterative refinement. The method begins with the least expensive variant and if slow convergence or divergence is detected using inexpensive monitoring, the algorithm switches to a more robust but more costly approach.

The second part of the thesis focuses on mixed precision iterative approaches for solving least squares problems. It includes the articles:

[C4] E. Carson, N. J. Higham, and S. Pranesh, *Three-precision GMRES-based iterative refinement for least squares problems*, SIAM Journal on Scientific Computing, 42 (2020), pp. A4063–A4083.
DOI: 10.1137/20M131682

[C5] E. Oktay and E. Carson, *Mixed precision Rayleigh quotient iteration for total least squares problems*, Numerical Algorithms, (2023).
DOI: 10.1007/s11075-023-01665-z

As for linear systems, it may also be desirable to perform iterative refinement in least squares problems to obtain a more accurate solution. In the case of highly incompatible least squares problems (i.e., when the residual $b - Ax$ is large), it becomes necessary to simultaneously refine the approximate solution and the residual. It was observed by Åke Björck that this can be achieved using what he calls the augmented system approach, which involves performing iterative refinement on a larger linear system equivalent to the normal equations [15]. In the article [C4], the approach of Björck is extended to the three-precision, Krylov subspace-based iterative refinement approaches developed in [C1] and [C2]. A key innovation is the development of a left-preconditioner for the augmented system composed of low-precision QR factors of $A$ and associated constraints under which GMRES-based iterative refinement is guaranteed to converge.

Unlike standard least squares problems, which are based on the standard linear model $Ax = b + r$, total least squares problems allow for errors in the matrix $A$ as well. That is, the total least squares problem is to solve $\min_{E,r} \|[E, r]\|_F$ subject to $(A + E)x = b + r$. This is commonly used in applications where the matrix $A$ itself is also subject to modeling or sampling errors. One approach appropriate for the case of

large, sparse problems is based on Rayleigh quotient iteration coupled with preconditioned conjugate gradient as an inner solver [17]. The article [C5] develops and analyzes a mixed precision variant of this approach, in which the expensive parts of the computation are performed in a precision potentially lower than the working precision. The experimental results indicate that while this can cause a slight delay in convergence rate of the Rayleigh quotient iteration, the mixed precision approach can achieve the same solution accuracy as the standard uniform precision approach.

The third part of the thesis focuses on mixed precision variants of Krylov subspace methods and the mixed precision construction and application of preconditioners within Krylov subspace methods. It includes the articles:

[C6] E. Carson and N. Khan, *Mixed precision iterative refinement with sparse approximate inverse preconditioning*, SIAM Journal on Scientific Computing, 48 (2023), pp. C131–C153.
DOI: 10.1137/22M148770

[C7] E. Carson and I. Daužitkaitė, *Single-pass Nyström approximation in mixed precision*, arXiv preprint arXiv:2205.13355, (2023).
DOI: 10.48550/arXiv.2205.13355

[C8] E. Carson and I. Daužitkaitė, *The stability of split-preconditioned FGMRES in four precisions*, arXiv preprint arXiv:2303.11901, (2023).
DOI: 10.48550/arXiv.2303.11901

[C9] E. Carson, T. Gergelits, and I. Yamazaki, *Mixed precision s-step Lanczos and conjugate gradient algorithms*, Numerical Linear Algebra with Applications, 29 (2022), e2425.
DOI: 10.1002/nla.2425

[C10] I. Yamazaki, E. Carson, and B. Kelley, *Mixed precision s-step conjugate gradient with residual replacement on GPUs*, In Proceedings of the 36th IEEE International Parallel and Distributed Processing Symposium (IPDPS) (2022), pp. 886–896.
DOI: 10.1109/IPDPS53621.2022.00091

In [C6], we develop a finite precision analysis of the construction of sparse approximate inverse (SPAI) preconditioners and show how the unit roundoff should be chosen based on the tolerance parameter $\varepsilon$, which controls the quality of the approximation in terms of sparsity. The conclusions are largely intuitive: the larger the threshold $\varepsilon$, the larger the unit roundoff can be without preconditioner degradation. These results are also used to develop and analyze a variant of mixed precision GMRES-based iterative refinement which uses SPAI preconditioning, called SPAI-GMRES-IR.

Randomized preconditioners are another class of approximate preconditioners which are attractive in practical scenarios. In [C7], we develop a full finite precision analysis of a two-precision single-pass Nyström method, where the potentially lower precision is used for the costly matrix product. We develop a heuristic to estimate how to set this lower precision and demonstrate the use of our mixed precision variant in the preconditioned conjugate gradient method.

The work [C8] analyzes the forward and backward errors in a finite precision split preconditioned Flexible GMRES (FGMRES) method. Four potentially different precisions are used: For the application of the left preconditioner, for the application of the matrix $A$ to a vector, for the application of the right preconditioner, and a general working precision. Our analysis and experimental results show the (perhaps counter-intuitive result) that the precision with which the left preconditioner is applied has a significant affect on the magnitude of the achievable forward and backward errors, whereas this is not the case for the right preconditioner.

The so-called $s$-step variants of Krylov subspace methods can theoretically reduce synchronization cost per iteration by a factor of $O(s)$. However, this comes at the cost of potentially increased convergence delay, which can overshadow any potential per-iteration performance improvement. In the work [C9], we show, theoretically and experimentally, that the selective use of higher precision within $s$-step conjugate gradient and Lanczos algorithms can significantly improve their numerical behavior without a drastic increase in cost.

The work [C10] expands upon the work [C9], presenting a performance study of the mixed precision $s$-step conjugate gradient algorithm studied in [C9] in a multi-GPU environment. The primary conclusion is that the selective use of higher precision as advocated in [C9] has no significant overhead as long as the precisions used are implemented in hardware.

Mixed precision hardware is now an integral part of supercomputing technology, and this trend is expected to continue in the future. In the coming decade following exascale, it is expected that we will see increasing specialization and extreme heterogeneity in a "Cambrian explosion of novel computer architectures" in the words of Hennessey and Patterson [60], which includes novel precision formats. Whereas these trends have thus far resulted in many mixed precision analyses of basic numerical linear algebra computations, the combination of mixed precision computation with algorithmic approximation techniques (including low-rank approximation, sparsification, etc.) remains a largely unexplored area. This thesis provides a preliminary push toward the rigorous mathematical analysis of the combination of different forms of inexactness in matrix computations.

# Chapter 1

# Mixed precision matrix computations

In developing numerical algorithms, it is critical to understand how an algorithm behaves and how the solution is ultimately affected due to the presence of errors. In some cases, errors are due to intentional approximations, made in order to render the problem computationally feasible on the available computing hardware. Some errors arise at the point of data collection, for example, measurement errors or human input errors. Finally, rounding errors due to the use of finite precision computation are an inherent part of computing. Because we have a finite number of bits with which to store data and perform computations, rounding errors are potentially made every time we store a real number in memory and every time we execute a floating point operation.

As mentioned in the preface, a key aspect of modern hardware, in particular GPUs, is the inclusion of low- and mixed-precision capabilities. Originally motivated by low-precision neural network training, these devices have also seen application in broader areas of matrix computations; see, e.g., [57, 59].

The coexistence of many different precision formats within one computing ecosystem offers significant opportunities for developing mixed precision algorithms, where higher precision is used in some parts of the computation and lower precision in others. The idea is to use lower precisions in the most expensive parts of an algorithm in order to improve performance and selectively use higher precisions in relatively inexpensive parts of the algorithm in order to maintain acceptable numerical behavior.

A central theme of this thesis is the analysis of the *stability* of algorithms when they are executed in mixed precision arithmetic. In this chapter, we give a brief background on notions of stability in numerical algorithms, finite precision arithmetic, and mixed precision matrix computations.

## 1.1 Numerical stability

Suppose we have an input $z$ and the exact execution of an algorithm produces the output $y = f(z)$. Due to errors made during the computation, the answer output by the algorithm will not be $y$ but instead some $\widehat{y} = y + \Delta y$. The difference $\Delta y$ between $y$ and $\widehat{y}$ is called the *forward error*. In other words, the forward error indicates how close the computed output is to the true, desired output.

A powerful technique in numerical analysis, pioneered by James Wilkinson and others, is to analyze computations in terms of what is called the *backward error*. The backward error is the quantity $\Delta z$ such that if one were to execute the algorithm exactly (i.e., without errors) on the perturbed input $z + \Delta z$, one would obtain exactly the output $\widehat{y}$. In other words, we have solved the perturbed problem $z + \Delta z$ exactly. Given a bound on the backward error, we can easily obtain a bound on the forward error as well using standard perturbation theory.

The *condition number* of a problem is defined as the ratio of the relative change in the output to the relative change in the input, indicating the sensitivity of a problem to changes in the input. The condition number is useful because it governs the relationship between the forward and backward errors. As a rule of thumb, the forward error is bounded by the product of the condition number and the backward error. This means that even if the backward error (i.e., perturbation to the input) is very small, we might still have a very large forward error and thus a very inaccurate solution if the problem itself is ill conditioned, *even if the perturbed problem is solved exactly.* In contrast, for a well-conditioned problem, a small backward error guarantees that we will also have a small forward error as long as the problem is solved via a backward stable algorithm (see below). We stress that the condition number is a property of the *problem* rather than the *algorithm* used to solve the problem. For example, for linear systems $Ax = b$, the condition number is defined as $\kappa(A) = \|A^{-1}\|\|A\|$ for a given norm. Note that in this thesis, a subscript on $\kappa(A)$ is used to indicate the particular norm when relevant.

Bounding the backward and forward errors for any potential input allows us to define whether an algorithm is (conditionally) forward stable, backward stable, or both. We say that an algorithm for computing $f(z)$ is *backward stable* if for all possible inputs $z$, the algorithm returns $f(z + \Delta z)$ where $\Delta z$ is small (usually relative to the machine precision). In this case, we can say that the algorithm produces the right answer to a slightly wrong question. In contrast, we can view a small forward error as obtaining a slightly wrong answer to the right question. We call an algorithm *forward stable* if it produces an answer with a forward error of similar magnitude to that produced by some backward stable algorithm. Note that this implies that a backward stable algorithm is also forward stable, although the converse does not hold. Notions of forward and backward stability are often combined in what is called *mixed stability.* We can think of an algorithm that is stable in a mixed sense as one that gives a slightly wrong answer to a slightly wrong question.

## 1.2 Floating point arithmetic

As mentioned, computers have only a finite number of bits with which to store real numbers. The standard is to store real numbers as approximations called floating point numbers. The idea of floating point numbers is that they provide a tradeoff between the range of values that can be stored and the precision with which they are stored. The current IEEE 754 standard defines a number of interchange formats with varying numbers of total bits, distributed between a sign bit, a number of exponent bits (which determines the numerical range), and a number of significand bits (which determines the unit roundoff). Base 2 formats are summarized in Table 1.1, which gives the total number of bits, the number of exponent bits, the number of significand bits, the resulting normalized range, which gives the thresholds for underflow/overflow, and the

Table 1.1: IEEE 754 floating point format parameters

| Format | Size | Exponent | Significand | Range | Unit Roundoff ($u$) |
|---|---|---|---|---|---|
| quadruple (fp128) | 128 bits | 15 | 113 | $10^{\pm 4932}$ | $2^{-113} \approx 9.6 \times 10^{-35}$ |
| double (fp64) | 64 bits | 11 | 52 | $10^{\pm 308}$ | $2^{-53} \approx 1.1 \times 10^{-16}$ |
| single (fp32) | 32 bits | 8 | 23 | $10^{\pm 38}$ | $2^{-24} \approx 6.0 \times 10^{-8}$ |
| half (fp16) | 16 bits | 5 | 10 | $10^{\pm 5}$ | $2^{-11} \approx 4.9 \times 10^{-4}$ |

unit roundoff $u$ (also called machine epsilon).

Within the works in this thesis, we will generally use the notation $\text{fl}(\cdot)$ to denote a computation performed in floating point arithmetic. We use a standard model of floating point arithmetic: for scalars $a, b$, we have

$$\text{fl}(a \text{ op } b) = (a \text{ op } b)(1 + \delta), \quad |\delta| \leq u,$$

where op $= +, -, \times$, or $/$. Note that this model assumes that no overflow or underflow occurs, which may no longer be a reasonable assumption in case very low precision is used. Using this simple model, we can bound roundoff errors in computations with matrices and vectors, which consist of a sequence of scalar operations. We direct the unfamiliar reader to, e.g., [63, Chapters 1-3] for further background on finite precision computation and floating point arithmetic.

Historically, supercomputer hardware has typically been limited to at most single and double precision capabilities. As mentioned, modern computer hardware includes capabilities for half and even quarter (8-bit) precision storage and computation as well. We note that there is also recent work in developing new, non-IEEE compliant number formats. One popular example is the bfloat16 number format [14], used in many emerging processors designed for artificial intelligence applications, including Google's tensor processing units and Intel's Nervana Neural Network Processor.

## 1.3 History of mixed precision iterative refinement

The classical example of an algorithm in numerical linear algebra which has historically used mixed precision is iterative refinement. We discuss the basics of iterative refinement for linear systems $Ax = b$ in Chapter 2.2.1 and for least squares problems in Chapter 3.1.1. In short, in the linear system case, given a starting approximate solution $x_0$, step $k = 1, 2, \ldots$ of iterative refinement consists of computing the residual $b - Ax_{k-1}$, solving $Ae_k = r_{k-1}$ for the error $e_k$, and updating the approximate solution via $x_k = x_{k-1} + e_k$. The particular way in which these computations are performed and the precisions used in various parts has given rise to many different variants of iterative refinement, all of which were motivated by the available hardware of the time. Because mixed precision iterative refinement is featured in many of the included works in this thesis, we give a brief overview of historical references here. We direct the reader to [131, Chapter 3] for a longer and more thorough exposition.

In what we call "traditional" iterative refinement, an LU factorization is used to solve for the error term in each iteration and double the working precision is used to compute the residuals (i.e., if the working precision has unit roundoff $u$, we use

a precision with unit roundoff $u^2$ to compute the residuals). This results in relative forward and backward errors on the order of the working precision as long as $\kappa_\infty(A) \leq u^{-1}$. This style of iterative refinement was popular up to the late 1960s, largely due to the available hardware at the time, on which the accumulation of inner products in double the working precision was essentially free. Traditional iterative refinement was used quite early on by Wilkinson in his experiments with Turing on the Pilot Automatic Computing Engine in the 1940s [132]. Error analyses of traditional iterative refinement were presented by Wilkinson for fixed point arithmetic [133] and Moler for floating point arithmetic [89]. Björck extended these ideas, providing an analysis of traditional iterative refinement for least squares problems using what he called an "augmented system" approach [15].

Again in the 1970s and 1980s, hardware developments motivated the study of new variants of iterative refinement. Due to the disappearance of the ability to cheaply accumulate inner products extra precisely from available hardware, work began on the study of fixed precision iterative refinement, in which a uniform precision is used throughout the process. Although this type of iterative refinement was typically seen as not very effective, Jankowski and Woźniakowski proved that fixed precision iterative refinement can provide normwise backward stability for a general linear solver under certain constraints [72]. This was thus the first work that allowed a general linear solver to be used in solving for the error correction. Skeel [109] also provided an analysis of fixed precision iterative refinement for the case of LU factorization, which Higham later extended to a general solver [62]. Shortly, when the refinement is performed entirely in working precision $u$ and an LU factorization is used, this guarantees a relative backward error on the order $u$ and a relative forward error on the order $u \cdot \text{cond}(A, x)$, where $\text{cond}(A, x) = \||A||A^{-1}||x|\|_\infty / \|x\|_\infty$, as long as $\kappa_\infty(A) \leq u^{-1}$. Fixed precision iterative refinement is still used frequently today, with implementations in popular solver packages for both dense and sparse problems including LAPACK, MUMPS, PaStiX, and SuperLU.

In the 2000s, there were two main hardware trends which again motivated the development of different iterative refinement variants. First, communication (data movement) was growing increasingly expensive relative to floating point operations. Second, SIMD (single instruction, multiple data) instructions became available in CPUs. The latter meant that single precision floating point operations could be accomplished at twice the rate of double precision. The former meant that there was a significant performance benefit to moving fewer bits between levels of the memory hierarchy. This motivated researchers to develop what we call "low-precision factorization" variants of iterative refinement, in which the LU factorization (the most expensive part of iterative refinement) was performed in a lower (i.e., single) precision, and double working precision was used in other parts in order to recover accuracy. Here, the resulting guarantees on relative forward and backward error are the same as in fixed precision iterative refinement, but now the constraint for convergence becomes $\kappa_\infty(A) \leq u^{-1/2}$. This approach was used in the work of Langou et al. [78], Buttari et al. [20], Hogg and Scott [70], Arioli and Duff [7], and many others around this time.

Today, hardware is again changing. Motivated by the potential for low precision training and inference in neural network applications, emerging hardware features multiple different precisions, from quarter precision to double precision. This has motivated the work on the development and analysis of the new mixed precision iterative refine-

ment variants included in this thesis; see [C1], [C2], [C4], [C3], and [C6]. We note that, in particular, the works [C1] and [C2] have inspired a flurry of recent work in this area. See, for example, the related work of Amestoy et al. [4], Vieublé [131], Haidar et al. [57], Higham and Pranesh [65], Amestoy et al. [5], and Abdelfattah et al. [2].

## 1.4 Summary of other related work in mixed precision numerical linear algebra

Efforts in mixed precision numerical linear algebra and matrix computations have a long history, and of course go beyond iterative refinement for linear systems and least squares problems. We do not attempt to give a complete survey of references here, but aim to merely point out key works. For more complete surveys, we refer the reader to [1] and [64], where we note that the present author is a co-author of the former.

Newton's method is a method for iteratively finding a zero $x$ of a continuously differentiable function $f(x)$. Iterative refinement can be seen as a special case of Newton's method in which $f(x) = b - Ax$. Thus it is natural that mixed precision approaches are also useful in Newton's method. Motivated by iterative refinement for the generalized eigenvalue problem, Tisseur [120] provided an analysis of the limiting error and residual in a mixed precision Newton iteration in which extra precision is used in computing the residuals and the linear system in each iteration may be solved in a lower precision (or using a less stable solver). The resulting convergence behavior when the Jacobian is stored and factored in lower precisions (including half precision) has been recently investigated by Kelley [74].

We can view the computation of an eigenvalue $\lambda$ and eigenvector $x$ as the solution of the nonlinear system of equations $(A - \lambda I)x = 0$. Thus Newton's method can also be used for the refinement of eigenvalues and eigenvectors. This connection was exploited in the 1980s by Dongarra [34] and Dongarra, Moler, and Wilkinson [35] to develop a mixed precision algorithm for the iterative refinement of eigenvalues and eigenvectors. The approach of [34] has recently been revisited by Tsai et al. [123], who replaced the Givens rotations with a Sherman-Morrison update to make the algorithm more parallelizable. Petschow et al. [101] developed a mixed precision variant of the multiple relatively robust representations (MRRR) algorithm for symmetric tridiagonal eigenproblems. By using higher precision in select computations, the method becomes as accurate as competing approaches without sacrificing its performance advantages. Ogita and Aishima [91, 92] developed a mixed precision iterative refinement approach for symmetric eigenvalue problems, which they later extended to singular value problems [93]. Kressner et al. [75] proposed a mixed precision variant of the locally optimal block preconditioned conjugate gradient method (LOBPCG) for finding a small number of eigenvalue/eigenvector pairs. They provide a rounding error and convergence analysis for a simplied variant of LOBPCG, called PINVIT. Also exploiting connections with Newton iteration, Bujanović et al. [18] recently developed a mixed precision iterative refinement approach for updating a Schur decomposition, which has applications in eigenvalue problems as well as matrix functions and matrix equations.

Mixed precision has also been used successfully in orthogonalization routines. In [135], Yamazaki et al. provide an analysis of a mixed precision Cholesky QR algorithm, in which intermediate quantities are selectively computed in double the working

precision. They prove that this reduces the loss of orthogonality from quadratic dependence on condition number to only linear dependence. This mixed precision Cholesky QR was subsequently used by Yamazaki et al. as the panel factorization routine in a block modified Gram-Schmidt algorithm [137]. Yamazaki et al. [136] also developed an adaptive mixed precision singular value QR (SVQR) algorithm, which adaptively determines whether low precision can be used for the triangular solves (the most computationally expensive part) without affecting the loss of orthogonality. Yang et al. [138] have recently developed a rounding error analysis of mixed precision Householder QR and its block variant in which inner products are computed in high precision and subsequently rounded to lower precision, which is applicable to block fused multiply add operations (FMAs).

Mixed precision approaches to the multigrid method usually involve an outer iterative refinement scheme with multigrid used as the inner solver for the correction vector. In 2014, Sumiyoshi et al. [114] provided performance results for such an approach, in which the correction equation is solved via algebraic multigrid (AMG) in single precision and double precision is used elsewhere. Similar approaches have also been used by Göddeke et al. [46], Göddeke and Strzodka [45], and Kronbichler and Ljungkvist [76]. In [86] and [117], McCormick, Benzaken, and Tamstorf provide a theoretical rounding error analysis of mixed precision multigrid solvers using the framework of iterative refinement framework. Their work shows that the coarser the grid, the lower the precision that can be used.

Mixed precision has been used in various ways within Krylov subspace methods; for details on various Krylov subspace methods, see Chapter 2.3. One possibility is to use mixed precision within a restarted generalized minimal residual method (GMRES), which is equivalent to iterative refinement without a preconditioner; see the 1992 work of Turner and Walker [124] and the more recent approach of Lindquist et al. [81]. Another general possibility is to use a mixed or low precision preconditioner within a Krylov subspace method. There are many variations on this which have appeared in the literature, including the work of Arioli and Duff [7], who used a flexible GMRES method preconditioned by a single precision LU factorization, Giraud et al. [42], who used a single precision domain decomposition preconditioner within a double precision conjugate gradient (CG) method, Emans and van der Meer [38], who used a single precision AMG method within CG, Anzt and Fleger et al. [6, 39], who used a mixed precision Jacobi preconditioner within CG, and Göbel et al. [44], who used a mixed precision sparse approximate inverse preconditioner for the stabilized biconjugate gradient method (BiCGStab). The included works [C6], [C5], and [C8] in this thesis also fit into this category. Another possible technique is to selectively use extra precision in order to stabilize communication-avoiding variants of Krylov subspace methods. This idea was used by Yamazaki et al. [134, 135] and is also used in the included works [C9] and [C10]. Finally, we mention the related work on "relaxed" and "inexact" Krylov subspace methods, which show that in some cases, the accuracy of the matrix vector products or inner products can be relaxed at a rate inversely proportional to the convergence of the residual norm without affecting attainable accuracy; see, e.g., [41], [108], [125], and [50].

## 1.5 Outlook

In the current exascale era, hardware trends indicate that the prevalence of machines with mixed precision capabilities will only increase. We thus expect the emphasis on the study of finite precision matrix computations, and especially mixed precision matrix computations, to continue in the coming years.

In addition to increasingly low precision IEEE formats, with increasing hardware heterogeneity, we expect new non-IEEE arithmetics to gain traction as well [82]. Indeed, many domain-specific alternatives have emerged, such as bfloat16 format [14] and NVIDIA's 19-bit TensorFloat format [118], although these still use a fixed number of bits for the exponent and significand. Another alternative is "posit" arithmetic, in which the number of exponent and fraction bits can vary [55]. This can have advantages from both a numerical and a performance standpoint, but analysis is difficult, as the relative error can't be bounded even for simple computations [19].

We believe that developing algorithms for mixed precision matrix computations and analyzing the numerical stability under these new novel arithmetic formats, especially when combined with the analysis of other algorithmic sources of inexactness, will be a fruitful area of research, leading to new high-performance implementations.

# Chapter 2

# Introduction to iterative methods for linear systems

A main focus of this thesis is on the solution of nonsingular $n \times n$ linear systems $Ax = b$. In the realm of large-scale computational and data science applications, many problems give rise to a matrix $A$ that is naturally sparse, meaning that the underlying computational graph has only nearest neighbor connections. Iterative solvers are often the methods of choice for solving large, sparse linear systems (as well as least squares problems and eigenvalue problems). The advantages over direct methods are that iterative methods often only require sparse matrix-vector products (and do not even require an explicitly-stored matrix), and can be stopped once the required accuracy is attained. Among the considered iterative methods, we distinguish two subclasses: stationary iterative methods and Krylov subspace methods.

## 2.1 Preconditioning

Iterative methods, in particular Krylov subspace methods, almost always use a preconditioner in practical applications. In the case of linear systems, preconditioning means that we transform the system $Ax = b$ into the equivalent system $M_1^{-1}AM_2^{-1}\tilde{x} = M_1^{-1}b$, with $M_2x = \tilde{x}$, where $M_1^{-1}AM_2^{-1}$ is not formed explicitly. The nonsingular matrices $M_1$ and $M_2$ are referred to as preconditioners.

There are two goals in designing a preconditioner; first, convergence should be faster for the preconditioned system, and second, the preconditioner should be inexpensive to compute and apply. The design of a good preconditioner is almost always problem-specific and often involves insights from the domain of the particular application. This alone is an area of deep research.

Depending on the context, it may make sense to precondition only on the left (i.e., $M_2 = I$), only on the right (i.e., $M_1 = I$), or to use a split preconditioner with both $M_1$ and $M_2$ differing from the identity. Note that when left preconditioning is used, the solution $\tilde{x}$ to the preconditioned system is the same as the solution $x$ to the original system, but the residuals differ. In contrast, for the case of right preconditioning, the residual is the same for both the preconditioned and the original system, but the solutions differ. This will naturally have consequences on the considered forward and backward errors, as well as the stopping criteria used.

### 2.1.1 Algebraic preconditioners

Algebraic preconditioning refers to the construction and use of preconditioners based solely on the numerical entries in a matrix $A$, rather than the underlying problem itself (for example, the solution of a partial differential equation). Preconditioning approaches that fall into this class include, for example, Jacobi, Gauss-Seidel, (incomplete) LU factorization, sparse approximate inverse (SPAI) preconditioners, and algebraic multigrid. While such preconditioners can sometimes perform poorly compared to those that take into account the underlying problem to be solved, such as operator preconditioning approaches (see, e.g., [83]), they are more versatile and applicable to a diverse set of problems. We focus on this class of preconditioners in this work, since our perspective is that of a developer of general algorithms and software for solving problems given an input matrix.

One example of an algebraic preconditioning technique that we use within this thesis is SPAI preconditioning. Here, we seek to construct a sparse matrix $M$ such that $M \approx A^{-1}$, where $A, M \in \mathbb{R}^{n\times n}$. Such preconditioners are attractive for use within Krylov subspace methods since their application involves only a sparse matrix-vector product. There are a wide variety of strategies for constructing such an $M$; see, for example, Chapter 11 of the recent book [107].

In particular, in [C6], we consider the popular variant of SPAI construction due to Grote and Huckle [54]. Here $M$ is constructed based on Frobenius norm minimization, where $M$ is the solution to

$$\min_{\mathcal{J}\in\mathcal{S}} \|I - AM\|_F,$$

where $\mathcal{J} \in \mathbb{B}^{n\times n}$ is a prescribed binary sparsity pattern chosen from the set of all binary sparsity patterns $\mathcal{S} \in \mathbb{B}^{n\times n}$. A key property of the Frobenius norm minimization approach is that we can write

$$\min_{\mathcal{J}\in\mathcal{S}} \|I - AM\|_F^2 = \sum_{k=1}^{n} \min_{\mathcal{J}_k\in\mathcal{S}_k} \|e_k - Am_k\|_2^2,$$

where $e_k$ is the $k$th column of the identity and $\mathcal{J}_k$, $\mathcal{S}_k$, and $m_k$ are the $k$th columns of $\mathcal{J}$, $\mathcal{S}$, and $M$, respectively. In other words, the computation can be decoupled into the solution of $n$ independent linear least squares problems, which can theoretically be solved in a highly parallel manner.

Whereas early works used a fixed sparsity pattern $\mathcal{J}$, a key innovation of Grote and Huckle [54] was to use an adaptive iterative approach that, in each iteration, determines some number of "most important" nonzero indices to add to the sparsity pattern $\mathcal{J}$. Nonzeros are added to the pattern until the constraint $\|e_k - Am_k\|_2 \leq \varepsilon$ is satified for each column, where $\varepsilon$ is a user-specified tolerance parameter. For further details, see [54].

Many other works included in this thesis touch on aspects involving algebraic preconditioners, including the use of LU-based preconditioners ([C1], [C2], [C3], and [C8]), and QR-based preconditioners ([C4] and [C5]).

### 2.1.2 Randomized preconditioners

Randomized preconditioners, which generally fall into the class of algebraic preconditioners, are a relatively new development; while randomized algorithms have historically

been viewed as somewhat of a last resort in scientific computing, the last decade has seen exciting work on the use of randomization in matrix computations and numerical linear algebra. The potential benefits of randomized algorithms include a faster runtime, potentially better stability, and a greater level of interpretability [36]. Randomized matrix computations have since gained momentum and a randomized BLAS (Basic Linear Algebra Subroutines) library is currently in early stages of development.

A random "sketch" of a matrix (a matrix of reduced size, but which retains some numerical properties of the original matrix) can be obtained via row/column sampling or projection methods. In this regime, there are three major paradigms for solving matrix problems (see, e.g., [85, Section 10]):

1. sketch-and-solve, in which the problem is mapped to a smaller space, and the solution for the reduced problem is used as an approximate solution to the original problem;
2. iterative sketching, which applies the sketch-and-solve paradigm iteratively to reduce the error; and
3. sketch-and-precondition, which uses random embeddings to create a sketch of the input matrix and then uses this as a preconditioner within a iterative method.

There is deep theory underlying randomized numerical linear algebra and there has been significant work in analyzing how much approximation error is introduced through randomized sketching and how this affects solutions; see, e.g, the surveys [58, 85] and references therein. However, nearly ubiquitously, inexactness due to finite precision computation is not traditionally taken into account in such analyses.

Perhaps the most typical example of randomized preconditioners are those for overdetermined least squares problems developed by Rokhlin and Tygert [103] and further developed by the authors in [10]. The main idea is to use a randomized approach for computing the $R$ factor from the QR factorization of $A$, and to use this as a right preconditioner in the least squares QR (LSQR) algorithm. In short, we first create a sketch of the $m \times n$ matrix $A$ by premultiplying by an $s \times n$ random matrix $G$. The sketched version of $A$ will be much smaller than the original matrix. We then compute the QR factorization of this smaller matrix, and the obtained $n \times n$ $R$ factor is used as an approximation of the true $R$ factor of $A$. It is provable that the preconditioned matrix $AR^{-1}$ has a small condition number [103, 10]. Note that the authors in [87] have recently provided an analysis of this process accounting for finite precision error.

Another example of the use of randomized preconditioners is in the development of limited memory preconditioners for problems arising in data assimilation. In general, we can consider solving the shifted system of linear equations

$$(A + \mu I)x = b, \tag{2.1}$$

where $A \in \mathbb{R}^{n \times n}$ is symmetric positive semidefinite, $\mu \geq 0$ is chosen so that $A + \mu I$ is positive definite, and $x, b \in \mathbb{R}^n$. One can solve this system using the preconditioned conjugate gradient (CG) method with a limited memory preconditioner of the form

$$P = I - UU^T + \frac{1}{\alpha + \mu} U(\Theta + \mu I)U^T, \tag{2.2}$$

where $U\Theta U^T \approx A$ is a low-rank approximation of the eigendecomposition of $A$ and $\alpha$ is a shift parameter.

The quantities $U$ and $\Theta$ can be obtained via a randomized Nyström approximation. The randomized Nyström approximation $A_N$ of a symmetric positive semidefinite matrix $A \in \mathbb{R}^{n\times n}$ is of the form

$$A_N = (AX)(X^TAX)^\dagger(AX)^T, \tag{2.3}$$

where $X \in \mathbb{R}^{n\times k}$ is a random matrix and $\dagger$ denotes the Moore-Penrose pseudoinverse; see, e.g., [43]. In the work [C7], we provide a full finite precision analysis of a mixed precision single-pass variant of the Nyström method, first described in [122], which shows that in many practical cases the most expensive part of the computation can be performed in lower precision without degradation of preconditioner quality. See, e.g., [3, 32, 40] for further examples of the use of the Nyström method in developing preconditioners.

## 2.2 Stationary iterative methods

Stationary iterative methods are the simplest approach for iteratively solving linear systems. In such methods, the solution to a linear system is expressed as finding the stationary point of a fixed-point iteration. Methods in this class include Richardson iteration, Jacobi, Gauss-Seidel, and the symmetric successive overrelaxation method. Convergence of these methods is usually very slow; in practice, stationary methods are often used as preconditioners for Krylov subspace methods and as smoothers in multilevel methods rather than as standalone solvers.

A general technique for developing such methods is based on a splitting of the coefficient matrix $A$. We can write such a splitting $A = M - N$, where $M$ should be close to the matrix $A$ and solving linear systems with $M$ should be efficient. Then the linear system $Ax = b$ can be rewritten $Mx = Nx + b$.

Given an initial approximate solution $x_0$, we can then use this splitting to define an iterative method given by the update formula

$$Mx_k = Nx_{k-1} + b, \tag{2.4}$$

which, using $N = M - A$, can be written

$$x_k = x_{k-1} + M^{-1}(b - Ax_{k-1})$$

for $k = 1, 2, \ldots$. Let $e_k = x - x_k$ denote the error in the $k$th iteration. Via simple algebraic manipulation, we can obtain the formula

$$e_k = (M^{-1}N)^k e_0,$$

and taking norms, we have

$$\|e_k\| \leq \|M^{-1}N\|^k \|e_0\|.$$

Thus whether the iteration converges for any starting vector and the resulting rate of convergence depends on the spectral radius of $M^{-1}N$. In particular, if the spectral radius (the largest eigenvalue in absolute value) is less than one, the method is convergent.

The particular choices of $M$ and $N$ define various iterative methods. In the Jacobi method, for example, we take $M = D$ and $N = -(L+U)$, where $D$ is the diagonal of the

matrix, and $L$ and $U$ are the strictly lower and upper triangular entries, respectively. The Gauss-Seidel method uses $M = D + L$ and $N = -U$. For successive overrelaxation (SOR), we take $M = (1/\omega)D + L$ and $N = ((1/\omega) - 1)D - U$; $\omega > 1$ is the relaxation factor and should be chosen to minimize the spectral radius of $M^{-1}N$. For more details on these and related methods, see, e.g., Chapter 4 of [105].

### 2.2.1 Preconditioned Richardson iteration/iterative refinement

A particular stationary method which we will use extensively in this work is preconditioned Richardson iteration, which is more frequently in the literature called *iterative refinement*. For unpreconditioned Richardson iteration, we take $M = I$ and $N = I - A$. We then obtain the iteration

$$x_k = x_{k-1} + r_{k-1},$$

where $r_{k-1} = b - Ax_{k-1}$ is called the residual. We know that this is convergent if the spectral radius of $I - A$ is less than one.

This condition can be improved by the use of preconditioning. Instead of solving $Ax = b$, we will solve $BAx = Bb$ for some preconditioner $B$. Note that application of this preconditioner does not change the solution $x$. The hope is that the spectral radius of $I - BA$ is smaller than that of $I - A$. For the preconditioned Richardson iteration, we then have $M = I$ and $N = I - BA$, which, substituting into (2.4) gives the iteration

$$x_k = (I - BA)x_{k-1} + Bb,$$

which we rewrite as

$$x_k = x_{k-1} + Br_{k-1}.$$

Taking $B = A^{-1}$ gives the scheme referred to as iterative refinement. Given an initial approximate solution $x_0$, each iteration $k = 1, 2, \ldots$ of iterative refinement takes the following form:

1. Compute the residual $r_{k-1} = b - Ax_{k-1}$.
2. Solve the linear system $Ae_k = r_{k-1}$.
3. Update the approximate solution $x_k = x_{k-1} + e_k$.

Note that if the linear system in step 2 were to be solved exactly, we would converge within a single iteration. Of course in practice, this solution is not exact due to finite precision error. Traditionally, the solve in step 2 is performed by using an LU factorization of the matrix $A$ (which may already be computed during the computation of the initial approximate solution $x_0$). Alternatively, the linear system in step 2 can be solved using a preconditioned Krylov subspace method, where the LU factors are used as preconditioners; this is the topic of the included work [C1].

Iterative refinement is a classical example of an algorithm in numerical linear algebra that can benefit from mixed precision computation. The ideas of mixed precision iterative refinement go back to Wilkinson in 1948 [132]. A key point is that it may be beneficial to use a precision higher than the working precision in the residual computation (step 1). For convergent methods, this results in removing the effects of the

conditioning of the matrix $A$ on the resulting attainable error $x_k - x$. Later, other works showed that it is often possible to use a precision lower than the working precision in computing the LU factorization of $A$ used in step 2 in each iteration; see the historical description in Chapter 1.3. The combination of these two ideas into a three-precision approach is the topic of the work [C2] included in this thesis.

Note also that iterative refinement requires some choice of stopping criteria. Potential stopping criteria are discussed in [33] and are also developed in [C3] for the case of three-precision iterative refinement.

## 2.3 Krylov subspace methods

Krylov subspace methods are a popular class of general iterative solvers for linear systems, least squares problems, and eigenvalue/singular value problems. Methods in this category include the Lanczos method, the CG method, the Arnoldi method, the generalized minimum residual method (GMRES), and LSQR, among others. The benefit of Krylov subspace methods versus stationary iterative methods is their potential for superlinear convergence. This nonlinear convergence behavior is what gives Krylov subspace methods power, but also what makes them inherently difficult to analyze. While much progress has been made towards their analysis, there are still a number of open problems; see, for example, [79].

In general, we can think of a Krylov subspace method as a projection process onto an expanding Krylov subspace, denoted in terms of a dimension $i$, matrix $A$, and starting vector $r_0$ as

$$\mathcal{K}_i(A, r_0) = \text{span}\{r_0, Ar_0, A^2r_0, \ldots, A^{i-1}r_0\}.$$

In the case of solving linear systems, an approximate solution $x_i$ is chosen from the subspace $x_0 + \mathcal{K}_i$ according to the Petrov-Galerkin condition $b - Ax_i \perp \mathcal{L}_i$, where $x_0$ is an initial approximate solution and the choice of $\mathcal{L}_i$ distinguishes the various types of Krylov subspace methods.

All works included in this thesis involve Krylov subspace methods in some way, namely, in the development of mixed precision variants of $s$-step Lanczos and CG algorithms ([C9] and [C10]), in the analysis and application of mixed precision left-preconditioned GMRES ([C1], [C2], [C3], [C4], and [C6]), in the development and use of mixed precision preconditioners for CG ([C5] and [C7]), and in the analysis of a mixed precision split-preconditioned Flexible GMRES (FGMRES) method ([C8]). We therefore give brief overviews of Lanczos, CG, GMRES, and FGMRES methods and their preconditioned variants, with a focus on their behavior in finite precision arithmetic. We then also present a brief derivation and details on the $s$-step variants of Lanczos and CG.

### 2.3.1 Lanczos and CG

The Lanczos method, introduced by Cornelius Lanczos [77], is a popular approach for finding a few eigenvalues/eigenvectors of a symmetric sparse matrix. Given a symmetric $n \times n$ matrix and a starting vector $v_1$ with $\|v_1\|_2 = 1$, the algorithm iteratively constructs an orthonormal basis for the Krylov subspace $\mathcal{K}_i(A, v_1)$ via Gram-Schmidt

orthogonalization. We show a particular variant of the Lanczos algorithm in Algorithm 1, which is based on the use of 2-term recurrences. In matrix notation, in exact arithmetic the Lanczos iteration can be expressed as

$$AV_i = V_i T_i + \eta_{i+1} v_{i+1} e_i^T, \tag{2.5}$$

where $V_i = [v_1, \ldots, v_i]$ is an orthonormal basis for the Krylov subspace $\mathcal{K}_i(A, v_1)$ and $T_i$ denotes the tridiagonal matrix

$$T_i = \begin{bmatrix} \gamma_1 & \eta_2 & & & \\ \eta_2 & \gamma_2 & \eta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \eta_{i-1} & \gamma_{i-1} & \eta_i \\ & & & \eta_i & \gamma_i \end{bmatrix},$$

the entries of which are inner products computed during the Lanczos procedure.

**Algorithm 1** Lanczos (2-term recurrence variant)

**Require:** Real symmetric matrix $A \in \mathbb{R}^{n \times n}$; length $n$ starting vector $v_1$ such that $\|v_1\|_2 = 1$; maximum number of iterations $n_{\max}$.

1: $u_1 = Av_1$
2: **for** $i = 1, 2, \ldots, n_{\max}$ **do**
3: $\quad \gamma_i = v_i^T u_i$
4: $\quad w_i = u_i - \gamma_i v_i$
5: $\quad \eta_{i+1} = \|w_i\|_2$
6: $\quad v_{i+1} = w_i / \eta_{i+1}$
7: $\quad u_{i+1} = Av_{i+1} - \eta_{i+1} v_i$
8: **end for**

In exact arithmetic, there must exist some $k \leq n$ for which $AV_k = V_k T_k$ (since we cannot have a set of more than $n$ orthonormal vectors in $\mathbb{R}^n$), at which point we know that the eigenvalues of $T_k$ must be a subset of those of $A$. We note that in practice, the Lanczos algorithm is often stopped before the point of reaching an invariant subspace.

Because the Lanczos algorithm can be seen as a Rayleigh-Ritz procedure, the eigenvalues $\theta_1^{(i)}, \ldots, \theta_i^{(i)}$ of $T_i$, which give estimates of the eigenvalues of $A$, are referred to as the Ritz values. From the eigenvectors $z_1^{(i)}, \ldots, z_i^{(i)}$ of $T_i$, we can obtain the associated Ritz vectors $x_j^{(i)} = V_i z_k^{(i)}$, which are approximations of the eigenvectors of $A$. We note that there are other possible mathematically equivalent implementations of the Lanczos method than that presented in Algorithm 1, e.g., one could use a 3-term recurrence for updating the vectors $v_{i+1}$. It has been found that the variant presented in Algorithm 1 is preferable numerically; see, e.g., [94].

The CG method, which is closely related to the Lanczos method, was developed independently by Hestenes and Stiefel which resulted in the joint 1952 paper [61]. Let $A$ now be a symmetric positive definite (SPD) matrix, let $b$ be the right-hand side of the linear system $Ax = b$, and let $x_0$ be an initial approximate solution. In each iteration $i = 1, 2, \ldots$ of CG, we update the approximate solution using the formula $x_i = x_0 + V_i y_i$, where $V_i$ is the matrix generated by the Lanczos procedure with starting vector $v_1 =$

$r_0/\|r_0\|_2$, where $r_0 = b - Ax_0$. In iteration $i$ we can write the residual $r_i = b - Ax_i = b - A(x_0 + V_i y_i) = r_0 - AV_i y_i$. If we enforce that the residual $r_i$ is orthogonal to $V_i$ (taking $\mathcal{L}_i = \mathcal{K}_i(A, r_0)$ in the Petrov-Galerkin condition), then by premultiplying by $V_i^T$ and using (2.5), we have $T_i y_i = \|r_0\|_2 e_1$, and so the coordinates $y_i$ can be found by solving this small linear system with $T_i$. Note that the enforcement of the Petrov-Galerkin orthogonality condition also implies that $\|x - x_i\|_A = \min_{z \in x_0 + \mathcal{K}_i(A,r_0)} \|x - z\|_A$, i.e., the approximate solution $x_i$ that is chosen in iteration $i$ is that which minimizes the $A$-norm of the error. This also implies that $r_{n+1} = 0$, so we will solve the linear system exactly in at most $n$ iterations.

As stated, the advantage of Krylov subspace methods over stationary iterative methods is their nonlinear behavior, i.e., CG and other Krylov subspace methods *adapt* to the data and can exhibit superlinear convergence behavior. This thesis is not centered on the convergence behavior of CG in exact arithmetic and thus we do not discuss this in detail here; the interested reader is directed to the work [23]. In short, the convergence behavior of CG depends on the distribution of eigenvalues and the sizes of the components of the initial residual in the eigenbasis of $A$. While convergence bounds based on the condition number are frequently given in the literature, these are linear bounds which cannot capture the nonlinear behavior of CG. They do, however, tell us that when $A$ is well-conditioned, we can expect CG to converge quickly.

In Algorithm 2, we give one possible algorithm which realizes the conjugate gradient method. This variant, presented in the original paper [61], has also been found to be preferable numerically; see, e.g., [102], [56].

There are many interesting deep theoretical connections of the Lanczos and CG methods with, for example, orthogonal polynomials, the Stieljes moment problem, and Gauss quadrature. These connections are not the central topic of this thesis and thus we do not discuss the details here. We refer the interested reader to the works [79, Section 3.5], [84, Section 5.2], and [88]. We note also that there are many potential choices for the stopping criterion used in line 7 of Algorithm 2, and we cannot expect that the residual norm gives an indication of the $A$-norm of the error; see, e.g., [113] for a thorough discussion.

**Algorithm 2** Conjugate Gradient (CG; 2-term recurrence variant)

**Require:** Symmetric positive definite matrix $A \in \mathbb{R}^{n \times n}$; right-hand side $b$; initial approximation $x_0$; given stopping criterion; maximum number of iterations $n_{\max}$.

1: $r_0 = b - Ax_0$
2: $p_0 = r_0$
3: **for** $i = 1, 2, \ldots, n_{\max}$ **do**
4: $\quad \alpha_{i-1} = (r_{i-1}^T r_{i-1})/(p_{i-1}^T A p_{i-1})$
5: $\quad x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$
6: $\quad r_i = r_{i-1} - \alpha_{i-1} A p_{i-1}$
7: $\quad$ Test stopping criterion. If satisfied, then return $x_i$ and stop.
8: $\quad \beta_i = (r_i^T r_i)/(r_{i-1}^T r_{i-1})$
9: $\quad p_i = r_i + \beta_i p_{i-1}$
10: **end for**

**Preconditioning** As mentioned in Section 2.1, preconditioning is almost always in practice used in conjunction with Krylov subspace methods in order to accelerate convergence. In the case of CG, since $A$ is SPD, we assume that the preconditioner $M$ is also SPD. We can thus write $M = LL^T$ where $L$ is the Cholesky factor of $M$. Although left and right preconditioned variants of CG are also possible (see, e.g., [105, Chapter 9.2]), a natural way to precondition CG is to use a split preconditioner so that the preconditioned system becomes $L^{-1}AL^{-T}\tilde{x} = L^{-1}b$, with $x = L^{-T}\tilde{x}$. In this thesis, we use split preconditioned CG in [C5] and [C7].

**Finite precision behavior** In finite precision arithmetic, the mathematical properties of Lanczos and CG no longer hold. This was known even in very early works; see [77] and [61]. The computed basis for the Krylov subspace is no longer perfectly orthonormal, and thus for CG, the $A$-norm of the error is not minimized in each iteration. We are thus no longer guaranteed that we will find the exact solution to $Ax = b$ within $n$ iterations. The effects of finite precision errors manifest in Krylov subspace methods in two main ways: delayed convergence and a loss of attainable accuracy.

The most important treatment of the finite precision behavior of Lanczos was accomplished by Chris Paige. In a series of papers, Paige gave a complete rounding error analysis of the Lanczos algorithm and showed that the loss of orthogonality in Lanczos follows a pattern and actually implies the convergence of Ritz values to eigenvalues of $A$ [94, 95, 96]. A number of other important fundamental results are also included; for example, Paige proved that the computed Ritz values always lie between the extreme eigenvalues of $A$ to within a small multiple of the unit roundoff. The present author has previously extended the work of Paige to the $s$-step Lanczos algorithm [22, 27], and extends the work of Paige to a mixed precision variant of the $s$-step Lanczos algorithm in the included work [C9]; see also Chapter 2.3.3. Greenbaum [51] used the analysis of Paige to subsequently prove an important result. In short, the finite precision Lanczos and CG algorithms can be viewed as the exact algorithms run on a larger matrix $\tilde{A}$ whose eigenvalues lie in tight clusters around those of $A$. It is also known that the sensitivity of CG to convergence delay caused by rounding errors depends on the distribution of eigenvalues, with large outlying eigenvalues causing CG to be more susceptible to these effects; see [73], [112], and [23, Section 2.6].

The mechanism by which accuracy is lost within the CG algorithm is related to the difference between the recursively updated residual $r_i$ and the true residual $b - Ax_i$. Notice that there is no "self-correction" mechanism which would keep these two quantities close, and thus local rounding errors cause them to grow further and further apart as the iterations proceed. It can be shown that the size of the true residual is limited by the quantity $\|b - Ax_i - r_i\|$, which is often called the "residual gap". For CG, this bound can be written as a simple accumulation of local rounding errors; see, e.g., [52], [127], and [110]. The author has previously extended these analyses of the maximum attainable accuracy to the $s$-step CG method [21, 27].

For further details and references related to the finite precision behavior of Lanczos and CG, see [88, Sections 4 & 5].

### 2.3.2 GMRES

GMRES [106] is a Krylov subspace method for solving general linear systems. GMRES iteratively constructs an orthonormal basis for the Krylov subspace $\mathcal{K}_i(A, v_1)$ where $v_1 = r_0/\|r_0\|_2$ via the Arnoldi method and enforces the Petrov-Galerkin condition with $\mathcal{L}_i = A\mathcal{K}_i$ in choosing the next approximate solution. As the name suggests, this results in the selection of an approximate solution in each iteration that minimizes the 2-norm of the residual.

Similar to Lanczos, the Arnoldi process in matrix form can be written

$$AV_i = V_i H_i + h_{i+1,i} v_{i+1} e_i^T = V_{i+1} H_{i+1,i}.$$

The primary difference with Lanczos is that since $A$ is now nonsymmetric, we have an upper Hessenberg matrix $H_i$ instead of the tridiagonal $T_i$. Within the Arnoldi method, one can choose various orthogonalization routines to construct the orthonormal basis. Common choices include classical Gram-Schmidt (CGS), reorthogonalized CGS, modified Gram-Schmidt (MGS), and Householder orthogonalization.

The choice of orthogonalization scheme will determine both performance and backward stability of the resulting GMRES method. While Householder, MGS, and reorthogonalized CGS are suitable choices to ensure the backward stability of GMRES (see [37] and [97]), CGS does not guarantee a sufficient level of orthogonality for the resulting GMRES algorithm to be backward stable. By backward stable we mean that a backward stable solution to $Ax = b$ is produced within $n$ iterations; see Chapter 1.1. Note that this implies that there is no convergence delay in finite precision GMRES as long as sufficient orthogonality of the Krylov basis is maintained.

Because MGS and reorthogonalized CGS are more suitable than Householder in high-performance settings due to communication cost, these algorithms are often used in practice. We note that there is also much recent work in developing stable, low-synchronization orthogonalization routines for use within high-performance variants of GMRES; see, e.g., [12], [115], and [119], the last of which was co-authored by the present author. In the included works [C1] and [C2], we extend the analysis of the backward stability of MGS-GMRES in [97] to the case where we use mixed precision and left preconditioning.

As in CG, in GMRES we search for an approximate solution in the affine subspace $x_0 + \mathcal{K}_i$, which we can write as $x_i = x_0 + V_i y_i$. We can write the residual as

$$\begin{aligned} b - Ax_i &= b - A(x_0 + V_i y_i) \\ &= r_0 - AV_i y_i \\ &= \|r_0\|_2 v_1 - V_{i+1} H_{i+1,i} y_i \\ &= V_{i+1}(\|r_0\|_2 e_1 - H_{i+1,i} y_i). \end{aligned}$$

Since the columns of $V_{i+1}$ are orthonormal, the 2-norm of the residual can be minimized by selecting $y_i$ to be $\arg\min_y \|\|r_0\|_2 e_1 - H_{i+1,i} y\|_2$. In other words, in order to select the update to $x_i$ that minimizes the 2-norm of the residual in each iteration, we must solve an $(i+1) \times i$ least squares problem. This can be accomplished efficiently by updating a QR factorization of the matrix $H_{i+1,i}$ using Givens rotations, from which we can obtain an estimate of the residual 2-norm without explicitly forming the approximate solution. The resulting GMRES algorithm (in which MGS orthogonalization is used) is shown in Algorithm 3.

We note that because, unlike for a symmetric matrix, the Arnoldi relation requires full recurrences (i.e., we must orthogonalize each new vector against all previous vectors), the cost of Arnoldi/GMRES grows with the iteration number $i$. For this reason, a restarted variant of GMRES is often used in practice, although we cannot in general guarantee that restarted GMRES will produce a backward stable solution within $n$ iterations.

In contrast to CG, in which the convergence behavior depends heavily on the eigenvalue distribution, for GMRES, any nonincreasing convergence curve is possible for a matrix (which is nonnormal in general) having any prescribed set of eigenvalues [53]. See [9] for a complete parametrization of all pairs $\{A, b\}$ for which GMRES generates a prescribed convergence curve. In the case of normal matrices, the location of its eigenvalues does give information about the worst-case GMRES behavior. For example, complete stagnation until the final iteration can be observed for cases where the eigenvalues form a cluster about the origin. For details, see, e.g., [80].

**Algorithm 3** Generalized Minimal Residual Method (GMRES)

**Require:** Nonsingular matrix $A \in \mathbb{R}^{N\times N}$, right-hand side $b$, initial approximation $x_0$; convergence tolerance $\tau$; maximum number of iterations $n_{\max}$.

$r_0 = b - Ax_0$, $\beta = \|r_0\|_2$, $v_1 = r_0/\beta$
**for** $i = 1, 2, \ldots, n_{\max}$ **do**
  $w_i = Av_i$
  **for** $j = 1, \ldots, i$ **do**
    $h_{ji} = v_j^T w_i$
    $w_i = w_i - h_{ji} v_j$
  **end for**
  $h_{i+1,i} = \|w_i\|_2$
  Let $V_i = [v_1, \ldots, v_i]$ and $H_{i+1,i} = \{h_{k\ell}\}_{1\le k\le i+1, 1\le \ell\le i}$.
  Update the QR factorization of $H_{i+1,i}$ and compute $\|r_i\|_2 = |e_{i+1}^T Q e_1|$.
  **if** $\|r_i\|_2 \le \tau$ or $h_{i+1,i} = 0$ **then** break
  $v_{i+1} = w_i / h_{i+1,i}$
**end for**
$y_i = \arg\min_{y\in\mathbb{R}^i} \|b - A(x_0 + V_i y)\|_2 = \arg\min_{y\in\mathbb{R}^i} \|\beta e_1 - H_{i+1,i} y\|_2$
Return $x_i = x_0 + V_i y_i$.

**Preconditioning and Flexible GMRES** As for CG, for GMRES we can use left, right, or split preconditioning. In some cases, which of these options we choose can result in significantly different convergence behavior. Our primary concern here is not with the resulting convergence behavior, but with the implications of the choice of left or right preconditioning on backward stability guarantees.

The first analysis for a left-preconditioned GMRES method was given in [C1], where the preconditioner is constructed from LU factors computed in a potentially lower precision. It is shown that under the assumption that the preconditioned matrix is applied to a vector extra precisely (in double the working precision), the algorithm produces a solution to the preconditioned system with relative backward error on the order of the unit roundoff. Since the preconditioned system and the original system

have the same solution, this means that the relative forward error can be bounded in terms of the unit roundoff and the condition number of the *preconditioned matrix*, which we expect to be small. Note that in [C1] (and [C2] and other included works), it is important that preconditioned GMRES produce a small relative forward error since this is necessary for proving the convergence of the outer iterative refinement scheme. We also note that the work in [C1] has since been improved upon; see [4] for backward and forward error bounds for a variant which uses two general precisions, and see [131] for an analysis of mixed precision left-preconditioned GMRES with a general preconditioner.

In contrast with left-preconditioned GMRES, right-preconditioned GMRES poses a challenge if we wish to guarantee a small relative forward error. Even if we can prove that right-preconditioned GMRES produces a backward stable solution to the preconditioned linear system, we cannot bound the forward error in terms of the backward error times the condition number of the preconditioned matrix because the solutions to the preconditioned linear system and the original system are different. However, it is often desirable to use right or split preconditioning in practice. In the included work [C8], we present a complete analysis of the backward and forward errors in a four-precision split-preconditioned FGMRES method.

The FGMRES method is inherently a right-preconditioned method. Let $z_i$ represent the preconditioned basis vectors $v_i$, i.e., $z_i = M^{-1}v_i$. The name "flexible" comes because one can allow the preconditioner $M$ to change in each iteration, i.e., $z_i = M_i^{-1}v_i$. In FGMRES, these preconditioned basis vectors are stored in addition to the $v_i$'s, and the approximate solution is computed as $x_i = x_0 + Z_i y_i$ where $Z_i = [z_1, \ldots, z_i]$ (instead of $x_i = x_0 + M^{-1}V_i y_i$, as would be the case in right-preconditioned GMRES). For mathematical details, see, e.g., [105, Chapter 9.4]. The work [8] showed that in finite precision, FGMRES with a particular right-preconditioner is backward stable, while this is not true for GMRES, and that FGMRES is in general more robust than GMRES. The subsequent work [7] presented an analysis of the backward error in a two-precision FGMRES variant.

### 2.3.3 *s*-step variants

Krylov subspace methods, including Lanczos/CG and GMRES, are often limited by the cost of communication (i.e., data movement) at scale. This is because they often suffer from low computational intensity (the number of floating point operations performed per word moved) and require one or more global synchronizations in each iteration, requiring a collective communication between all processes involved in the computation in distributed memory settings. This has motivated the design of new algorithm variants which overcome these bottlenecks. In the class of Krylov subspace methods, examples include *s*-step (also called communication-avoiding) and pipelined variants, which are mathematically equivalent to their classical counterparts, meaning that in absence of errors, they would compute the exact same quantities. We give a brief overview of the derivation of *s*-step CG for illustration purposes. A full derivation of the Lanczos method can be found in the included work [C9]. For other *s*-step variants of Krylov subspace methods including historical references, see, e.g., [11] and [27].

The key idea of *s*-step CG (and *s*-step Krylov subspace methods in general) is to

compute a block $s \geq 1$ iterations at a time by expanding the Krylov subspace by $s$ dimensions and then performing a block orthogonalization. From Algorithm 2, notice that after iteration $i$, for $j = 0, \ldots, s$, the vectors $x_{i+j} - x_i$, $r_{i+j}$, and $p_{i+j}$ all lie in the subspace $\mathcal{K}_{s+1}(A, p_i) + \mathcal{K}_s(A, r_i)$. Thus we will compute, up front, a basis matrix $\mathcal{Y}$ such that $\text{span}(\mathcal{Y}) = \mathcal{K}_{s+1}(A, p_i) + \mathcal{K}_s(A, r_i)$. Then, for iterations $i + j$, $j = 0, \ldots, s$, rather than updating the length-$n$ vectors $x_{i+j} - x_i$, $r_{i+j}$, and $p_{i+j}$, we can update their length-$(2s + 1)$ coordinates $x'_j$, $r'_j$, and $p'_j$ in the basis encoded in the columns of $\mathcal{Y}$. That is, we have

$$x_{i+j} - x_i = \mathcal{Y}x'_j, \quad r_{i+j} = \mathcal{Y}r'_j, \quad p_{i+j} = \mathcal{Y}p'_j,$$

where $x'_0 = 0_{2s+1}, r'_0 = [0_{s+1}; 1; 0_{s-1}]$, and $p'_0 = [1; 0_{2s}]$.

The basis vectors can be generated according to any polynomial basis desired; monomial, Newton, and Chebyshev polynomials are common choices. Let $\mathcal{B}$ denote the $(s + 1) \times (s + 1)$ matrix that stores the polynomial coefficients, and let $\underline{\mathcal{Y}}$ be the same as $\mathcal{Y}$ but with columns $s+1$ and $2s$ replaced with zeros. We then have the relation $A\underline{\mathcal{Y}} = \mathcal{Y}\mathcal{B}$. We can therefore write the sparse matrix-vector product in each iteration of CG as

$$Ap_{i+j} = A\underline{\mathcal{Y}}p'_j = \mathcal{Y}(\mathcal{B}p'_j).$$

Further, denoting the Gram matrix $\mathcal{G} = \mathcal{Y}^T\mathcal{Y}$, the inner products can all be replaced by much smaller operations, e.g.,

$$r_{i+j}^T r_{i+j} = r_j'^T \mathcal{G} r'_j.$$

The resulting $s$-step CG algorithm is shown in Algorithm 4. We now discuss how this approach may reduce the communication cost in parallel settings (for a discussion of the sequential setting, see [11, Section 8]). Under certain assumptions on $s$ and the sparsity of $A$, the computation of the $2s+1$ basis vectors in line 4 can be accomplished for the same asymptotic communication cost of $O(1)$ sparse matrix-vector product using the communication-avoiding matrix powers kernel; see [11, Section 7]. Similarly, the computation of the Gram matrix in line 5 requires only a single global synchronization per $s$ iterations rather than the $O(s)$ that are required in CG. The inner loop (lines 8-12) can be performed locally by each processor without communication.

Unfortunately, the modifications made to improve performance in $s$-step Krylov subspace methods can also cause an algorithmic amplification of errors, including finite precision errors. This can result in both reduced attainable accuracy and significant convergence delays, which can potentially negate any performance advantage. This has been studied extensively by the present author for Lanczos/CG-based algorithms; see, e.g., [27, 30]. In short, the effects of finite precision errors on convergence delay and attainable accuracy grow worse as the $s$-step basis becomes more ill-conditioned.

In [C9] we show, theoretically and experimentally, that the undesirable convergence delay caused by the $s$-step formulation can be mitigated by the use of extra precision in certain computations involving the Gram matrices $\mathcal{G}_k$. Since these matrices are small, the extra computational cost is expected to be minimal, and the asymptotic communication cost is not affected. This was confirmed experimentally under certain scenarios in [C10], where a residual replacement strategy adapted from [21] was used in combination with the mixed precision approach to also improve the attainable accuracy in a multi-GPU environment.

**Algorithm 4** $s$-step Conjugate Gradient ($s$-step CG)

**Require:** Symmetric positive definite matrix $A \in \mathbb{R}^{n \times n}$; right-hand side $b$; initial approximation $x_0$; given stopping criterion; maximum number of iterations $n_{\max}$, block size $s$.

$r_0 = b - Ax_0$
$p_0 = r_0$
**for** $k = 0, \ldots, n_{\max}/s$ **do**
    Compute $\mathcal{Y}_k$ and $\mathcal{B}_k$ such that $A\underline{\mathcal{Y}}_k = \mathcal{Y}_k\mathcal{B}_k$ and $\text{span}(\mathcal{Y}_k) = \mathcal{K}_{s+1}(A, p_{sk}) + \mathcal{K}_s(A, r_{sk})$.
    $\mathcal{G}_k = \mathcal{Y}_k^T\mathcal{Y}_k$
    $x'_0 = 0_{2s+1}, r'_0 = [0_{s+1}; 1; 0_{s-1}], p'_0 = [1; 0_{2s}]$
    **for** $j = 1 : s$ **do**
        $\alpha_{sk+j-1} = (r'^T_{j-1}\mathcal{G}_k r'_{j-1})/(p'^T_{j-1}\mathcal{G}_k\mathcal{B}_k p'_{j-1})$
        $x'_j = x'_{j-1} + \alpha_{sk+j-1}p'_{j-1}$
        $r'_j = r'_{j-1} - \alpha_{sk+j-1}\mathcal{B}_k p'_{j-1}$
        $\beta_{sk+j} = (r'^T_j\mathcal{G}_k r'_j)/(r'^T_{j-1}\mathcal{G}_k r'_{j-1})$
        $p'_j = r'_j + \beta_{sk+j}p'_{j-1}$
    **end for**
    $[x_{s(k+1)} - x_{sk}, r_{s(k+1)}, p_{s(k+1)}] = \mathcal{Y}_k[x'_s, r'_s, p'_s]$
**end for**

# Chapter 3

# Introduction to least squares problems

Included works on mixed precision algorithms in this thesis involve the solution of standard least squares problems ([C4] and [C6]) as well as total least squares problems ([C5]). Here we give a brief overview of these topics with regard to the content of this thesis. This chapter should not be considered a comprehensive overview; many topics, including perturbation theory, a survey of algorithms, and other types of least squares problems such as generalized least squares problems and constrained least squares problems, are omitted. For a comprehensive overview of the numerical solution of least squares problems, we direct the reader to [16].

## 3.1 Standard least squares problems

Here we consider the standard least squares problem

$$\min_x \|b - Ax\|_2, \tag{3.1}$$

where $A$ is an $m \times n$ matrix, where $m \geq n$ and $A$ has rank $n$. One method for solving this problem is via the QR factorization of $A$. Let

$$A = Q \begin{bmatrix} R \\ 0 \end{bmatrix}$$

be the QR factorization of $A$, where $Q = [Q1, Q2] \in \mathbb{R}^{m \times m}$ is an orthogonal matrix with $Q_1 \in \mathbb{R}^{m \times n}$ and $Q_2 \in \mathbb{R}^{(m-n) \times n}$, and $R$ is $n \times n$ upper triangular. The solution to the standard least squares problem is then given by $x = R^{-1} Q_1^T b$ and $\|b - Ax\|_2 = \|Q_2^T b\|_2$. For an overview of the numerical stability of this and other approaches to solving the standard least squares problem, see, e.g., [63, Chapter 20].

Least squares problems arise within two contexts in this thesis. The first is in the development of a mixed precision Krylov subspace-based iterative refinement approach [C4]. The second is in the construction of low-precision sparse approximate inverse preconditioners [C6]. In the following subsection, we discuss the iterative refinement of least squares problems.

### 3.1.1 Iterative refinement

As for linear systems, if the matrix $A$ is ill-conditioned, it may be desirable to perform iterative refinement for least squares problems in order to improve the accuracy. If the least squares problem is nearly consistent (i.e., the residual $b - Ax$ is very close to 0 for the least squares solution $x$), then it is possible to use the refinement procedure $r_{k-1} = b - Ax_{k-1}$, $e_k = R^{-1}Q^T r_{k-1}$, $x_k = x_{k-1} + e_k$. This refinement procedure has been studied by Golub [47] and was also used by Bauer [13]. That this strategy only works in the case of nearly consistent systems was first pointed out by Golub and Wilkinson [49]. For inconsistent systems, the residual may not converge to the working precision and the forward error may be arbitrarily large. The issue is that when the residual is not close to zero, it is necessary to refine *both* the approximate least squares solution $x$ and the residual $r$.

A clever way of accomplishing this was developed by Björck [15]. The insight is that the least squares problem (3.1) can be written as the augmented system

$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}. \tag{3.2}$$

Notice that the above is equivalent to the normal equations $A^T Ax = A^T b$, the solution of which gives the solution to (3.1). This is an $(m+n) \times (m+n)$ linear system, and thus the usual iterative refinement scheme for linear systems can be applied; see Chapter 2.2.1. Of course, we do not want to explicitly compute an LU factorization of the augmented matrix as in standard iterative refinement for linear systems. Björck has shown that in each step of iterative refinement, one can solve linear systems with the augmented matrix using a QR factorization of $A$; see [15, Section 5] for a full finite precision analysis of a two-precision based approach, in which extra precision is used in computing the residuals. Björck [15] also gives details on how a simple scaling can be applied to the linear system (3.2) in order to minimize the condition number of the augmented matrix.

In the included work [C4], we extend the analysis of Björck to the three-precision case, in which a third, potentially lower precision may be used to compute the QR factorization. We then develop a new GMRES-based approach to iterative refinement for least squares problems. The primary difference with the general linear system case is that we now wish to use the computed QR factors of $A$ in constructing a suitable left preconditioner for the augmented system. We present one possibility and prove constraints under which forward and backward stability of the refinement process can be attained. Note that the augmented matrix in (3.2) is a saddle point matrix, and there is a wealth of work on developing preconditioners for saddle point systems; see, e.g., [104]. In [C4], we also show experimentally that the commonly-used block diagonal preconditioner for saddle point systems [90] works well within three-precision GMRES-based iterative refinement in practice. We note that another approach to Krylov subspace method-based iterative refinement for least squares problems is presented in [65], which uses a Cholesky factorization of $A^T A$ and performs iterative refinement on the normal equations.

## 3.2 Total least squares problems

Standard least squares problems are based on the assumption of a standard linear model, in which the linear relation

$$Ax = b + r$$

holds, where $A$ is a given $m \times n$ matrix, $b$ is a length-$m$ vector, and $r$ is a length-$m$ vector of random, uncorrelated errors with zero mean and identical variance. In practical applications, such assumptions may not be realistic. In particular, the matrix $A$ itself may also be subject to errors, including modeling errors, sampling errors, etc. The *total least squares* (TLS) problem arises from considering this case. Under what is frequently called the error-in-variables model, we now assume the model

$$(A + E)x = b + r.$$

The matrix $[E, r]$ is called the error matrix, and we now assume that the rows of this matrix are independently and identically distributed, again with zero mean and identical variance. The total least squares problem can thus be formulated as

$$\min_{E,r} \|[E, r]\|_F, \quad \text{subject to} \quad (A + E)x = b + r.$$

Any $x$ which satisfies the above is a solution to the TLS problem.

Notice that in order for the solution to be unique, the matrix $[A+E, b+r]$ must have exactly $n$ linearly independent columns. In other words, we want to find the matrix $[E, r]$ of smallest Frobenius norm such that $[A + E, b + r]$ becomes rank deficient. The exact solution to the TLS problem can thus be stated in terms of the SVD of the matrix $[A, b]$. Let $[A, b] = U\Sigma V^T$ where $U$ is $m \times (n + 1)$ with orthonormal columns, $V = [v_1, \ldots, v_{n+1}]$ is an $(n+1) \times (n+1)$ orthogonal matrix, and $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_{n+1})$ with $\sigma_1 \geq \cdots \geq \sigma_{n+1}$. If $\sigma_{n+1} = 0$, then we know that $[E, r] = 0$. Otherwise, we have the solution

$$[E, r] = -\sigma_{n+1} v_{n+1} v_{n+1}^T,$$

for which $\|[E, r]\|_F = \sigma_{n+1}$, and the solution to the TLS problem is given by

$$x_{TLS} = -\frac{1}{v_{n+1,n+1}} [v_{1,n+1}, v_{n,n+1}]^T.$$

Notice that if $v_{n+1,n+1} = 0$, the TLS solution does not exist; see also the examples and exposition in [48] and [16, Chapter 4.6]. Letting the singular values of $A$ be denoted by $\sigma'_1 \geq \cdots \geq \sigma'_n$, this condition will not occur as long as $\sigma'_n > \sigma_{n+1}$.

The first analysis of a numerically stable approach for solving the TLS problem was given by Golub and Van Loan [48]. See, e.g., [111] and [126] for other early works. Expositions on the TLS problem can be found in [129] and [16, Chapter 4.6]. Paige and Strakoš have also studied (scaled) total least squares problems [98, 99]. The latter of these works involves the development of an elegant theory for scaled TLS problems based on the so-called "core problem". The core problem formulation allows one to decompose a TLS problem into two independent parts, separating out the necessary and sufficient information from the data, such that one of these parts contains the unique TLS solution. Further developments involving the core problem formulation as it relates to TLS problems can be found in, e.g., [100], [68], [66], [67], and [69].

### 3.2.1 Rayleigh quotient iteration for TLS problems

Although the SVD is the classic approach to solving TLS problems, it can be computationally expensive. One potential alternative due to Van Huffel and Zha is based on a rank-revealing complete orthogonal decomposition [130]. Various iterative approaches are also possible, such as inverse iteration and inverse Chebyshev iteration [128].

A particular iterative approach suitable for large-scale problems is due to Björck et al. [17]. This approach, called RQI-PCGTLS, is based on Rayleigh quotient iteration (RQI) with preconditioned conjugate gradient (PCG) as the inner solver. In the included work [C5] we develop a three-precision variant of this approach. We thus briefly introduce the RQI-PCGTLS algorithm.

The Rayleigh quotient of a Hermitian matrix $B$ and a vector $x$ is the scalar quantity

$$\rho(x) = \frac{x^T B x}{x^T x}.$$

If $x$ is an eigenvector of $B$, then $\rho(x)$ is the corresponding eigenvalue. Rayleigh quotient iteration is an algorithm for finding the smallest eigenvalue and eigenvector of a given matrix $B$. It is equivalent to inverse iteration (the power method with $B^{-1}$) with a shift equal to the Rayleigh quotient. It is known that this method is cubically convergent.

Rayleigh quotient iteration can be used to solve the total least squares problem as follows. The value $\lambda = \sigma_{n+1}^2$ and $x = x_{TLS}$ satisfy

$$\begin{bmatrix} A^T A & A^b \\ b^T A & b^T b \end{bmatrix} \begin{bmatrix} x \\ -1 \end{bmatrix} = \lambda \begin{bmatrix} x \\ -1 \end{bmatrix}, \tag{3.3}$$

and thus we can use RQI to solve this eigenvalue problem. In order to ensure that RQI converges to the smallest eigenvalue and thus to the TLS solution, one or more steps of inverse iteration can be used to find a suitable starting vector [116].

Iteration $k$ of Rayleigh quotient iteration on the augmented matrix in (3.3) requires solving two linear systems with the coefficient matrix $A^T A - \sigma_k I$, where $\sigma_k$ is the Rayleigh quotient computed in iteration $k$. Under the assumption that a solution to the TLS problem exists, this matrix is symmetric positive definite, and thus a preconditioned conjugate gradient algorithm can be used to solve these linear systems. This results in the RQI-PCGTLS algorithm of Björck et al. [17], in which they advocate using a fixed number of PCG iterations and using the Cholesky factor of $A^T A$ as a preconditioner. For details, see [17].

The structure of the RQI-PCGTLS algorithm bears resemblance to the Krylov subspace method-based iterative refinement approaches discussed in Chapter 2.2.1. Namely, we have an iterative outer-inner solve approach, in which the inner solve is performed via a preconditioned Krylov subspace method. This leads to the intuition that lower precision can likely be used in parts of the RQI-PCGTLS algorithm. In particular, depending on the size of $A$ and the relative dimensions $m$ and $n$, the most expensive computation is likely the computation of the preconditioner for the system with the coefficient matrix $A^T A - \sigma_k I$. In the included work [C5], we develop a mixed precision variant of the algorithm, which we call RQI-PCGLTS-MP. Here we advocate for the use of the $R$ factor from a low-precision QR factorization of $A$ as the preconditioner within the PCG inner solve. Our analysis provides theoretical constraints on how the precision used for the QR factorization should be chosen to ensure stability.

Perhaps unsurprisingly, in contrast with the standard least squares case, for TLS, a suitable choice of precision depends not only on the matrix $A$ but also on the right-hand side $b$.

# Bibliography


[1] A. Abdelfattah, H. Anzt, E. G. Boman, E. Carson, T. Cojean, J. Dongarra, M. Gates, T. Grützmacher, N. J. Higham, S. Li, N. Lindquist, Y. Liu, J. Loe, P. Luszczek, P. Nayak, S. Pranesh, S. Rajamanickam, T. Ribizel, B. Smith, K. Świrydowicz, S. Thomas, S. Tomov, Y. M. Tsai, I. Yamazaki, and U. M. Yang, *A survey of numerical methods utilizing mixed precision arithmetic*, The International Journal of High Performance Computing Applications, 35 (2021), pp. 344–369.

[2] A. Abdelfattah, S. Tomov, and J. Dongarra, *Investigating the benefit of FP16-enabled mixed-precision solvers for symmetric positive definite matrices using GPUs*, in International Conference on Computational Science, Springer, 2020, pp. 237–250.

[3] H. Al Daas, T. Rees, and J. Scott, *Two-level Nyström-Schur preconditioner for sparse symmetric positive definite matrices*, SIAM Journal on Scientific Computing, 43 (2021), pp. A3837 – A3861.

[4] P. Amestoy, A. Buttari, N. J. Higham, J.-Y. L'Excellent, T. Mary, and B. Vieublé, *Five-precision GMRES-based iterative refinement*, MIMS EPrint 2021.5, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, April 2021. Revised April 2022.

[5] P. Amestoy, A. Buttari, N. J. Higham, J.-Y. L'Excellent, T. Mary, and B. Vieublé, *Combining sparse approximate factorizations with mixed-precision iterative refinement*, ACM Transactions on Mathematical Software, 49 (2023), pp. 1–29.

[6] H. Anzt, J. Dongarra, G. Flegar, N. J. Higham, and E. S. Quintana-Ortí, *Adaptive precision in block-Jacobi preconditioning for iterative sparse linear system solvers*, Concurrency and Computation: Practice and Experience, 31 (2019), p. e4460.

[7] M. Arioli and I. S. Duff, *Using FGMRES to obtain backward stability in mixed precision*, Electronic Transactions on Numerical Analysis, 33 (2009), pp. 31–44.

[8] M. Arioli, I. S. Duff, S. Gratton, and S. Pralet, *A note on GMRES preconditioned by a perturbed $LDL^T$ decomposition with static pivoting*, SIAM Journal on Scientific Computing, 29 (2007), pp. 2024–2044.

[9] M. Arioli, V. Pták, and Z. Strakoš, *Krylov sequences of maximal length and convergence of GMRES*, BIT Numerical Mathematics, 38 (1998), pp. 636–643.

[10] H. Avron, P. Maymounkov, and S. Toledo, *Blendenpik: Supercharging LAPACK's least-squares solver*, SIAM Journal on Scientific Computing, 32 (2010), pp. 1217–1236.

[11] G. Ballard, E. Carson, J. Demmel, M. Hoemmen, N. Knight, and O. Schwartz, *Communication lower bounds and optimal algorithms for numerical linear algebra*, Acta Numerica, 23 (2014), pp. 1–155.

[12] J. L. Barlow, *Block modified Gram–Schmidt algorithms and their analysis*, SIAM Journal on Matrix Analysis and Applications, 40 (2019), pp. 1257–1290.

[13] F. L. Bauer, *Elimination with weighted row combinations for solving linear equations and least squares problems*, Numerische Mathematik, 7 (1965), pp. 338–352.

[14] *Bfloat16 – hardware numerics definition*, Tech. Rep. 338302-001US, Revision 1.0, Intel, November 2018.

[15] Å. Björck, *Iterative refinement of linear least squares solutions I*, BIT Numerical Mathematics, 7 (1967), pp. 257–278.

[16] ——, *Numerical methods for least squares problems*, SIAM, 1996.

[17] Å. Björck, P. Heggernes, and P. Matstoms, *Methods for large scale total least squares problems*, SIAM Journal on Matrix Analysis and Applications, 22 (2000), pp. 413–429.

[18] Z. Bujanović, D. Kressner, and C. Schröder, *Iterative refinement of Schur decompositions*, Numerical Algorithms, 92 (2023), pp. 247–267.

[19] N. Buoncristiani, S. Shah, D. Donofrio, and J. Shalf, *Evaluating the numerical stability of posit arithmetic*, in Proceedings of the 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS), IEEE, 2020, pp. 612–621.

[20] A. Buttari, J. Dongarra, J. Langou, J. Langou, P. Luszczek, and J. Kurzak, *Mixed precision iterative refinement techniques for the solution of dense linear systems*, The International Journal of High Performance Computing Applications, 21 (2007), pp. 457–466.

[21] E. Carson and J. Demmel, *A residual replacement strategy for improving the maximum attainable accuracy of s-step Krylov subspace methods*, SIAM Journal on Matrix Analysis and Applications, 35 (2014), pp. 22–43.

[22] E. Carson and J. W. Demmel, *Accuracy of the s-step Lanczos method for the symmetric eigenproblem in finite precision*, SIAM Journal on Matrix Analysis and Applications, 36 (2015), pp. 793–819.

[23] E. Carson, J. Liesen, and Z. Strakoš, *Solving linear algebraic equations with Krylov subspace methods is still interesting!*, arXiv preprint arXiv:2211.00953, (2023).

[24] E. Carson, K. Lund, and M. Rozložník, *The stability of block variants of classical Gram–Schmidt*, SIAM Journal on Matrix Analysis and Applications, 42 (2021), pp. 1365–1380.

[25] E. Carson, K. Lund, M. Rozložník, and S. Thomas, *Block Gram-Schmidt algorithms and their stability properties*, Linear Algebra and its Applications, 638 (2022), pp. 150–195.

[26] E. Carson and Z. Strakoš, *On the cost of iterative computations*, Philosophical Transactions of the Royal Society A, 378 (2020), p. 20190050.

[27] E. C. Carson, *Communication-Avoiding Krylov Subspace Methods in Theory and Practice*, PhD thesis, University of California - Berkeley, 2015.

[28] E. C. Carson, *The adaptive s-step conjugate gradient method*, SIAM Journal on Matrix Analysis and Applications, 39 (2018), pp. 1318–1338.

[29] E. C. Carson, *An adaptive s-step conjugate gradient algorithm with dynamic basis updating*, Applications of Mathematics, 65 (2020), pp. 123–151.

[30] E. C. Carson, M. Rozložník, Z. Strakoš, P. Tichý, and M. Tůma, *The numerical stability analysis of pipelined conjugate gradient methods: Historical context and methodology*, SIAM Journal on Scientific Computing, 40 (2018), pp. A3549–A3580.

[31] T. Chen and E. Carson, *Predict-and-recompute conjugate gradient variants*, SIAM Journal on Scientific Computing, 42 (2020), pp. A3084–A3108.

[32] I. Daužickaitė, A. S. Lawless, J. A. Scott, and P. J. van Leeuwen, *Randomised preconditioning for the forcing formulation of weak constraint 4D-Var*, Quarterly Journal of the Royal Meteorological Society, 147 (2021), pp. 3719 – 3734.

[33] J. Demmel, Y. Hida, W. Kahan, X. S. Li, S. Mukherjee, and E. J. Riedy, *Error bounds from extra-precise iterative refinement*, ACM Transactions on Mathematical Software, 32 (2006), pp. 325–351.

[34] J. J. Dongarra, *Algorithm 589: SICEDR: A FORTRAN subroutine for improving the accuracy of computed matrix eigenvalues*, ACM Transactions on Mathematical Software, 8 (1982), pp. 371–375.

[35] J. J. Dongarra, C. B. Moler, and J. H. Wilkinson, *Improving the accuracy of computed eigenvalues and eigenvectors*, SIAM Journal on Numerical Analysis, 20 (1983), pp. 23–45.

[36] P. Drineas and M. W. Mahoney, *RandNLA: Randomized numerical linear algebra*, Communications of the ACM, 59 (2016), pp. 80–90.

[37] J. Drkošová, A. Greenbaum, M. Rozložník, and Z. Strakoš, *Numerical stability of GMRES*, BIT Numerical Mathematics, 35 (1995), pp. 309–330.

[38] M. Emans and A. van der Meer, *Mixed-precision AMG as linear equation solver for definite systems*, Procedia Computer Science, 1 (2010), pp. 175–183.

[39] G. Flegar, H. Anzt, T. Cojean, and E. S. Quintana-Orti, *Adaptive precision block-Jacobi for high performance preconditioning in the Ginkgo linear algebra software*, ACM Transactions on Mathematical Software, 47 (2021), pp. 1–28.

[40] Z. Frangella, J. A. Tropp, and M. Udell, *Randomized Nyström preconditioning*, SIAM J. Matrix Anal. Appl., 44 (2023), pp. 718–752.

[41] L. Giraud, S. Gratton, and J. Langou, *Convergence in backward error of relaxed GMRES*, SIAM Journal on Scientific Computing, 29 (2007), pp. 710–728.

[42] L. Giraud, A. Haidar, and L. T. Watson, *Mixed-precision preconditioners in parallel domain decomposition solvers*, in Domain decomposition methods in science and engineering XVII, Springer, 2008, pp. 357–364.

[43] A. Gittens and M. W. Mahoney, *Revisiting the Nyström method for improved large-scale machine learning*, J. Mach. Learn. Res., 17 (2016), pp. 3977 – 4041.

[44] F. Göbel, T. Grützmacher, T. Ribizel, and H. Anzt, *Mixed precision incomplete and factorized sparse approximate inverse preconditioning on GPUs*, in Euro-Par 2021: Parallel Processing, Springer, 2021, pp. 550–564.

[45] D. Göddeke and R. Strzodka, *Cyclic reduction tridiagonal solvers on GPUs applied to mixed-precision multigrid*, IEEE Transactions on Parallel and Distributed Systems, 22 (2010), pp. 22–32.

[46] D. Göddeke, R. Strzodka, and S. Turek, *Performance and accuracy of hardware-oriented native-, emulated- and mixed-precision solvers in FEM simulations*, International Journal of Parallel, Emergent, and Distributed Systems, 22 (2007), pp. 221–256.

[47] G. Golub, *Numerical methods for solving linear least squares problems*, Numerische Mathematik, 7 (1965), pp. 206–216.

[48] G. H. Golub and C. F. Van Loan, *An analysis of the total least squares problem*, SIAM Journal on Numerical Analysis, 17 (1980), pp. 883–893.

[49] G. H. Golub and J. H. Wilkinson, *Note on the iterative refinement of least squares solution*, Numerische Mathematik, 9 (1966), pp. 139–148.

[50] S. Gratton, E. Simon, D. Titley-Peloquin, and P. Toint, *Exploiting variable precision in GMRES*, arXiv preprint arXiv:1907.10550, (2019).

[51] A. Greenbaum, *Behavior of slightly perturbed Lanczos and conjugate-gradient recurrences*, Linear Algebra and its Applications, 113 (1989), pp. 7–63.

[52] ——, *Estimating the attainable accuracy of recursively computed residual methods*, SIAM Journal on Matrix Analysis and Applications, 18 (1997), pp. 535–551.

[53] A. Greenbaum, V. Pták, and Z. Strakoš, *Any nonincreasing convergence curve is possible for GMRES*, SIAM Journal on Matrix Analysis and Applications, 17 (1996), pp. 465–469.

[54] M. J. Grote and T. Huckle, *Parallel preconditioning with sparse approximate inverses*, SIAM Journal on Scientific Computing, 18 (1997), pp. 838–853.

[55] J. L. Gustafson and I. T. Yonemoto, *Beating floating point at its own game: Posit arithmetic*, Supercomputing Frontiers and Innovations, 4 (2017), pp. 71–86.

[56] M. H. Gutknecht and Z. Strakoš, *Accuracy of two three-term and three two-term recurrences for Krylov space solvers*, SIAM Journal on Matrix Analysis and Applications, 22 (2000), pp. 213–229.

[57] A. Haidar, S. Tomov, J. Dongarra, and N. J. Higham, *Harnessing GPU tensor cores for fast FP16 arithmetic to speed up mixed-precision iterative refinement solvers*, in Proceedings of the 2018 International Conference for High Performance Computing, Networking, Storage and Analysis (SC18), IEEE, 2018, pp. 603–613.

[58] N. Halko, P.-G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Review, 53 (2011), pp. 217–288.

[59] S. Hatfield, P. Düben, M. Chantry, K. Kondo, T. Miyoshi, and T. Palmer, *Choosing the optimal numerical precision for data assimilation in the presence of model error*, Journal of Advances in Modeling Earth Systems, 10 (2018), pp. 2177–2191.

[60] J. L. Hennessy and D. A. Patterson, *A new golden age for computer architecture*, Communications of the ACM, 62 (2019), pp. 48–60.

[61] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, Journal of Research of the National Bureau of Standards, 49 (1952), pp. 409–436.

[62] N. J. Higham, *Iterative refinement enhances the stability of QR factorization methods for solving linear equations*, BIT Numerical Mathematics, 31 (1991), pp. 447–468.

[63] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, second ed., 2002.

[64] N. J. Higham and T. Mary, *Mixed precision algorithms in numerical linear algebra*, Acta Numerica, 31 (2022), pp. 347–414.

[65] N. J. Higham and S. Pranesh, *Exploiting lower precision arithmetic in solving symmetric positive definite linear systems and least squares problems*, SIAM Journal on Scientific Computing, 43 (2021), pp. A258–A277.

[66] I. Hnětynková, M. Plešinger, D. M. Sima, Z. Strakoš, and S. Van Huffel, *The total least squares problem in $AX \approx B$: a new classification with the relationship to the classical works*, SIAM Journal on Matrix Analysis and Applications, 32 (2011), pp. 748–770.

[67] I. Hnětynková, M. Plešinger, and Z. Strakoš, *The core problem within a linear approximation problem $AX \approx B$ with multiple right-hand sides*, SIAM Journal on Matrix Analysis and Applications, 34 (2013), pp. 917–931.

[68] I. Hnětynková and Z. Strakoš, *Lanczos tridiagonalization and core problems*, Linear Algebra and its Applications, 421 (2007), pp. 243–251.

[69] I. Hnětynková, M. Plešinger, and Z. Strakoš, *Band generalization of the Golub–Kahan bidiagonalization, generalized Jacobi matrices, and the core problem*, SIAM Journal on Matrix Analysis and Applications, 36 (2015), pp. 417–434.

[70] J. D. Hogg and J. A. Scott, *A fast and robust mixed-precision solver for the solution of sparse symmetric linear systems*, ACM Transactions on Mathematical Software, 37 (2010), pp. 17:1–17:24.

[71] *HPL-MxP mixed-precision benchmark*. `https://hpl-mxp.org/`, 2023.

[72] M. Jankowski and H. Woźniakowski, *Iterative refinement implies numerical stability*, BIT Numerical Mathematics, 17 (1977), pp. 303–311.

[73] A. Jennings, *Influence of the eigenvalue spectrum on the convergence rate of the conjugate gradient method*, IMA Journal of Applied Mathematics, 20 (1977), pp. 61–72.

[74] C. T. Kelley, *Newton's method in mixed precision*, SIAM Review, 64 (2022), pp. 191–211.

[75] D. Kressner, Y. Ma, and M. Shao, *A mixed precision LOBPCG algorithm*, Numerical Algorithms, (2023), pp. 1–19.

[76] M. Kronbichler and K. Ljungkvist, *Multigrid for matrix-free high-order finite element computations on graphics processors*, ACM Transactions on Parallel Computing, 6 (2019), pp. 1–32.

[77] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, Journal of Research of the National Bureau of Standards, 45 (1950), pp. 255–282.

[78] J. Langou, J. Langou, P. Luszczek, J. Kurzak, A. Buttari, and J. Dongarra, *Exploiting the performance of 32 bit floating point arithmetic in obtaining 64 bit accuracy (revisiting iterative refinement for linear systems)*, in Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, 2006.

[79] J. Liesen and Z. Strakoš, *Krylov subspace methods: Principles and analysis*, Oxford University Press, 2013.

[80] J. Liesen and P. Tichý, *The worst-case GMRES for normal matrices*, BIT Numerical Mathematics, 44 (2004), pp. 79–98.

[81] N. Lindquist, P. Luszczek, and J. Dongarra, *Accelerating restarted GMRES with mixed precision arithmetic*, IEEE Transactions on Parallel and Distributed Systems, 33 (2021), pp. 1027–1037.

[82] P. Lindstrom, S. Lloyd, and J. Hittinger, *Universal coding of the reals: Alternatives to IEEE floating point*, in Proceedings of the Conference for Next Generation Arithmetic, 2018, pp. 1–14.

[83] J. Málek and Z. Strakoš, *Preconditioning and the conjugate gradient method in the context of solving PDEs*, SIAM Spotlights, SIAM, 2015.

[84] J. Málek and Z. Strakoš, *Preconditioning and the conjugate gradient method in the context of solving PDEs*, vol. 1 of SIAM Spotlights, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2015.

[85] P.-G. Martinsson and J. Tropp, *Randomized numerical linear algebra: Foundations and algorithms*, Acta Numerica, 29 (2020), pp. 403–572.

[86] S. F. McCormick, J. Benzaken, and R. Tamstorf, *Algebraic error analysis for mixed-precision multigrid solvers*, SIAM Journal on Scientific Computing, 43 (2021), pp. S392–S419.

[87] M. Meier, Y. Nakatsukasa, A. Townsend, and M. Webb, *Are sketch-and-precondition least squares solvers numerically stable?*, arXiv preprint arXiv:2302.07202, (2023).

[88] G. Meurant and Z. Strakoš, *The Lanczos and conjugate gradient algorithms in finite precision arithmetic*, Acta Numerica, 15 (2006), pp. 471–542.

[89] C. B. Moler, *Iterative refinement in floating point*, Journal of the ACM, 14 (1967), pp. 316–321.

[90] M. F. Murphy, G. H. Golub, and A. J. Wathen, *A note on preconditioning for indefinite linear systems*, SIAM Journal on Scientific Computing, 21 (2000), pp. 1969–1972.

[91] T. Ogita and K. Aishima, *Iterative refinement for symmetric eigenvalue decomposition*, Japan Journal of Industrial and Applied Mathematics, 35 (2018), pp. 1007–1035.

[92] ——, *Iterative refinement for symmetric eigenvalue decomposition II: clustered eigenvalues*, Japan Journal of Industrial and Applied Mathematics, 36 (2019), pp. 435–459.

[93] ——, *Iterative refinement for singular value decomposition based on matrix multiplication*, Journal of Computational and Applied Mathematics, 369 (2020), p. 112512.

[94] C. C. Paige, *Computational variants of the Lanczos method for the eigenproblem*, IMA Journal of Applied Mathematics, 10 (1972), pp. 373–381.

[95] ——, *Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix*, IMA Journal of Applied Mathematics, 18 (1976), pp. 341–349.

[96] C. C. Paige, *Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem*, Linear Algebra and its Applications, 34 (1980), pp. 235–258.

[97] C. C. Paige, M. Rozložník, and Z. Strakoš, *Modified Gram-Schmidt (MGS), least squares, and backward stability of MGS-GMRES*, SIAM Journal on Matrix Analysis and Applications, 28 (2006), pp. 264–284.

[98] C. C. Paige and Z. Strakoš, *Bounds for the least squares distance using scaled total least squares*, Numerische Mathematik, 91 (2002), pp. 93–115.

[99] ——, *Scaled total least squares fundamentals*, Numerische Mathematik, 91 (2002), pp. 117–146.

[100] C. C. Paige and Z. Strakoš, *Core problems in linear algebraic systems*, SIAM Journal on Matrix Analysis and Applications, 27 (2005), pp. 861–875.

[101] M. Petschow, E. S. Quintana-Ortí, and P. Bientinesi, *Improved accuracy and parallelism for MRRR-based eigensolvers—a mixed precision approach*, SIAM Journal on Scientific Computing, 36 (2014), pp. C240–C263.

[102] J. K. Reid, *On the method of conjugate gradients for the solution of large sparse systems of linear equations*, in Large Sparse Sets of Linear Equations (Proc. Conf. St. Catherine's Coll., Oxford, 1970), 1971, pp. 231–254.

[103] V. Rokhlin and M. Tygert, *A fast randomized algorithm for overdetermined linear least-squares regression*, Proceedings of the National Academy of Sciences, 105 (2008), pp. 13212–13217.

[104] M. Rozložník, *Saddle-point problems and their iterative solution*, Springer, 2018.

[105] Y. Saad, *Iterative methods for sparse linear systems*, SIAM, 2003.

[106] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM Journal on Scientific and Statistical Computing, 7 (1986), pp. 856–869.

[107] J. Scott and M. Tůma, *Algorithms for Sparse Linear Systems*, Springer Nature, 2023.

[108] V. Simoncini and D. B. Szyld, *Theory of inexact Krylov subspace methods and applications to scientific computing*, SIAM Journal on Scientific Computing, 25 (2003), pp. 454–477.

[109] R. D. Skeel, *Iterative refinement implies numerical stability for Gaussian elimination*, Mathematics of Computation, 35 (1980), pp. 817–832.

[110] G. L. Sleijpen and H. A. Van der Vorst, *Maintaining convergence properties of BiCGstab methods in finite precision arithmetic*, Numerical Algorithms, 10 (1995), pp. 203–223.

[111] G. W. Stewart, *On the invariance of perturbed null vectors under column scaling*, Numerische Mathematik, 44 (1984), pp. 61–65.

[112] Z. Strakoš, *On the real convergence rate of the conjugate gradient method*, Linear Algebra and its Applications, 154 (1991), pp. 535–549.

[113] Z. Strakoš and P. Tichý, *On error estimation in the conjugate gradient method and why it works in finite precision computations*, Electronic Transactions on Numerical Analysis, 13 (2002), pp. 56–80.

[114] Y. Sumiyoshi, A. Fujii, A. Nukada, and T. Tanaka, *Mixed-precision AMG method for many core accelerators*, in Proceedings of the 21st European MPI Users' Group Meeting, 2014, pp. 127–132.

[115] K. Świrydowicz, J. Langou, S. Ananthan, U. Yang, and S. Thomas, *Low synchronization Gram–Schmidt and generalized minimal residual algorithms*, Numerical Linear Algebra with Applications, 28 (2021), p. e2343.

[116] D. B. Szyld, *Criteria for combining inverse and Rayleigh quotient iteration*, SIAM Journal on Numerical Analysis, 25 (1988), pp. 1369–1375.

[117] R. Tamstorf, J. Benzaken, and S. F. McCormick, *Discretization-error-accurate mixed-precision multigrid solvers*, SIAM Journal on Scientific Computing, 43 (2021), pp. S420–S447.

[118] *TensorFloat-32 in the A100 GPU accelerates AI training, HPC up to 20x.* NVIDIA, `https://blogs.nvidia.com/blog/2020/05/14/tensorfloat-32-precision-format/`, May 2020.

[119] S. Thomas, E. Carson, M. Rozložník, A. Carr, and K. Świrydowicz, *Iterated Gauss–Seidel GMRES*, SIAM Journal on Scientific Computing, (2023), pp. S254–S279.

[120] F. Tisseur, *Newton's method in floating point arithmetic and iterative refinement of generalized eigenvalue problems*, SIAM Journal on Matrix Analysis and Applications, 22 (2001), pp. 1038–1057.

[121] *TOP500 list.* `https://www.top500.org/lists/top500/`, June 2023.

[122] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher, *Fixed-rank approximation of a positive-semidefinite matrix from streaming data*, Advances in Neural Information Processing Systems, 30 (2017).

[123] Y. M. Tsai, P. Luszczek, and J. Dongarra, *Mixed-precision algorithm for finding selected eigenvalues and eigenvectors of symmetric and Hermitian matrices*, in 2022 IEEE/ACM Workshop on Latest Advances in Scalable Algorithms for Large-Scale Heterogeneous Systems (ScalAH), IEEE, 2022, pp. 43–50.

[124] K. Turner and H. F. Walker, *Efficient high accuracy solutions with GMRES(m)*, SIAM Journal on Scientific and Statistical Computing, 13 (1992), pp. 815–825.

[125] J. Van Den Eshof and G. L. Sleijpen, *Inexact Krylov subspace methods for linear systems*, SIAM Journal on Matrix Analysis and Applications, 26 (2004), pp. 125–153.

[126] A. Van Der Sluis and G. W. Veltkamp, *Restoring rank and consistency by orthogonal projection*, Linear Algebra and its Applications, 28 (1979), pp. 257–278.

[127] H. A. Van Der Vorst and Q. Ye, *Residual replacement strategies for Krylov subspace iterative methods for the convergence of true residuals*, SIAM Journal on Scientific Computing, 22 (2000), pp. 835–852.

[128] S. Van Huffel, *Iterative algorithms for computing the singular subspace of a matrix associated with its smallest singular values*, Linear Algebra and its Applications, 154 (1991), pp. 675–709.

[129] S. Van Huffel and J. Vandewalle, *The Total Least Squares Problem: Computational Aspects and Analysis*, SIAM, 1991.

[130] S. Van Huffel and H. Zha, *An efficient total least squares algorithm based on a rank-revealing two-sided orthogonal decomposition*, Numerical Algorithms, 4 (1993), pp. 101–133.

[131] B. Vieublé, *Mixed precision iterative refinement for the solution of large sparse linear systems*, PhD thesis, INP Toulouse, 2022.

[132] J. H. Wilkinson, *Progress report on the automatic computing engine*, Tech. Rep. MA/17/1024, Mathematics Division, Department of Scientific and Industrial Research, National Physical Laboratory, Teddington, UK, 1948.

[133] J. H. Wilkinson, *Rounding errors in algebraic processes*, Notes Appl. Sci., 32 (1963). Her Majesty's Stationery Office, London; also published by Prentice-Hall, Englewood Cliffs, NJ, reprinted by Dover, NewYork, 1994.

[134] I. Yamazaki, S. Tomov, T. Dong, and J. Dongarra, *Mixed-precision orthogonalization scheme and adaptive step size for improving the stability and performance of CA-GMRES on GPUs*, in International Conference on High Performance Computing for Computational Science, Springer, 2015, pp. 17–30.

[135] I. Yamazaki, S. Tomov, and J. Dongarra, *Mixed-precision Cholesky QR factorization and its case studies on multicore CPU with multiple GPUs*, SIAM Journal on Scientific Computing, 37 (2015), pp. C307–C330.

[136] ——, *Stability and performance of various singular value QR implementations on multicore CPU with a GPU*, ACM Transactions on Mathematical Software, 43 (2016), pp. 1–18.

[137] I. Yamazaki, S. Tomov, J. Kurzak, J. Dongarra, and J. Barlow, *Mixed-precision block Gram Schmidt orthogonalization*, in Proceedings of the 6th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems, 2015, pp. 1–8.

[138] L. M. Yang, A. Fox, and G. Sanders, *Rounding error analysis of mixed precision block Householder QR algorithms*, SIAM Journal on Scientific Computing, 43 (2021), pp. A1723–A1753.