PROBABILISTIC ROUNDING ERROR ANALYSIS FOR NUMERICAL LINEAR ALGEBRA

A thesis submitted to The University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Science and Engineering

2022

Michael P. Connolly
Department of Mathematics

CONTENTS

WORD COUNT: 41 487

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This thesis presents new results on probabilistic error analysis. By freeing ourselves of the burden of having to account for every possible worst-case scenario, we provide error bounds that are more informative than the long standing worst-case bounds. We also see in practice how these new bounds can guide the development of new algorithms.

We begin by studying stochastic rounding, a rounding mode which rounds a computed result to the next larger or smaller floating-point number randomly. We compare stochastic rounding with round-to-nearest, finding some similarities but also a large number of properties that hold for round-to-nearest but fail to hold for stochastic rounding. Importantly, we show that the rounding errors produced by stochastic rounding are mean zero, mean independent random variables. Using this fact, we build on earlier probabilistic error analysis to show that for a wide range of inner product based linear algebra computations, stochastic rounding provides backward error bounds where we can take the square root of the dimensional constants in the worst-case bounds. This has been a well known rule of thumb for some time, but we show it to be a rule for stochastic rounding.

Expanding on this work for inner-product based algorithms, we investigate orthogonal transformations. Using the same model of rounding errors as that which stochastic rounding satisfies, we show that for Householder QR factorization, and other related algorithms, we see the same square rooting of the dimensional constant in the backward error bound. The analysis makes use of a matrix concentration inequality, where previous probabilistic error analyses employed scalar concentration inequalities. We also derive a new backward error formula for QR factorization, used in our numerical experiments which validate our bounds.

Finally, we consider the randomized SVD, a well known method for computing low rank matrix approximations. We perform a complete rounding error analysis of the fixed-rank problem, the most straightforward version of the randomized SVD where we have a priori specified a target rank. This is complementary to the existing literature providing error bounds on this procedure in exact arithmetic. While our initial analysis is worst-case, we use our previous probabilistic results to refine these error bounds. Using these refined bounds, we propose a mixed-precision version of the algorithm that offers potential speedups by gradually reducing the precision during the execution of the algorithm.

DECLARATION

No portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

COPYRIGHT STATEMENT

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the "Copyright") and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made **only** in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trade marks and other intellectual property (the "Intellectual Property") and any reproductions of copyright works in the thesis, for example graphs and tables ("Reproductions"), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=24420), in any relevant Thesis restriction declarations deposited in the University Library, The University Library's regulations (see http://www.library.manchester.ac.uk/about/regulations) and in The University's Policy on Presentation of Theses.

PUBLICATIONS

- Chapter 3 is based on the journal article: Michael P. Connolly, Nicholas J. Higham and Theo Mary. Stochastic Rounding and Its Probabilistic Backward Error Analysis. *SIAM J. Sci. Comput.* 43.1 (2021), A566–A585.

- Chapter 4 is based on the preprint: Michael P. Connolly and Nicholas J. Higham. Probabilistic Rounding Error Analysis of Householder QR Factorization. MIMS EPrint 2022.5. UK: Manchester Institute for Mathematical Sciences, The University of Manchester, Feb. 2022; revised Aug. 2022, pp. 18. Submitted to *SIAM J. Matrix Anal. Appl.*

- Chapter 5 is based on the preprint: Michael P. Connolly, Nicholas J. Higham and Srikara Pranesh. Randomized Low Rank Matrix Approximation: Rounding Error Analysis and a Mixed Precision Algorithm. MIMS EPrint 2022.10. UK: Manchester Institute for Mathematical Sciences, The University of Manchester, Jan. 2022, pp. 18. Submitted to *SIAM J. Sci. Comput.*

1 INTRODUCTION

In the early days of scientific and digital computing, the landscape of floating-point arithmetic was rather turbulent. A seminal moment was the publication of the IEEE standard 754 [17], which defines a particular floating-point arithmetic. The aims of this standard were to encourage robust and portable codes. Reasons for its development were many [20], but the basic idea is simple: if the floating-point arithmetics on two different machines both adhere to the standard, then when we move a program from one machine to the other, the results will be identical! There are further complicating factors which mean that's not always strictly true, but the spirit of the idea is clear. Pre IEEE-754, most computers offered both a single and a double precision floating-point format, but the formats varied wildly across different machines. Some example formats are shown in Table 1.0.1. In Table 1.0.1: β refers to the base of the floating-point arithmetic; t is the precision; $e_{\min}$ and $e_{\max}$ are the minimum and maximum exponents; and u is the unit roundoff. Each of these quantities are defined in Section 2.2. Each different format brings its own idiosyncrasies, and from [20] " those idiosyncrasies and the programming contortions they induce imposes a numbing intellectual burden upon the software industry ". Since the IEEE standard, scientific computing has traditionally been dominated by one floating-point format: IEEE double precision, comprising 64 bits and sometimes called binary64 or fp64.

Table 1.0.1: Old floating-point formats [12, Tab. 2.1].

Arithmetic	β	t	$e_{\min}$	$e_{\max}$	u
Cray-1 single	2	48	-8192	8191	4×10^{-15}
Cray-1 double	2	96	-8192	8191	1×10^{-29}
IBM 3090 single	16	6	-64	63	5×10^{-7}
IBM 3090 double	16	14	-64	63	1×10^{-16}

The IEEE standard brought great stability, but in recent years this stability has once again been undermined. Hardware vendors have begun to provide support for a wide variety of formats, including 16 and 8 bit formats. The adoption of these low-precision formats has been primarily application driven. If 32 or 64 bit floating-point arithmetic can be replaced by low precision, then there is great potential for computational speedup and reduced memory and energy footprints. We detail some notable successes of using low precision arithmetics in modern scientific computing. There are many examples of this over the past decade or so, across a wide range of application areas.

One of the most prominent is machine learning. Two prominent papers advocating the use of low precision in neural networks are [4] and [11]. It has been acknowledged that machine learning algorithms are tolerant of large errors and so amenable to the use of low precision. It has also been suggested that the use of low precision can act as a form of regularization in training models.

The Ising model is a well known thermodynamic system, and often serves as a prototype system in statistical physics. It has become particularly well known as it has been a testing ground of sorts for computational algorithms. In [27], it was shown that fp32 can be replaced by bfloat16 while simulating the Ising model, without any loss of accuracy. Fp32 is a long standing 32 bit format, while bfloat16 is a relatively new format. The exact specifications of both are given in Chapter 2.

A large amount of work has been done in studying the effect of precision on weather forecasting codes. The reduced cost of low precision arithmetic, both in execution and memory footprint, means that its successful adoption can lead to more refined spatial grids, and thus more accurate prediction. It has been argued that the use of double precision is redundant in these models, as observations upon which the models are built are low precision and there are large inherent uncertainties at play [6], [24]. Computational gains in this area have been demonstrated in [22], [25].

Rather than simply replacing all high precision computations with low precision computations, what has become increasingly common is what are termed *mixed-precision algorithms*. Loosely speaking, we aim to use two or more precisions and hope to exploit the speedup and reduced energy and storage costs offered by low precisions, without sacrificing on the accuracy offered by higher precisions.

The idea of iterative refinement for solving a linear system $Ax = b$ dates back to the 1940's when it was programmed by Wilkinson. The basic concept is in the name: we start with an initial guess for a solution and then refine it until we have reached some acceptable accuracy. Recent works have shown great benefit in mixed-precision variants of these algorithms. [21] proposed a two precision version. [3] then proposed a three precision version along with convergence analysis, and [2] extended this idea to five precisions. Extensive surveys of mixed-precision in numerical linear algebra are [1], [14].

The interest of the computer hardware community and manufacturers in low precision computation and stochastic rounding, an important aspect of this thesis to be discussed in Chapter 3, has also increased in recent years. References include [5], [7], [9], [10], [11], [16], [18] and [23],

Adjacent to the rise of these low precisions has been the ever increasing capabilities of the largest supercomputers. The Frontier system at Oak Ridge National Laboratory is, at present, the most powerful supercomputer to ever exist. It achieved a score of 1.102 Exaflop/s on the June 2022 HPL benchmark[1] [8], the metric used to rank machines on the Top500 list[2]. This benchmark involves solving a linear system of equations. The size of the system solved by Frontier for which it achieved this flop rate was a massive 2.4×10^7.

A common quantity in error bounds is nu, where n is our problem size and u is the unit roundoff of our floating-point arithmetic. These two trends of ever increasing

1 https://www.top500.org/project/linpack/
2 https://top500.org/

problem sizes and much more variety in the floating-point landscape mean that the product nu can often exceed 1, which in the worst case scenario will render an error bound essentially meaningless. The central aim of this thesis is to employ *probabilistic* rounding error analysis to provide bounds that are more informative and valid for a wider range of problem sizes and unit roundoffs.

For many years, it has been a rule of thumb that one can replace a worst-case rounding error bound of the form $f(n)u$ by something of the form $\sqrt{f(n)}u$. This is attributable to Wilkinson. In [26, p. 318], he derives rounding error bounds for Gaussian elimination, Givens QR factorization and Householder QR factorization, and then states

> "In general, the statistical distribution of the rounding errors will reduce considerably the function of n occurring in the relative errors. We might expect in each case that this function should be replaced by something which is no bigger than its square root and is usually appreciably smaller."

Despite this rule of thumb, there was no proof that made it rigorous. Recent work by Higham and Mary [13], [15], and Ipsen and Zhou [19] progressed the field of probabilistic error analysis. These works modelled rounding errors as mean-zero, independent random variables. In this thesis, we identify a different model of rounding errors which relaxes the requirement of independence to *mean independence*. We identify that a particular mode of stochastic rounding satisfies this model, and use the model to derive probabilistic rounding error bounds. This is done for inner-product based computations in Chapter 3 and for orthogonal transformation based computations in Chapter 4. The results in these chapters show that, under the assumptions of this model of rounding errors, we see the reduction $f(n)u$ to $\sqrt{f(n)}u$ as predicted by Wilkinson, and the rule of thumb is now simply a *rule*. In Chapter 5, we provide a rounding error analysis of a randomized low-rank matrix approximation algorithm, the operations of which have probabilistic rounding error analyses thanks to our results in the previous two chapters. Using our probabilistic rounding error analysis, we are able to justify mixed-precision algorithms which offer potential speedup

without sacrificing on accuracy. Chapter 5 demonstrates that the results derived in Chapters 3 and 4 can be used to not only provide new error bounds, but also guide the development of new algorithms.

Chapter 2 summarizes the main definitions and properties that are used throughout the thesis but are not given in the introductory sections of the following chapters. Since the thesis is in journal format, a few background topics are discussed in more than one place. Some of these topics are discussed in more detail in Chapter 2 than in the following chapters, as we feel the expository treatment given to them will be beneficial to the overall readability of the thesis.

Chapters 3, 4 and 5 are presented in a format suitable for publication and are based on the preprints and journal papers listed on page 10. For coauthored papers, the authors contributed equally to the final manuscript.

REFERENCES

[1] A. Abdelfattah, H. Anzt, E. G. Boman, E. Carson, T. Cojean, J. Dongarra, A. Fox, M. Gates, N. J. Higham, X. S. Li, J. Loe, P. Luszczek, S. Pranesh, S. Rajamanickam, T. Ribizel, B. F. Smith, K. Swirydowicz, S. Thomas, S. Tomov, Y. M. Tsai, and U. M. Yang. "A survey of numerical linear algebra methods utilizing mixed-precision arithmetic." *Int. J. High Performance Computing Applications* 35.4 (Mar. 2021), pp. 344–369 (cited on p. 12).

[2] 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. UK: Manchester Institute for Mathematical Sciences, The University of Manchester, Apr. 2021, p. 21. Revised April 2022 (cited on p. 12).

[3] E. Carson and N. J. Higham. "Accelerating the solution of linear systems by iterative refinement in three precisions." *SIAM J. Sci. Comput.* 40.2 (2018), A817–A847 (cited on p. 12).

[4] M. Courbariaux, Y. Bengio, and J.-P. David. "BinaryConnect: Training Deep Neural Networks with Binary Weights During Propagations." In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. Curran Associates, Inc., 2015, pp. 3123–3131 (cited on p. 11).

[5] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C.-K. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y.-H. Weng, A. Wild, Y. Yang, and H. Wang. "Loihi: a neuromorphic manycore processor with on-chip learning." *IEEE Micro* 38.1 (2018), pp. 82–99 (cited on p. 12).

[6] A. Dawson, P. D. Düben, D. A. MacLeod, and T. N. Palmer. "Reliable low precision simulations in land surface models." *Climate Dynamics* 51.7 (Oct. 2018), pp. 2657–2666 (cited on p. 11).

[7] *Deep learning performance documentation*. https://docs.nvidia.com/deeplearning/performance/mixed-precision-training/index.html. Accessed: 2022-12-11 (cited on p. 12).

[8] J. J. Dongarra, P. Luszczek, and A. Petitet. "The LINPACK Benchmark: past, present and future." *Concurrency and Computation: Practice and Experience* 15.9 (2003), pp. 803–820. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.728 (cited on p. 12).

[9] M. Fasi and M. Mikaitis. "Algorithms for stochastically rounded elementary arithmetic operations in ieee 754 floating-point arithmetic." *IEEE Transactions on Emerging Topics in Computing* 9.3 (2021), pp. 1451–1466 (cited on p. 12).

[10] *Graphcore: Mixed-Precision Arithmetic for AI: A Hardware Perspective*. https://docs.graphcore.ai/projects/ai-float-white-paper/en/latest/ai-float.html. Accessed: 2022-12-11 (cited on p. 12).

[11] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan. "Deep Learning with Limited Numerical Precision." In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France: JMLR.org, 2015, pp. 1737–1746 (cited on pp. 11, 12).

[12] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. Second. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2002, pp. xxx+680 (cited on p. 10).

[13] N. J. Higham and T. Mary. "A new approach to probabilistic rounding error analysis." *SIAM J. Sci. Comput.* 41.5 (2019), A2815–A2835 (cited on p. 13).

[14] N. J. Higham and T. Mary. "Mixed precision algorithms in numerical linear algebra." *Acta Numerica* 31 (May 2022), pp. 347–414 (cited on p. 12).

[15] N. J. Higham and T. Mary. "Sharper probabilistic backward error analysis for basic linear algebra kernels with random data." *SIAM J. Sci. Comput.* 42.5 (2020), A3427–A3446 (cited on p. 13).

[16] M. Hopkins, M. Mikaitis, D. R. Lester, and S. Furber. "Stochastic rounding and reduced-precision fixed-point arithmetic for solving neural ordinary differential equations." *Phil. Trans. R. Soc. A* 378.2166 (2020), pp. 1–22 (cited on p. 12).

[17] "IEEE Standard for Binary Floating-Point Arithmetic". *ANSI/IEEE Standard 754-1985* (Oct. 1985) (cited on p. 10).

[18] *Intel Advanced Vector Extensions 512 - FP16 Instruction Set for Intel Xeon Processor Based Products*. https://builders.intel.com/docs/networkbuilders/intel-avx-512-fp16-instruction-set-for-intel-xeon-processor-based-products-technology-guide-1651874188.pdf. Accessed: 2022-12-11 (cited on p. 12).

[19] I. C. F. Ipsen and H. Zhou. "Probabilistic error analysis for inner products." *SIAM J. Matrix Anal. Appl.* 41.4 (Jan. 2020), pp. 1726–1741 (cited on p. 13).

[20] W. Kahan. *Why do we need a floating-point arithmetic standard?* https://people.eecs.berkeley.edu/~wkahan/ieee754status/why-ieee.pdf. 1981. Accessed: 2022-08-18 (cited on p. 10).

[21] 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: New York, NY, USA: Association for Computing Machinery, 2006 (cited on p. 12).

[22] C. Maynard and D. Walters. "Mixed-precision arithmetic in the ENDGame dynamical core of the Unified Model, a numerical weather prediction and climate model code." *Computer Physics Communications* 244 (2019), pp. 69–75 (cited on p. 11).

[23] M. Mikaitis. "Stochastic Rounding: Algorithms and Hardware Accelerator." In: *2021 International Joint Conference on Neural Networks (IJCNN)*. 2021, pp. 1–6 (cited on p. 12).

[24] O. Tintó Prims, M. C. Acosta, A. M. Moore, M. Castrillo, K. Serradell, A. Cortés, and F. J. Doblas-Reyes. "How to use mixed precision in ocean models: exploring a potential reduction of numerical precision in NEMO 4.0 and ROMS 3.6." *Geoscientific Model Development* 12.7 (2019), pp. 3135–3148 (cited on p. 11).

[25] F. Váňa, P. Düben, S. Lang, T. Palmer, M. Leutbecher, D. Salmond, and G. Carver. "Single precision in weather forecasting models: an evaluation with the IFS." *Monthly Weather Review* 145.2 (2017), pp. 495–502 (cited on p. 11).

[26] J. H. Wilkinson. "Error analysis of direct methods of matrix inversion." *J. ACM* 8.3 (July 1961), pp. 281–330 (cited on p. 13).

[27] K. Yang, Y.-F. Chen, G. Roumpos, C. Colby, and J. Anderson. "High Performance Monte Carlo Simulation of Ising Model on TPU Clusters." In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. SC '19. Denver, Colorado: Association for Computing Machinery, 2019 (cited on p. 11).

2 BACKGROUND MATERIAL

2.1 LINEAR ALGEBRA

Primary sources for this section are [5] and [6]. We assume that the reader is familiar with some basic ideas in linear algebra such as vector spaces, subspaces, linear independence, span, basis, dimension, and determinants.

Matrices. A *matrix* is a rectangular array of scalars. More precisely, a matrix over a field $\mathbb{F}$ is an array where the individual entries are elements of $\mathbb{F}$. If the matrix A has m rows and n columns, and so mn entries, we say $A \in \mathbb{F}^{m \times n}$. Throughout this thesis, $\mathbb{F}$ is either $\mathbb{R}$ or $\mathbb{C}$. We denote matrices with uppercase Latin or Greek letters, and individual entries with the corresponding lower case letter subscripted with its row and column indices. For example, for the matrix A, the element in the i-th row and j-th column is given by a_{ij}. When convenient, we will use MATLAB style notation to index matrices. This means we refer to a_{ij} as $A(i, j)$, and we can use colons to index entire rows or columns, so $A(i, :)$ is the i-th row of A and $A(:, j)$ is the j-th column of A. Matrices in $\mathbb{F}^{1 \times m}$ and $\mathbb{F}^{m \times 1}$ are called *row* and *column* vectors, respectively. In the case of column vectors we drop the dimension of 1 and simply write $\mathbb{F}^m$. We typically denote vectors with lower-case Latin or Greek letters.

Matrices with the same dimensions can be summed, and the sums are performed elementwise. If $A, B \in \mathbb{F}^{m \times n}$ then $(A + B) \in \mathbb{F}^{m \times n}$ and $(A + B)_{ij} = a_{ij} + b_{ij}$. Products are defined for matrices with appropriate dimensions. An inner product is a product

of a row and a column vector of the same dimension and the result is a scalar. For two vectors $a \in F^{1\times m}$ and $b \in \mathbb{F}^m$ the inner product $c = ab$ is given by

$$c = \sum_{i=1}^{m} a_i b_i.$$

For matrices $A \in \mathbb{F}^{m\times p}$ and $B \in \mathbb{F}^{q\times n}$, the product $C = AB$ is defined if $p = q$, with $C \in \mathbb{F}^{m\times n}$. The elements of C are given by

$$c_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj},$$

meaning the (i, j) element of C is given by the inner product of the i-th row of A with the j-th column of B.

The *range* and *nullspace* are two important subspaces associated with each matrix $A \in \mathbb{F}^{m\times n}$. The range of A is given by

$$\text{ran}(A) = \{y \in \mathbb{F}^m : y = Ax \text{ for some } x \in \mathbb{F}^n\},$$

and the nullspace by

$$\text{null}(A) = \{x \in \mathbb{F}^n : Ax = 0\}.$$

The *rank* of a matrix is the dimension of the range of A:

$$\text{rank}(A) = \dim(\text{ran}(A)).$$

We also have $\dim(\text{null}(A)) + \text{rank}(A) = n$. If $\text{rank}(A) < \min\{m, n\}$ we say that A is *rank-deficient*.

The multiplicative inverse of $A \in \mathbb{F}^{n\times n}$ is written as A^{-1} and we have $AA^{-1} = A^{-1}A = I_n$. The identity matrix I_n is a matrix with elements $i_{ij} = \delta_{ij}$, where δ_{ij} is the Kroenecker delta. The *transpose* of $A \in \mathbb{F}^{m\times n}$ is $A^T \in \mathbb{F}^{n\times m}$ and is the matrix where the element in position (i, j) is the (j, i) element of A. The *conjugate transpose* of A

is given by A^*, where as well as the transposition above we also take the complex conjugate of each element.

Special matrices. Some matrices are endowed with properties that make them important, either from the perspective of computation or error analysis. The main diagonal of a matrix is the set of entries that have the same row and column index. A *diagonal matrix* is one which has zero entries everywhere off the main diagonal, while the entries on the main diagonal may be zero or non-zero. Diagonal matrices may be rectangular, but typically refer to square matrices. The *identity matrix* $I_n \in \mathbb{R}^{n\times n}$ is the diagonal matrix with all ones on the main diagonal. Sometimes the subscript n is dropped if the dimension is obvious from context.

A matrix $T \in \mathbb{F}^{n\times n}$ is *upper triangular* if $t_{ij} = 0$ for $i > j$, and *lower triangular* if $t_{ij} = 0$ for $i < j$. These are important matrices as it is both fast and numerically stable to solve equations of the form $Tx = b$ with $x, b \in \mathbb{F}^n$. If $T \in \mathbb{F}^{m\times n}$ is rectangular, we instead use the terms *upper trapezoidal* or *lower trapezoidal*, when the same conditions on the individual elements of T hold. An important "close to triangular" matrix is a *Hessenberg matrix*. A matrix is upper Hessenberg if all entries below the first subdiagonal are zero, and lower Hessenberg if all entries above the first superdiagonal are zero. More precisely, $A \in \mathbb{F}^{n\times n}$ is upper Hessenberg if $a_{ij} = 0$ for $i < j + 1$ and lower Hessenberg if $a_{ij} = 0$ for $j > i + 1$.

A matrix $A \in \mathbb{R}^{n\times n}$ is *symmetric* if $A^T = A$ and *skew symmetric* if $A^T = -A$. A matrix $Q \in \mathbb{R}^{n\times n}$ is *orthogonal* if $QQ^T = Q^TQ = I_n$, or equivalently $Q^T = Q^{-1}$. For the complex case, we say a matrix $A \in \mathbb{C}^{n\times n}$ is *Hermitian* if $A^* = A$ and *skew-Hermitian* if $A^* = -A$. A matrix $U \in \mathbb{C}^{n\times n}$ is *unitary* if $UU^* = U^*U = I_n$, or equivalently $U^* = U^{-1}$.

A matrix $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite if $x^TAx > 0$ for all non-zero x, and symmetric positive semi-definite if $x^TAx \geqslant 0$ for all non-zero x. An equivalent condition for a matrix to be symmetric positive definite or semi-definite, is for the

eigenvalues to all be positive or non-negative respectively. Obvious analogues hold for the Hermitian case.

A matrix $A \in \mathbb{R}^{m \times n}$ is *low-rank* if $\text{rank}(A) = k \ll \min(m, n)$ and a matrix of rank k can be written $A = XY^T$ with $X \in \mathbb{R}^{m \times k}$ and $Y \in \mathbb{R}^{n \times k}$. Identifying and exploiting matrices that are low-rank can lead to large computational benefits.

A matrix is *sparse* if it contains enough non-zeros that it is worth exploiting them. An obvious and simple sparse matrix is a diagonal matrix. Algorithms and data structures that are tailored to sparse matrices can lead to large computational gains.

Eigenvalues and eigenvectors. For $A \in \mathbb{C}^{n \times n}$, $x \in \mathbb{C}^n$ and nonzero, and $\lambda \in \mathbb{C}$, if $Ax = \lambda x$ then we have λ is an eigenvalue of A, x is an eigenvector and (λ, x) is an eigenpair. Rearranging the above equation to $(A - \lambda I)x = 0$, we can see that λ is an eigenvalue if and only if $(A - \lambda I)$ is singular. This is equivalently written as $\det(A - \lambda I) = 0$, so the eigenvalues of A are given by the roots of this characteristic polynomial $\det(A - \lambda I)$. The spectral radius $\rho(A)$ is given by

$$\rho(A) = \max\{|\lambda| : \det(A - \lambda I) = 0\}. \tag{2.1.1}$$

Vector and matrix norms. Norms are a powerful tool in numerical linear algebra and error analysis. They allow for an $m \times n$ matrix to be compressed into a single scalar, which provides conciseness and interpretability to various results. A norm is a function $\|\cdot\| : \mathbb{C}^{m \times n} \to \mathbb{R}$ if, for any $A, B \in \mathbb{C}^{m \times n}$ and $\alpha \in \mathbb{C}$, the following conditions are satisfied:

1. $\|A\| \geqslant 0$, with equality if and only if $A = 0$.

2. $\|\alpha A\| = |\alpha| \|A\|$.

3. $\|A + B\| \leqslant \|A\| + \|B\|$.

We call this function a *vector norm* if the input is a vector ($n = 1$). Three common vector norms, defined below for $x \in \mathbb{C}^m$, are

$$\|x\|_1 = \sum_{i=1}^{m} |x_i|,$$
$$\|x\|_2 = (x^* x)^{1/2},$$
$$\|x\|_\infty = \max_{1 \leqslant i \leqslant m} |x_i|.$$

These are all special cases of the Hölder p-norm

$$\|x\|_p = \left(\sum_{i=1}^{m} |x_i|^p \right)^{1/p}, \quad p \geqslant 1.$$

The most immediate, and simplest, matrix norm is the Frobenius norm, given by

$$\|A\|_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{1/2} = \operatorname{trace}(A^* A)^{1/2},$$

with $\operatorname{trace}(B) = \sum_{i=1}^{n} b_{ii}$ for $B \in \mathbb{C}^{n \times n}$. The *subordinate* matrix norm on $\mathbb{C}^{m \times n}$, given vector norm $\|\cdot\|$, is given by

$$\|A\| = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|}.$$

For the $p = 1, 2,$ and ∞ vector norms, it can be shown that the corresponding subordinate matrix norms are given by

$$\|A\|_\infty = \max_{1 \leqslant i \leqslant m} \sum_{j=1}^{n} |a_{ij}|,$$
$$\|A\|_1 = \max_{1 \leqslant j \leqslant n} \sum_{i=1}^{m} |a_{ij}|,$$
$$\|A\|_2 = (\rho(A^* A))^{1/2} = \sigma_{\max}(A),$$

where $\rho(B)$ is the spectral radius (2.1.1) and $\sigma_{\max}(A)$ is the largest singular value. We can also express $\|A\|_F = \left(\sum_{i=1}^{n} \sigma_i^2 \right)^{1/2}$, with σ_i the i-th singular value (see (2.1.2)).

A norm is *consistent* if $\|AB\| \leqslant \|A\|\|B\|$. The Frobenius norm and all subordinate norms are consistent. This is a convenient property for error analysis and manipulating expressions containing norms.

A *unitarily invariant* norm is one for which $\|UAV\| = \|A\|$ for all unitary U and V. This is a useful property with implications for error analysis, as it means that multiplication of a matrix contaminated with errors by a unitary matrix does not magnify those errors when measured in a unitarily invariant norm. The 2-norm and the Frobenius norm are both unitarily invariant norms.

It is often important to be able to switch between different norms. Of most interest to us are the relations between the 2-norm and the Frobenius norm, where we have $\|A\|_2 \leqslant \|A\|_F$ and $\|A\|_F \leqslant \sqrt{\operatorname{rank}(A)}\|A\|_2 \leqslant \sqrt{n}\|A\|_2$.

Used often throughout is the notion of componentwise absolute value of a matrix or vector. Given matrix A and vector x, the matrix $|A|$ and vector $|x|$ have elements $|a_{ij}|$ and $|x_i|$ respectively.

Triangular systems. A crucial computational routine is that of solving a triangular linear system. The method for solving $Lx = b$ for lower triangular $L \in \mathbb{R}^{n\times n}$, called forward substitution, is summarized below:

$$
\begin{aligned}
x_1 &= b_1/l_{11}, \\
x_i &= \left(b_i - \sum_{j=1}^{i-1} l_{ij}x_j \right) / l_{ii}.
\end{aligned}
$$

A similar method called back substitution exists for solving $Ux = b$ for upper triangular U:

$$
\begin{aligned}
x_n &= b_n/u_{nn}, \\
x_i &= \left(b_i - \sum_{j=i+1}^{n} u_{ij}x_j \right) / u_{ii}.
\end{aligned}
$$

These methods are numerically stable, having backward error (see Section 2.2) as small as could be hoped for [6, Thm. 8.5]. They form the basis for many computational methods where the goal is to reduce the problem to solving one or more triangular systems.

LU factorization. The most common method for solving a general dense square system of linear equations $Ax = b$ is via LU factorization. This is composed of two stages: perform the decomposition $A = LU$ where L is *unit* lower triangular and U is upper triangular; and then perform successive triangular solves $Ly = b$ for y and $Ux = y$ for x. We have the following existence result for LU factorization [6, Thm. 9.2].

Theorem 2.1. *There exists a unique LU factorization of $A \in \mathbb{R}^{n\times n}$ if and only if $A(1:k, 1:k)$ is nonsingular for $k = 1:n-1$. If $A(1:k, 1:k)$ is singular for some $1 \leqslant k \leqslant n-1$ then the factorization may exist but it is not unique.*

In performing LU factorization, for the sake of numerical stability, one can implement what is called a pivoting strategy. The most common strategy is partial pivoting, where at each stage of the decomposition we perform a row interchange. This helps to avoid divisions by zero and the addition or subtraction of large numbers which could lead to a loss of significance. Mathematically this means that we compute the LU decomposition not of A, but the matrix PA where P is a permutation matrix that permutes the rows of A.

From the perspective of error analysis, an important method for computing the LU factorization is Doolittle's method. All methods of computing an LU factorization that are mathematically equivalent will satisfy a common error bound, and since Doolittle's method is composed of simply inner products and substitutions, its analysis is rather straightforward. There is also no need to analyse the method with pivoting, as it is equivalent to the method without pivoting applied to the permuted matrix. Standard results show that LU factorization and solution of linear systems

Algorithm 2.1 Doolittle's method for computing the LU factorization of $A \in \mathbb{R}^{n\times n}$.

for $k = 1 : n$ **do**
  for $j = k : n$ **do**
    $u_{kj} = a_{kj} - \sum_{i=1}^{k-1} l_{ki}u_{ij}$
  end for
  for $i = k+1 : m$ **do**
    $l_{ik} = (a_{ik} - \sum_{j=1}^{k-1} l_{ij}u_{jk})/u_{kk}$
  end for
end for

by the method are, save for some highly unlikely worst-case scenarios, numerically stable.

Cholesky factorization. A method similar to LU factorization exists for symmetric positive definite systems. If $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite then there is a unique upper triangular $R \in \mathbb{R}^{n\times n}$ with positive diagonal elements such that $A = R^T R$. The textbook algorithm for computing a Cholesky decomposition is shown.

Algorithm 2.2 Computing a Cholesky factorization $A = R^T R$ for symmetric positive definite $A \in \mathbb{R}^{n\times n}$.

for $j = 1 : n$ **do**
  for $i = 1 : (j-1)$ **do**
    $r_{ij} = (a_{ij} - \sum_{k=1}^{i-1} r_{ki}r_{kj})/r_{ii}$
  end for
  $r_{jj} = (a_{jj} - \sum_{k=1}^{j-1} r_{kj}^2)^{1/2}$
end for

The computational cost is half that of LU factorization, and it is also a numerically stable routine. Again, having computed $A = R^T R$ we can solve the system $Ax = b$ by solving $R^T y = b$ and then $Rx = y$.

QR factorization. A QR factorization of $A \in \mathbb{R}^{m\times n}$ with $m \geqslant n$ is a factorization $A = QR$ where $Q \in \mathbb{R}^{m\times m}$ and R is upper trapezoidal. We can also consider the economy-size decomposition $A = Q_1 R_1$ where $Q_1 \in \mathbb{R}^{m\times n}$ and R_1 is upper triangular. The QR factorization has uses in solving linear systems and least squares and eigenvalue problems. In Chapter 4, we consider the numerical properties of algo-

rithms used to compute the QR factorization. Here we provide some detail on those algorithms.

A Householder matrix is a matrix of the form

$$P = I - \frac{2}{v^T v} v v^T, \quad 0 \neq v \in \mathbb{R}^n.$$

The matrix P is symmetric, orthogonal and involutory ($P^2 = I$). The power in Householder matrices lies in their ability to introduce zeros into a vector. Given two nonequal vectors x and y such that $\|x\|_2 = \|y\|_2$, we can always find a Householder matrix P such that $Px = y$. A QR factorization can then be computed by premultiplying the matrix A by a sequence of Householder matrices, so that we produce the upper trapezoidal matrix R. To begin, we construct a Householder matrix P_1 that when applied to the first column of A leaves all but the first entry as zero. This process is repeated for subsequent columns, where we leave the second column with two non-zeros by applying P_2, and so on until we have computed $R = P_n \dots P_2 P_1 A$ with $Q = (P_n \dots P_2 P_1)^T = P_1 P_2 \dots P_n$. If we require the matrix Q explicitly, we can form it by accumulating this product of Householder matrices, but it is often advantageous from a computational perspective to keep the matrix in its factored form.

Givens rotations operate by a similar idea of introducing zeroes to a vector, but do so one at a time. A Givens rotation $G(i, j, \theta)$ is equal to the identity except

$$G([i, j], [i, j]) = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}.$$

In the multiplication $y = G(i, j, \theta)x$, the individual components of y are given by

$$y_k = \begin{cases} x_k & k \neq i, j, \\ \cos(\theta)x_i + \sin(\theta)x_j, & k = i, \\ -\sin(\theta)x_i + \cos(\theta)x_j, & k = j. \end{cases}$$

We can set $y_j = 0$ if we choose

$$\sin(\theta) = \frac{x_j}{\sqrt{x_i^2 + x_j^2}}, \quad \cos(\theta) = \frac{x_i}{\sqrt{x_i^2 + x_j^2}}.$$

We can then proceed in a similar fashion to a Householder QR factorization, introducing zeros to the required locations in order to bring A to an upper trapezoidal form. For a general, dense $m \times n$ matrix, a Givens QR factorization is more expensive than a Householder QR factorization. Its utility lies in computing the QR factorization of specially structured matrices, such as tridiagonal or Hessenberg matrices. Householder matrices are good at introducing a large amount of zeros in one go to a matrix, while Givens matrices are the chosen method when we must be more selective with where we want to zero.

Singular value decomposition. Given $A \in \mathbb{C}^{m\times n}$, the singular value decomposition (SVD) of A computes orthogonal matrices $U \in \mathbb{C}^{m\times m}$, $V \in \mathbb{C}^{n\times n}$ and a diagonal matrix $\Sigma \in \mathbb{C}^{m\times n}$ such that

$$A = U\Sigma V^*. \tag{2.1.2}$$

As for the QR decomposition, we can obtain economy size versions of the SVD. For $m > n$, we have

$$\begin{aligned} U &= \begin{bmatrix} U_1 \; U_2 \end{bmatrix}, \quad U_1 \in \mathbb{C}^{m\times n}, U_2 \in \mathbb{C}^{m\times(m-n)}, \\ \Sigma &= \begin{bmatrix} \Sigma_1 \\ 0 \end{bmatrix}, \quad \Sigma_1 \in C^{n\times n}, \\ V &= V_1. \end{aligned}$$

For $m < n$, we have

$$\begin{aligned} U &= U_1, \\ \Sigma &= \begin{bmatrix} \Sigma_1 \; 0 \end{bmatrix}, \quad \Sigma_1 \in \mathbb{C}^{m\times m}, \\ V &= \begin{bmatrix} V_1 \; V_2 \end{bmatrix}, \quad V_1 \in \mathbb{C}^{n\times m}, V_2 \in \mathbb{C}^{n\times(n-m)}. \end{aligned}$$

In either case, $A = U\Sigma V^* = U_1\Sigma_1 V_1^*$.

The entries σ_i, $i = 1 : n$ are in order of decreasing magnitude, and are called the singular values of A. The singular values of A are the positive square roots of the eigenvalues of the matrix AA^*.

Often, one truncates an SVD even further than is done in the economy size versions. We denote this truncated SVD as $A_k = U_k\Sigma_k V_k^*$, where $U_k \in \mathbb{C}^{m\times k}$ contains the first k columns of U, $\Sigma_k \in \mathbb{C}^{k\times k}$ is diagonal containing only the first k singular values and $V_k \in \mathbb{C}^{n\times k}$, containing only the first k columns of V.

The SVD is an immensely important matrix. Some examples, amongst many, of its use are computing pseudoinverse [4]; computing the numerical rank and 2-norm of a matrix; and performing principal component analysis [1], a popular method of dimensionality reduction. The SVD also provides optimal solutions in the area of low rank approximation. Given a matrix A, a truncated SVD provides the nearest matrix to A of a given rank. The famous Eckart-Young Theorem describes this precisely [1].

Theorem 2.2 (Eckart-Young Theorem). *Let $A = U\Sigma V^*$. If $k < r = \text{rank}(A)$ and*

$$A_k = U_k\Sigma_k V_k^*,$$

then

$$\min_{\text{rank}(B)=k} \|A - B\|_2 = \|A - A_k\|_2 = \sigma_{k+1}.$$

This result says that the closest a rank-k matrix can be to the matrix A is σ_{k+1} when measured in the 2-norm, and this matrix is given by the truncated SVD A_k. A similar result holds in the Frobenius norm except the closest we can be is $\sqrt{\sigma_{k+1}^2 + \cdots + \sigma_n^2}$.

1 https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

2.2 COMPUTER ARITHMETIC

Primary sources for this section include [2], [6], and [11]. Our aim is to cover some basics of floating-point arithmetic and error analysis. It is not intended to be a complete treatment of these topics, only to provide some background for the remainder of the thesis. We refer the reader to the above references for more information.

2.2.1 Floating-point arithmetic

The most prevalent choice for representing the real numbers on a computer is that of floating-point arithmetic. A floating-point number system $F \subset \mathbb{R}$ has elements of the form

$$y = \pm m \times \beta^{e-t}. \tag{2.2.1}$$

Four integer parameters characterize a floating point system F: the base β, the precision t and the minimum and maximum exponents $e_{\min}$ and $e_{\max}$, such that $e_{\min} \leqslant e \leqslant e_{\max}$. Another common way of expressing elements of F is given by:

$$y = \pm \beta^e \times .d_1 d_2 \dots d_t, \tag{2.2.2}$$

with each digit d_i satisfying $0 \leqslant d_i \leqslant \beta - 1$. The preferred choice is typically (2.2.1) due to its relative simplicity.

The integer m is called the *significand*. It satisfies $0 \leqslant m \leqslant \beta^t - 1$ and we say a floating-point system F is normalized if for every $y \in F$ such that y is nonzero, we have that $m \geqslant \beta^{t-1}$. This ensures that every nonzero y has a unique representation. In the representation given by (2.2.2), normalised numbers are those for which $d_1 > 0$.

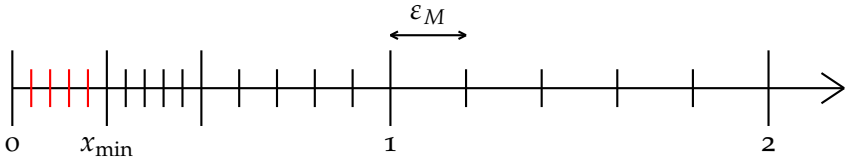


Figure 2.2.1: The positive numbers of a toy floating-point system with base $\beta = 2$. Shown are the machine epsilon ε_M, the minimum normalized number $x_{\min}$ and the subnormal numbers in red.

A floating-point system can be extended by including those numbers that possess the minimum exponent $e_{\min}$ and for which $0 < m < \beta^{t-1}$. These numbers are termed *subnormal*. In the representation given by (2.2.2), subnormal numbers have $d_1 = 0$. These numbers have fewer digits of precision than normalized numbers (as at least the first digit is 0), and are all lesser in magnitude than the smallest normalized floating-point number $x_{\min} = \beta^{e_{\min}-1}$.

An important feature of (normalized) floating-point numbers is their uneven spacing. At every power of β, the distance between successive floating-point numbers increases by a factor β. This is characterized by an important quantity, *machine epsilon*. This is defined as the distance from 1 to the next largest floating-point number and is given by $\varepsilon_M = \beta^{1-t}$. This spacing between numbers is then constant between 1 and β. Between β and β^2 it increases to β^{2-t}, and similarly decreases to β^{-t} between 1 and $1/\beta$. The spacing between the subnormals is constant (as they all have the same exponent) and is the same as that between $x_{\min}$ and $\beta x_{\min}$, given by $x_{\min}\varepsilon_M = \beta^{e_{\min}-t}$. This can all be seen in Figure 2.2.1.

One of the most important features of any floating-point system is how we choose to represent a number $x \in \mathbb{R}$ where $x \notin F$. We denote this by the transformation $x \to \mathrm{fl}(x)$ where $\mathrm{fl}(x)$ is chosen by some rounding rule we have prescribed. The first case to consider is where $\mathrm{fl}(x)$ does not lie in the range of F. If $|\mathrm{fl}(x)|$ is greater than the largest magnitude element of F, then we say $\mathrm{fl}(x)$ has overflowed. Similarly, if $|\mathrm{fl}(x)|$ is non-zero but less than the smallest magnitude non-zero element of F, we say $\mathrm{fl}(x)$ has underflowed. More details will be given on how these cases are handled

when we discuss IEEE arithmetic. For numbers which lie within the range of F, the most common rounding mode is round to nearest, which simply maps x to its nearest number in F, where different rules can be chosen for tie breaks. The following result [6, Thm. 2.2] bounds the relative distance from any real x within the range of F to its nearest floating-point representation.

Theorem 2.3. *If $x \in \mathbb{R}$ lies in the range of F then*

$$\mathrm{fl}(x) = x(1+\delta), \quad |\delta| \leqslant u = \frac{1}{2}\beta^{1-t}. \tag{2.2.3}$$

The relative error δ we call a *rounding error* and u is called the unit roundoff. We can also define directed rounding modes: round up, round down, round towards zero and round away from zero. Analogous results to Theorem 2.3 hold but with the substitution $u \to 2u$. If we assume elementary operations are correctly rounded, we arrive at what is called the *standard model of floating-point arithmetic* [6, Sec. 2.2]. Correctly rounded here means that we compute the true result exactly, and then round accordingly. We have

$$\mathrm{fl}(x \operatorname{op} y) = (x \operatorname{op} y)(1+\delta), \quad |\delta| \leqslant u, \quad \operatorname{op} = +, -, \times, \div, \sqrt{\cdot}, \tag{2.2.4}$$

which tells us that the computed value of any scalar operation is equal to the rounded exact answer. The model (2.2.4) proves crucial for error analysis of floating-point arithmetic.

From here on we discard the use of the general β for the base and simply assume $\beta = 2$, as it is in virtually any scenario of practical interest. Note for normalized floating-point numbers we require d_1 in (2.2.2) to be greater than 0, but for $\beta = 2$ the only choices for each d_i are 0 and 1. This means for any normalized base 2 floating-point number, we know $d_1 = 1$ and there is no need to store it. This means we can gain an extra bit for free in the significand, called the *hidden bit*.

Table 2.2.1: IEEE-754 floating-point formats

Arithmetic	Bits	t	$e_{\min}$	$e_{\max}$	u	Range
IEEE single (fp32)	32	$23+1$	-126	127	$2^{-24} \approx 6 \times 10^{-8}$	$10^{\pm 38}$
IEEE double (fp64)	64	$52+1$	-1022	1023	$2^{-53} \approx 1 \times 10^{-16}$	$10^{\pm 308}$

2.2.2 IEEE arithmetic

So far we have taken a theoretical perspective on floating-point arithmetic. Here we provide some details of its implementation in practice. A seminal moment was the publication of the IEEE standard 754 [8], which defines a particular floating-point arithmetic. The two formats specified by IEEE-754 are widespread in modern scientific computing and detailed in Table 2.2.1. The formats have base 2. A 1987 revision to the standard included base 10, but we are only concerned with base 2. The "+1" in the t column indicates that the formats take advantage of the hidden bit discussed in Section 5.3. For the rest of this thesis, we follow the convention seen in Table 2.2.1, where IEEE compliant floating point formats are denoted fpxx, with the trailing numbers equalling the number of bits the format has.

We further detail some of the specifications beyond Table 2.2.1.

- All arithmetic operations must be computed as if we had computed the exact result, and then rounded according to one of the allowed rounding modes. The standard includes the square root as an arithmetic operation. Elementary function such as exp, sin, and cos are not mentioned in the standard.

- The default rounding mode is round-to-nearest, with round to even (least significant bit 0) in case of a tie. Using this rounding mode, we satisfy (2.2.3). The other rounding modes supported by the standard are directed: round to $+\infty$, round to $-\infty$, and round to zero.

- Subnormal numbers are required to be supported, and so the IEEE formats will gradually underflow.

- IEEE arithmetic is a closed system, meaning the result of every arithmetic operation produces a result. We must introduce some special values that are used in exceptional situations. Without these special values, in situations like division by zero or taking the square root of a negative number, there would be no alternative but to abort the computation. See [2, Sec. 2.2] for a more exhaustive treatment.

  - A NaN (Not a Number) is generated by invalid operations, such as $0/0$, $0/\infty$ and $\sqrt{-1}$. Whenever a NaN then participates in any floating-point operation the result is a NaN. A NaN is unordered and unequal to everything including itself. A NaN is represented by exponent field $e_{\max}+1$ and a non-zero significand.

  - Infinity is represented by a zero significand and the same exponent field as a NaN with the sign bit distinguishing $+\infty$ and $-\infty$. An infinity is returned in the case of overflow. An interesting point to note is that while $0/0$ returns a NaN, division of a non-zero by zero returns the appropriate infinity.

  - Zero is represented by the exponent field $e_{\min}-1$ and a zero significand, with the sign bit distinguishing $+0$ and -0.

- IEEE-754 actually defined four precisions, those in Table 2.2.1, plus single extended and double extended. The extended precisions offer a little extra precision and range, with only a lower bound provided on how much extra precision is required. Extended precision is useful as it allows for the easy creation of accurate routines for the elementary functions [7]. It is also useful for conversion between binary floating-point and decimal, as it guarantees that in the sequence of conversions: floating-point $\rightarrow$ decimal $\rightarrow$ floating-point, we can recover the original floating-point number.

Table 2.2.2: New floating-point formats

Arithmetic	Bits	t	$e_{\min}$	$e_{\max}$	u	Range
IEEE half (fp16)	16	$10+1$	-14	15	$2^{-11} \approx 5 \times 10^{-4}$	$10^{\pm 5}$
Bfloat16	16	$7+1$	-126	127	$2^{-8} \approx 4 \times 10^{-3}$	$10^{\pm 38}$
IEEE quad (fp128)	128	$112+1$	-16382	16383	$2^{-114} \approx 1 \times 10^{-34}$	$10^{\pm 4932}$

2.2.3 The modern landscape

A 2008 revision to the standard [9] introduced two new formats: a quadruple precision (fp128) and a half precision (fp16). The half precision was originally defined only as a storage format. An early motivation for this format was computer graphics, where the reduced precision was still satisfactory. Another half precision format, bfloat16, was later proposed by Google. Bfloat16 also occupies 16 bits, but sacrifices some bits in the significand for increased dynamic range. Fp16 has approximately 4 decimal digits of precision, while bfloat16 has approximately 3. The exact specifications of these formats are given in Table 2.2.2.

2.2.4 Error analysis

Here we describe some key concepts in error analysis. Given some computed solution $\widehat{y} \approx f(x)$ computed in precision u, the field of error analysis is concerned with assessing the quality of the solution $\widehat{y}$. The best solution we could reasonably hope for is that the relative error in $\widehat{y}$ is approximately u, but this can be an unreasonable burden. Instead, we consider what we call the *backward error*. Generally, for some computed $\widehat{y} \approx f(x)$, we ask what is the smallest Δx such that $\widehat{y} = f(x + \Delta x)$. Many parts of this thesis are concerned with backward error analysis. Backward error analysis is attractive for two reasons. Firstly, if there are some inherent uncertainties in our input, and we can bound our backward error to be no larger than them, then we have certainly produced the best solution we can reasonably ask for. Secondly, we

can relate backward errors to the usual relative and absolute errors, called *forward errors* quite easily. Forward error is concerned with the distance of the computed solution $\widehat{y}$ from the true solution y. This is dependent on the conditioning of the problem, where we have the approximate rule of thumb that the forward error is bounded by the backward error times the condition number. Works by Turing [14] and von Neumann and Goldstine [3] contain ideas about backward error analysis, but it was Wilkinson who matured and popularized backward error analysis in the 1950s and 1960s [15], [16].

A primary focus in this thesis is analyzing how rounding errors accumulate and providing informative backward error bounds. Lemma 2.4 [6, Lem. 3.1] is an important result that is widespread in error analysis.

Lemma 2.4. *If* $|\delta_i| \leqslant u$ *and* $\rho_i = \pm 1$ *for* $i = 1 : n$, *and* $nu < 1$, *then*

$$\prod_{i=1}^{n}(1+\delta_i)^{\rho_i} = 1 + \theta_n,$$

where

$$|\theta_n| \leqslant \frac{nu}{1-nu} := \gamma_n.$$

The need for this result arises from considering the accumulation of rounding errors arising from the model (2.2.4). Consider as an example the inner product $y_n = a^T b$ where $a, b \in \mathbb{R}^n$. Computed in the usual recursive pattern $y_i = y_{i-1} + a_i b_i$, then it is easy to see by repeated application of (2.2.4) that we will have a final result of the form

$$\widehat{y}_n = a_1 b_1 (1+\theta_n) + a_2 b_2 (1+\theta_n') \ldots a_n b_n (1+\theta_2), \tag{2.2.5}$$

Each term on the right hand side of (2.2.5) has the form $a_i b_i(1+\alpha_i)$, where we certainly have $|\alpha| \leqslant \gamma_n$ for all i. This then leads to the standard backward error result for inner products. We have

$$\widehat{y}_n = (a+\Delta a)^T b = a^T(b+\Delta b), \quad |\Delta a| \leqslant \gamma_n |a|, \quad |\Delta b| \leqslant \gamma_n |b|, \tag{2.2.6}$$

where the inequalities above hold componentwise. As discussed, once we have a backward error result a forward error bound is readily available:

$$|\widehat{y}_n - y_n| \leqslant \gamma_n |a|^T |b|.$$

2.3 PROBABILITY THEORY

This section is intended to introduce some notions of probability theory. It is not intended to be a complete coverage, but to provide the reader with the necessary background for what will follow. We assume the reader is familiar with the concepts of probability and random variables. Much of this section draws from [10] and [12].

2.3.1 Random variables

A random variable X is said to be discrete if it takes values in a countable subset $\{x_1, x_2, \dots\}$ of $\mathbb{R}$. Otherwise it is said to be continuous. The probability mass function (PMF) of a discrete random variable X is given by $f_X(x) = \Pr(X = x)$. Suppose the set of all outcomes Ω is given by $\Omega = \{x_1, x_2, \dots, x_N\}$. We then have

$$\mathbb{E}(X) = \sum_{x_i \in \Omega} x_i \Pr(X = x_i).$$

For X a continuous random variable, define the cumulative distribution function F_X by

$$F_X(x) = \Pr(X \leqslant x).$$

If there exists a function f_X such that for all a,

$$F_X(a) = \int_{-\infty}^{a} f_X(x)dx,$$

then we call $f_X(x)$ the probability density function (PDF) of X and we have $F'_X(x) = f_X(x)$. The PDF is the continuous analogue of the PMF. We have that $\Pr(a \leqslant X \leqslant b) = \int_a^b f_X(x)dx$. Expected value and the higher moments are given by

$$\mathbb{E}(X^k) = \int_{-\infty}^{\infty} x^k f_X(x)dx, \quad k = 1, 2, \ldots$$

The expectation of a function g of a random variable is given similarly by

$$\mathbb{E}(g(X)) = \int_{-\infty}^{\infty} g(x) f_X(x)dx.$$

Variance, denoted $\mathrm{Var}(X)$, is given by

$$\mathrm{Var}(X) = \mathbb{E}[(X - \mathbb{E}(X))^2] = \mathbb{E}(X^2) - \mathbb{E}(X)^2.$$

Two random variables are described below.

1. A continuous random variable X is said to be *uniform* over the interval $[a, b]$ if it has a probability density function given by

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } a < x < b, \\ 0 & \text{otherwise.} \end{cases}$$

   We have $\mathbb{E}(X) = (a + b)/2$.

2. A continuous random variable X is said to be *normal* with parameters μ and $\sigma > 0$ if its probability density function is given by

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{-(x-\mu)^2/2\sigma^2}.$$

   We have $\mathbb{E}(X) = \mu$ and $\mathrm{Var}(X) = \sigma^2$. A normal random variable with parameters $\mu = 0$ and $\sigma = 1$ is often referred to as a standard normal random variable. We will later refer to a standard normal random variable as a *Gaussian* random variable.

Some definitions and discussion follows on the properties of independence, mean independence and correlation.

Definition 2.1. Discrete random variables X and Y are independent if and only if

$$\Pr(X = x, Y = y) = \Pr(X = x) \Pr(Y = y) \text{ for all } x, y,$$

that is their joint PMF is the product of their individual PMF's.

Definition 2.2. Continuous random variables X and Y are independent if and only if

$$f_{X,Y}(x, y) = f_X(x) f_Y(y) \text{ for all } x, y,$$

that is their joint PDF is the product of their individual PDF's.

Definition 2.3. The conditional probability of event $X = x$ given event $Y = y$ is given by

$$\Pr(X = x \mid Y = y) = \Pr(X = x, Y = y) / \Pr(Y = y).$$

Definition 2.4. For discrete random variables X and Y, the conditional expectation of X given $Y = y$ is given by

$$\mathbb{E}(X \mid Y = y) = \sum_x x P(X = x \mid Y = y).$$

Definition 2.5. For X and Y continuously distributed, the conditional expectation of X given that $Y = y$ is given by

$$\mathbb{E}(X \mid Y = y) \int_{-\infty}^{\infty} x f_{X|Y}(x \mid y) dx,$$

with $f_{X|Y}(x, y) = f_{X,Y}(x, y) / f_Y(y)$.

Definition 2.6. The conditional expectation $\mathbb{E}(X \mid Y)$ is the random variable that takes the value $\mathbb{E}(X \mid Y = y)$ when $Y = y$.

Definition 2.7. Random variables X is mean independent of Y if and only if $\mathbb{E}(X \mid Y) = \mathbb{E}(X)$.

Definition 2.8. Random variables X and Y are uncorrelated if and only if $\mathbb{E}(XY) = \mathbb{E}(X)\mathbb{E}(Y)$.

Lemma 2.5 (Law of total expectation)**.** *If X and Y are random variables defined on the same probability space and $\mathbb{E}(X)$ is defined, then*

$$\mathbb{E}(X) = \mathbb{E}(\mathbb{E}(X \mid Y)).$$

Lemma 2.6. *Let X and Y be discrete random variables. Then their independence implies their mean independence.*

Proof. From Definition 2.3 we have $\Pr(X = x \mid Y = y) = \Pr(X = x, Y = y)/\Pr(Y = y)$. From independence we have $\Pr(X = x, Y = y) = \Pr(X = x)\Pr(Y = y)$. Mean independence follows by a straightforward calculation.

$$\begin{aligned}
\mathbb{E}(X \mid Y = y) &= \sum_x x \Pr(X = x \mid Y = y) \\
&= \sum_x x \frac{\Pr(X = x, Y = y)}{\Pr(Y = y)} \\
&= \sum_x x \Pr(X = x) \\
&= \mathbb{E}(X).
\end{aligned}$$

□

Lemma 2.7. *Mean independence implies uncorrelated, that is, if X and Y are random variables and X is mean independent of Y then $\mathbb{E}(XY) = \mathbb{E}(X)\mathbb{E}(Y)$.*

Proof. By the law of total expectation,

$$\begin{aligned}\mathbb{E}(XY) &= \mathbb{E}(\mathbb{E}(XY \mid Y)) \\ &= \mathbb{E}(Y\mathbb{E}(X \mid Y)) \\ &= \mathbb{E}(Y\mathbb{E}(X))) \\ &= \mathbb{E}(Y)\mathbb{E}(X).\end{aligned}$$

□

The proof of Lemma 2.7 is valid for continuous or discrete random variables. We thus have the relationship

$$\text{Independence} \implies \mathbb{E}(X) = \mathbb{E}(X \mid Y) \implies \mathbb{E}(XY) = \mathbb{E}(X)\mathbb{E}(Y).$$

The converse is not true for either of the above. Unlike independence, mean independence is not symmetric, so $\mathbb{E}(X) = \mathbb{E}(X \mid Y)$ does not imply $\mathbb{E}(Y) = \mathbb{E}(Y \mid X)$.

2.3.2 Concentration inequalities

Concentration inequalities deal with the deviation of random variables from their mean. Sometimes called deviation bounds, they are a formal way of describing the intuition that if we sum many independent random variables, that sum will with very high probability be close to its expected value. The most straightforward results of this type are stated for independent random variables, but can be extended to cases where we do not have independence, as we will see in later chapters. The derivation of concentration inequalities requires the notion of *moment generating functions* and is beyond the scope of this thesis. We will state them without proof, but refer the reader to any of the provided references on concentration inequalities for further details.

Our motivation here is to introduce some well know scalar and matrix concentration inequalities to familiarise the reader with the nature of the bounds.

One of the best known results of this type is Hoeffding's bound [10, Thm. 4.12].

Theorem 2.8. *Let $X_1, \ldots, X_n$ be independent random variables such that for all $1 \leqslant i \leqslant n$, $\mathbb{E}(X_i) = \mu$ and $X_i \in [a, b]$. Then for any $t > 0$*

$$\left| \frac{1}{n} \sum_{i=1}^{n} X_i - \mu \right| \leqslant t,$$

with probability at least $1 - 2\exp(-2nt^2/(b-a)^2)$.

The way we have written this result is slightly non-standard. More typical would be to say

$$\left| \frac{1}{n} \sum_{i=1}^{n} X_i - \mu \right| \geqslant t,$$

with probability at *most* $2\exp(-2nt^2/(b-a)^2)$. The former convention is cleaner for our later treatment of this type of result in a numerical linear algebra setting.

Theorem 2.8 is a very general result, with no underlying assumption on the distribution of the random variables. Introducing more information about the random variables can lead to more refined results. In Theorem 2.9, one of many results known as *Bernstein's Inequality*, we incorporate information about the variance of the random variables to provide a tighter bound on the probability.

Theorem 2.9. *Let $X_1, \ldots, X_n$ be independent random variables with $\mathbb{E}(X_i) = 0$ and assume that $X_i \leqslant 1$. Let*

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} \mathrm{Var}(X_i).$$

Then for any $t > 0$

$$\frac{1}{n} \sum_{i=1}^{n} X_i < t$$

with probability at least $1 - \exp(-nt^2/(2(\sigma^2 + t/3)))$.

These types of result can be extended to the matrix setting. Expectation extends easily here. The expectation $\mathbb{E}(Z)$ of random matrix Z is simply the component-wise expectation. Consider Theorem 2.10 [13, Thm. 6.1.1].

Theorem 2.10. *Consider a finite sequence S_k of independent, random matrices with common dimension $m \times n$. Assume that*

$$\mathbb{E}(S_k) = 0, \quad \|S_k\|_2 \leqslant L$$

for all k. Introduce the random matrix $Z = \sum_k S_k$. Let $v(Z)$ be the matrix variance statistic:

$$\begin{aligned} v(Z) &= \max\{\|\mathbb{E}(ZZ^T)\|_2, \|\mathbb{E}(Z^TZ)\|_2\} \\ &= \max\{\|\sum_k \mathbb{E}(S_k S_k^T)\|_2, \|\sum_k \mathbb{E}(S_k^T S_k)\|_2\}. \end{aligned}$$

Then for all $t > 0$ we have that $\|Z\|_2 \leqslant t$ with probability at least

$$1 - (m+n)\exp\left(\frac{-t^2/2}{v(Z) + Lt/3}\right).$$

The particular results shown here will not be required later, but rather similar ones which will be introduced when required. Our aim here has been simply to introduce concentration inequalities to the reader.

REFERENCES

[1] C. Eckart and G. Young. "The approximation of one matrix by another of lower rank." *Psychometrika* 1.3 (Sept. 1936), pp. 211–218 (cited on p. 28).

[2] D. Goldberg. "What every computer scientist should know about floating-point arithmetic." *ACM Computing Surveys* 23.1 (Jan. 1991), pp. 5–48 (cited on pp. 29, 33).

[3] H. H. Goldstine and J. V. Neumann. "Numerical inverting of matrices of high order. II." *Proceedings of the American Mathematical Society* 2.2 (1951), pp. 188–202 (cited on p. 35).

[4] G. Golub and W. Kahan. "Calculating the singular values and pseudo-inverse of a matrix." *Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis* 2.2 (1965), pp. 205–224 (cited on p. 28).

[5] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Fourth. Baltimore, MD, USA: Johns Hopkins University Press, 2013, pp. xxi+756 (cited on p. 18).

[6] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. Second. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2002, pp. xxx+680 (cited on pp. 18, 24, 29, 31, 35).

[7] D. Hough. "Applications of the proposed IEEE 754 standard for floating-point arithetic". *Computer* 14.03 (1981), pp. 70–74 (cited on p. 33).

[8] "IEEE Standard for Binary Floating-Point Arithmetic". *ANSI/IEEE Standard 754-1985* (Oct. 1985) (cited on p. 32).

[9] "IEEE Standard for Floating-Point Arithmetic." *IEEE Std 754-2008* (Aug. 2008), pp. 1–70 (cited on p. 34).

[10] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. 2nd. USA: Cambridge University Press, 2017 (cited on pp. 37, 42).

[11] J.-M. Muller, N. Brunie, F. de Dinechin, C.-P. Jeannerod, M. Joldes, V. Lefèvre, G. Melquiond, N. Revol, and S. Torres. *Handbook of Floating-Point Arithmetic*. Second. Boston, MA, USA: Birkhäuser, 2018, pp. xxv+627 (cited on p. 29).

[12] H. Tijms. *Understanding Probability*. 3rd ed. Cambridge University Press, 2012 (cited on p. 37).

[13] J. A. Tropp. "An introduction to matrix concentration inequalities." *Foundations and Trends in Machine Learning* 8.1-2 (2015), pp. 1–230 (cited on p. 43).

[14] A. M. Turing. "Rounding-off errors in matrix processes." *The Quarterly Journal of Mechanics and Applied Mathematics* 1.1 (Jan. 1948), pp. 287–308. eprint: https://academic.oup.com/qjmam/article-pdf/1/1/287/5323145/1-1-287.pdf (cited on p. 35).

[15] J. H. Wilkinson. *The Algebraic Eigenvalue Problem*. USA: Oxford University Press, Inc., 1988 (cited on p. 35).

[16] J. H. Wilkinson. *Rounding Errors in Algebraic Processes*. USA: Dover Publications, Inc., 1994 (cited on p. 35).

3 STOCHASTIC ROUNDING AND ITS PROBABILISTIC BACKWARD ERROR ANALYSIS

Abstract. Stochastic rounding rounds a real number to the next larger or smaller floating-point number with probabilities 1 minus the relative distances to those numbers. It is gaining attention in deep learning because it can increase the success of low precision computations. We compare basic properties of stochastic rounding with those for round to nearest, finding properties in common as well as significant differences. We prove that for stochastic rounding the rounding errors are mean independent random variables with zero mean. We derive a new version of our probabilistic error analysis theorem from [N. J. Higham, and T. Mary, *SIAM J. Sci. Comput.*, 41 (2019), pp. A2815–A2835], weakening the assumption of independence of the random variables to mean independence. These results imply that for a wide range of linear algebra computations the backward error for stochastic rounding is unconditionally bounded by a multiple of $\sqrt{n}u$ to first order, with a certain probability, where n is the problem size and u is the unit roundoff. This is the first scenario where the rule of thumb that one can replace nu by $\sqrt{n}u$ in a rounding error bound has been shown to hold without any additional assumptions on the rounding errors. We also explain how stochastic rounding avoids the phenomenon of stagnation in sums, whereby small addends are obliterated by round to nearest when they are too small relative to the sum.

Keywords: floating-point arithmetic, rounding error analysis, numerical linear algebra, stochastic rounding, round to nearest, probabilistic backward error analysis, stagnation.

2010 MSC: 65G50, 65F05.

3.1 INTRODUCTION

The results of most elementary floating-point operations can not themselves be represented as floating-point numbers. This simple fact leads to one of the defining features of floating-point arithmetic: rounding error. To define a floating-point arithmetic we must prescribe how to round the result of an operation to a nearby floating-point number. The IEEE standard 754 for binary floating-point arithmetic [21] defines four rounding modes.

- Round to nearest. The default, where we round towards even (least significant bit 0) to break ties.
- Round towards 0.
- Round towards $+\infty$.
- Round towards $-\infty$.

The latter three modes are called directed rounding modes. Here, we consider two stochastic rounding modes. Let $F \subseteq \mathbb{R}$ denote the floating-point number system. In the first mode, we round $x \in \mathbb{R}$ with $x \notin F$ to the next larger or next smaller floating-point number with a probability that is 1 minus the relative distance of x to each of

those numbers. In the second mode, we round up or down with equal probability. For $x \in \mathbb{R}$, define

$$\lfloor x \rfloor = \max\{\, y \in F : y \leqslant x \,\}, \quad \lceil x \rceil = \min\{\, y \in F : y \geqslant x \,\},$$

so that $\lfloor x \rfloor \leqslant x \leqslant \lceil x \rceil$, with equality throughout if $x \in F$. For $x \notin F$, $\lfloor x \rfloor$ and $\lceil x \rceil$ are adjacent floating-point numbers. For $x \in \mathbb{R}$ with $x \notin F$ the two stochastic rounding modes are

$$\text{mode 1:} \qquad \mathrm{fl}(x) = \begin{cases} \lceil x \rceil \text{ with probability } p = (x - \lfloor x \rfloor)/(\lceil x \rceil - \lfloor x \rfloor), \\ \lfloor x \rfloor \text{ with probability } 1 - p, \end{cases} \tag{3.1.1}$$

$$\text{mode 2:} \qquad \mathrm{fl}(x) = \begin{cases} \lceil x \rceil \text{ with probability } 1/2, \\ \lfloor x \rfloor \text{ with probability } 1/2. \end{cases} \tag{3.1.2}$$

Stochastic rounding is an old idea, proposed in the 1950s and 1960s by Barnes, Cooke-Yarborough, and Thomas [2], Forysthe [10], [9] and Hull and Swenson [20]. It is attracting renewed interest in deep learning, especially where low precision arithmetic is used. It is shown in [14], in the context of neural network training, that using a 16-bit fixed-point representation with mode 1 stochastic rounding can be as effective as using 32-bit floating-point numbers with round to nearest. Stochastic rounding solves the problem of the obliteration of small parameter updates in the neural network, which is an instance of what we call stagnation. If a parameter ϕ is updated by a quantity h that is less than half the spacing of the floating-point numbers (or fixed-point numbers) around ϕ then $\mathrm{fl}(\phi + h) = \phi$ with round to nearest, so the information in h is lost. Stochastic rounding helps to preserve this information. Much recent work applies stochastic rounding in neural network training and inference; see, for example, [6], [8], [26], [29], [30], [38], [41], [42], [46], and the references therein.

Another application where mode 1 stochastic rounding has been shown to improve accuracy with fixed-point arithmetic is the numerical solution of neural ODEs [19].

Much work on stochastic rounding with floating-point arithmetic has focused on using it to validate numerical methods through an empirical approach. The CESTAC method [5], [39] and its implementation CADNA [25], [34] use mode 2 stochastic rounding, termed "stochastic arithmetic", to detect instabilities in numerical routines and to provide estimates of the accuracy of the computed results. Further references on this topic include [13], [12], [40].

Parker's Monte Carlo arithmetic [31], [32] is more general than stochastic rounding, not least because as well as randomly rounding it can randomly perturb the input to a floating-point operation and the output of it.

We are not aware of any analysis of stochastic rounding or any work on rounding error analysis for stochastic rounding. The purpose of this paper is to fill this gap in the literature. We make the following contributions.

- We analyze the properties of stochastic rounding in floating-point arithmetic vis-à-vis the properties of round to nearest, finding both common properties and significant differences.
- We show that the recent probabilistic backward error analysis of Higham and Mary [16], which assumes that rounding errors are independent random variables with zero mean, holds with the weaker assumption of mean independence. We also show that mode 1 stochastic rounding produces rounding errors that are mean independent random variables with zero mean. We conclude that the long-standing rule of thumb that one can replace a worst-case error bound nu by a more realistic (probabilistic) error bound $\sqrt{n}u$ [43, p. 318], [44, p. 26] holds unconditionally for stochastic rounding.
- We show that the expected value of a computed result from mode 1 stochastic rounding is the true value for summation, inner products, matrix–vector and

matrix–matrix products, and the solution of triangular systems, and we explain why this property does not extend to matrix factorizations.

- We prove that mode 1 stochastic rounding avoids stagnation in summation and thereby can lead to more accurate results than round to nearest.

We begin, in section 3.2, by recalling some basic properties of floating-point arithmetic. In section 3.3 we investigate properties of stochastic rounding and compare them with key properties of round to nearest. In section 3.4 we generalize the probabilistic backward error analysis of Higham and Mary [16], showing that the assumption that rounding errors are independent random variables can be relaxed to them being mean independent random variables. Then, in section 3.5, we show that this strengthened analysis applies to mode 1 stochastic rounding, which therefore enjoys unconditional $\sqrt{n}u$ error bounds in place of the worst-case nu bounds. In section 3.6 we analyze the expected value of computations under mode 1 stochastic rounding. We illustrate the benefits of the $\sqrt{n}u$ error bounds for mode 1 stochastic rounding in section 3.7 with numerical experiments on sums and inner products. Finally, we give concluding remarks in section 3.8.

3.2 FLOATING-POINT ARITHMETIC

We recall some basic properties of floating-point arithmetic. For more details, see [11], [15, Chap. 2], [28]. A number y in the floating-point number system F has the form

$$y = \pm m \times \beta^{e-t}, \tag{3.2.1}$$

which involves four integers:

- β is the base, which is 2 throughout this paper,

- t is the precision,
- e is the exponent, which satisfies $e_{\min} \leqslant e \leqslant e_{\max}$, and
- m is the significand, which satisfies $0 \leqslant m \leqslant \beta^t - 1$.

Normalized numbers are those for which $m \geqslant \beta^{t-1}$. The machine epsilon ε is the distance from 1 to the next larger floating-point number and is given by $\varepsilon = \beta^{1-t}$. The spacing of floating-point numbers increases by a factor β at each power of β. For $\beta = 2$ the spacing in the interval $(1/2, 1]$ is $\varepsilon/2 = u = 2^{-t}$, the unit roundoff. With round to nearest it can be shown that [15, Thm. 2.2]

$$\mathrm{fl}(x) = x(1+\delta), \quad |\delta| \leqslant u. \tag{3.2.2}$$

The standard model of floating-point arithmetic assumes that the elementary operations and the square root are correctly rounded (as indeed is the case for IEEE standard arithmetic [21]), so that with round to nearest they satisfy

$$\mathrm{fl}(x \operatorname{op} y) = (x \operatorname{op} y)(1+\delta), \quad |\delta| \leqslant u, \quad \operatorname{op} \in \{+, -, *, /, \sqrt{}\}. \tag{3.2.3}$$

Under stochastic rounding we define the elementary floating-point operations $+, -, *, /, \sqrt{}$ to be the stochastically rounded exact ones. Therefore for stochastic rounding, equations (3.2.2) and (3.2.3) hold with u replaced by $2u$:

$$\mathrm{fl}(x) = x(1+\delta), \qquad |\delta| \leqslant 2u, \tag{3.2.4a}$$

$$\mathrm{fl}(x \operatorname{op} y) = (x \operatorname{op} y)(1+\delta), \quad |\delta| \leqslant 2u, \quad \operatorname{op} \in \{+, -, *, /, \sqrt{}\}. \tag{3.2.4b}$$

We make reference throughout to various floating-point systems, the parameters of which are shown in Table 3.2.1. All those beginning with "fp" are from the IEEE standard. Bfloat16 [22] is a half-precision format supported by the Google Tensor

Table 3.2.1: Parameters of floating-point systems. (sig., exp.) denotes number of bits in significand (including implicit most significant bit) and exponent, u is the unit roundoff, $x_{\min}$ is the smallest normalized positive number, and $x_{\max}$ is the largest finite number.

	(sig., exp.)	u	$x_{\min}$	$x_{\max}$
bfloat16	(8, 8)	3.91×10^{-3}	1.18×10^{-38}	3.39×10^{38}
fp16	(11, 5)	4.88×10^{-4}	6.10×10^{-5}	6.55×10^{4}
fp32	(24, 8)	5.96×10^{-8}	1.18×10^{-38}	3.40×10^{38}
fp64	(53, 11)	1.11×10^{-16}	2.22×10^{-308}	1.80×10^{308}

Processing Unit[1] (TPU), the NVIDIA A100 GPU, the Intel Cooper Lake processor, and the Armv8-A architecture [1].

3.3 PROPERTIES OF STOCHASTIC ROUNDING

In this section, stochastic rounding refers to either mode 1 or mode 2, defined by (3.1.1) and (3.1.2), and all the results are valid for both. We compare stochastic rounding with round to nearest, identifying properties in common as well as significant differences that should be borne in mind when stochastic rounding is used.

3.3.1 Properties that continue to hold

We begin by identifying properties of round to nearest that continue to hold under stochastic rounding. First, we note that $\mathrm{fl}(\mathrm{fl}(x)) = \mathrm{fl}(x)$ with stochastic rounding, that is, rounding a floating-point number leaves it unchanged.

Sterbenz's lemma [15, Thm. 2.5], [37] is a property of floating-point numbers that is independent of the rounding mode, so it certainly holds for stochastic rounding.

1 https://cloud.google.com/tpu/

Lemma 3.1 (Sterbenz). *If x and y are floating-point numbers with $y/2 \leqslant x \leqslant 2y$ then $\mathrm{fl}(x-y) = x-y$ under stochastic rounding (assuming $x-y$ does not underflow).*

Under round to nearest we have (in base 2, but not for all bases [15, Probs. 2.7, 2.8]) that for floating-point numbers x and y with $x \leqslant y$

$$x \leqslant \mathrm{fl}((x+y)/2) \leqslant y. \tag{3.3.1}$$

These inequalities are an immediate consequence of the monotonicity of round to nearest, where monotonicity of rounding is the property that for $x \in \mathbb{R}$ and $y \in \mathbb{R}$, the inequality $x \leqslant y$ implies $\mathrm{fl}(x) \leqslant \mathrm{fl}(y)$. We show that they remain true for stochastic rounding, even though it is not monotonic (as shown in the next section). Since division by 2 is exact in base 2 arithmetic, we need to show that $2x \leqslant \mathrm{fl}(x+y) \leqslant 2y$. For the case $x = y$, the inequalities trivially hold. We thus consider $x < y$. Let $y = x + \delta$, where $\delta > 0$. Then $x + y = 2y - \delta < 2y$, so $\mathrm{fl}(x+y) \leqslant 2y$. Furthermore, $x + y = 2y - \delta \geqslant 2y - 2\delta = 2(y-\delta) = 2x$, so $\mathrm{fl}(x+y) \geqslant 2x$.

3.3.2 Properties that no longer hold

Some properties that are trivial under round to nearest do not hold under stochastic rounding. Since rounding is probabilistic, two different evaluations of $\mathrm{fl}(x)$ can give different results. Similarly, in general we have

$$\begin{aligned}
&\mathrm{fl}(|x|) \neq |\mathrm{fl}(x)|,\\
&\mathrm{fl}(-x) \neq -\mathrm{fl}(x),\\
&\mathrm{fl}(2^p x) \neq 2^p \mathrm{fl}(x), \quad p \text{ an integer},
\end{aligned}$$

but in each case the two possible values of the left-hand side are equal to the two possible values of the right-hand side (in the third case this follows from $\lceil 2^p x \rceil = 2^p \lceil x \rceil$ and $\lfloor 2^p x \rfloor = 2^p \lfloor x \rfloor$).

Round to nearest is monotonic but stochastic rounding is not: if we have two adjacent floating-point numbers $a < b$, then for $a < x \leqslant y < b$, $\mathrm{fl}(x) > \mathrm{fl}(y)$ is possible under stochastic rounding.

In [7], [15, Prob. 2.12] it is shown that for x satisfying $1 \leqslant x < 2$, $\mathrm{fl}(x * (1/x))$ is either 1 or $1 - \varepsilon/2$ with round to nearest, where ε is the machine epsilon. Under stochastic rounding we have two more possibilities for the result.

Theorem 3.2. *For $1 \leqslant x < 2$, $\mathrm{fl}(x * (1/x)) \in \{1 - \varepsilon, 1 - \varepsilon/2, 1, 1 + \varepsilon\}$ under stochastic rounding.*

Proof. The spacing of the floating-point numbers in the interval $(1/2, 1]$ is $\varepsilon/2$. This means that under stochastic rounding we have

$$\begin{aligned} & \left| \frac{1}{x} - \mathrm{fl}\Big(\frac{1}{x}\Big) \right| < \frac{\varepsilon}{2} \\ \Longrightarrow & \left| 1 - x\mathrm{fl}\Big(\frac{1}{x}\Big) \right| < \frac{x\varepsilon}{2} < \varepsilon \\ \Longrightarrow & 1 - \varepsilon < x\mathrm{fl}\Big(\frac{1}{x}\Big) < 1 + \varepsilon. \end{aligned} \tag{3.3.2}$$

The floating-point numbers in the interval $[1 - \varepsilon, 1 + \varepsilon]$ are $\{1 - \varepsilon, 1 - \varepsilon/2, 1, 1 + \varepsilon\}$. We therefore have $\mathrm{fl}(x * \mathrm{fl}(1/x)) \in \{1 - \varepsilon, 1 - \varepsilon/2, 1, 1 + \varepsilon\}$. □

Consider the computation of $\mathrm{fl}(n * \mathrm{fl}(m/n))$, where m and n are integers. If m/n is a floating-point number then $\mathrm{fl}(n * \mathrm{fl}(m/n)) = \mathrm{fl}(n * (m/n)) = \mathrm{fl}(m) = m$ for any rounding scheme, as no rounding takes place. For round to nearest, Kahan proved that the same identity holds for many other choices of m and n [11, Thm. 7]. Recall that a floating-point number has precision t and that we are assuming base 2.

Theorem 3.3 (Kahan)**.** *Let m and n be integers such that $|m| < 2^{t-1}$ and $n = 2^i + 2^j$ for some i and j. Then $\mathrm{fl}(n * \mathrm{fl}(m/n)) = m$ with round to nearest.*

The sequence of allowable n begins $2, 3, 4, 5, 6, 8, 9, 10, 12, 16, 17, 18, 20$ (and is A048645 in the On-Line Encyclopedia of Integer Sequences [36]), so Kahan's theorem covers many common cases. As an example of where the result is useful, if we partition $[0, 1]$ into n intervals of length $h = 1/n$, we may want, for consistency in a computation, that $\mathrm{fl}(nh) = 1$. Kahan's result shows that n does not need to be a power of 2 for this condition to hold.

Theorem 3.3 does not hold for stochastic rounding because there are three possibilities for the computed result, as the next result shows.

Theorem 3.4. *Let m and n be integers such that $|m| < 2^{t-1}$ and $n = 2^i + 2^j$ for some i and j. Under stochastic rounding, $\mathrm{fl}(n * \mathrm{fl}(m/n))$ is either m, the next smaller floating-point number, or the next larger floating-point number.*

Proof. The proof is a modification of the proof of [11, Thm. 7]. Without loss of generality we can assume that $m > 0$. It is harmless to scale n and m by powers of 2, since it changes only the exponents. Scale n so that $2^{t-1} \leqslant n < 2^t$ and scale m so that $1/2 \leqslant q = m/n < 1$. We then have $2^{t-2} \leqslant m < 2^t$. Since the original m has been reduced by at most a factor 2, m now has at most 1 bit to the right of the binary point. We will show that $\overline{q} = \mathrm{fl}(m/n) = \mathrm{fl}(q)$ satisfies

$$|n\overline{q} - m| \leqslant \frac{1}{4}. \tag{3.3.3}$$

Since m has at most 1 bit to the right of the binary point, if (3.3.3) is satisfied then under stochastic rounding $\mathrm{fl}(n\bar{q})$ will equal either m or one of the two adjacent floating-point numbers. (It would, in fact, be enough to prove (3.3.3) with $1/2$ on the right-hand side.)

We now seek to bound $|n\overline{q} - m|$. Write $q = .q_1 q_2 \ldots$ and let $\widehat{q} = .q_1 q_2 \ldots q_t 1$. From the proof of [11, Thm. 7] we have $|\widehat{q} - q| \geqslant 1/(n \times 2^{t+1-r})$, where n must have the form $n = 2^{t-1} + 2^r$ and $r \leqslant t - 2$. Assume $q < \widehat{q}$. The proof for $q > \widehat{q}$ is similar. We now have two cases.

Case 1: For $q < \widehat{q}$, with round to nearest we would necessarily round down and so $\overline{q} = \widehat{q} - 2^{-t-1} =: \overline{q}_d$. This is one possibility with stochastic rounding. In this case we have $n\overline{q}_d < nq = m$ and so

$$\begin{aligned} |m - n\overline{q}_d| &= m - n\overline{q}_d = n(q - \overline{q}_d) \\ &= n(q - \widehat{q} + 2^{-t-1}) \\ &\leqslant n\left(2^{-t-1} - \frac{1}{n \times 2^{t+1-r}}\right) = \frac{1}{4}. \end{aligned}$$

Case 2: With stochastic rounding we have another possibility. As $\overline{q}_d < q$, the other value we can compute for $\overline{q}$ must be $\overline{q}_u = \overline{q}_d + 2^{-t}$. We then have $\overline{q}_u = \widehat{q} + 2^{-t-1}$. Following a similar procedure as before we can show $|m - n\overline{q}_u| \leqslant 1/4$, concluding the proof. □

With round to nearest (and specifically for base 2), we have that $\mathrm{fl}(\sqrt{x^2}) = |x|$ for x a floating-point number [15, Prob. 2.20], barring underflow and overflow. We show that this identity can fail under stochastic rounding, and $\mathrm{fl}(\sqrt{x^2})$ can be one of three values.

Theorem 3.5. *For a floating-point number $x \in (1,2)$, $\mathrm{fl}(\sqrt{x^2}) \in \{|x| - \varepsilon, |x|, |x| + \varepsilon\}$ under stochastic rounding.*

Proof. By (3.2.4b), we have $\sqrt{\mathrm{fl}(x^2)} = \sqrt{x^2(1+\delta)} = |x|(1+\delta)^{1/2}$, $|\delta| \leqslant 2u$, and

$$|x|(1+\delta)^{1/2} = |x|\left(1 + \frac{\delta}{\sqrt{1+\delta}+1}\right) =: |x| + \theta.$$

To maximize $|\theta|$, take $\delta = -2u$ and $x = 2 - 2u$, which is the largest floating-point number that lies in $(1,2)$. Then

$$|\theta| \leqslant \frac{(2-2u)2u}{\sqrt{1-2u}+1}.$$

For $u \leqslant 1/2$, we have $|\theta| \leqslant 2u = \varepsilon$. Since the spacing of the floating-point numbers on $(1,2)$ is ε, it follows that $\mathrm{fl}(\sqrt{\mathrm{fl}(x^2)})$ can round to any of $\{|x| - \varepsilon, |x|, |x| + \varepsilon\}$. □

We have verified by numerical experiments in MATLAB that each of the cases for the computed results in Theorems 3.2, 3.4, and 3.5 is attainable in bfloat16, fp16, and fp32 arithmetic for some choices of the data.

Theorem 3.5 implies that the inequality $\mathrm{fl}(x/\sqrt{x^2+y^2}) \leqslant 1$ (which always holds under round to nearest [15, Prob. 2.21]) can fail under stochastic rounding. This means that the formula $\mathrm{acos}(x/\sqrt{x^2+y^2})$ for one of the angles in a right-angled triangle with smallest sides of length x and y can fail. Indeed, take y to be zero, or so small that $\mathrm{fl}(x^2+y^2) = \mathrm{fl}(x^2)$ holds with high probability. For $x > 0$, Theorem 3.5 shows that $\mathrm{fl}(\sqrt{x^2}) = x - \varepsilon$ is possible, in which case

$$\frac{x}{\mathrm{fl}(\sqrt{x^2})} = \frac{x}{x-\varepsilon} > 1,$$

and it follows that under stochastic rounding the result can exceed 1.

Stochastic rounding has two drawbacks in common with a fused multiply-add operation [15, sect. 2.6]. First, if we compute the modulus squared of a complex number from the formula

$$(x - \mathrm{i}y)(x + \mathrm{i}y) = x^2 + y^2 + \mathrm{i}(xy - yx),$$

the result may be nonreal, since $\mathrm{fl}(xy) \neq \mathrm{fl}(yx)$ is possible. Second, in evaluating a discriminant $b^2 - ac$, even if $b^2 \geqslant ac$ the discriminant can evaluate as negative because of the nonmonotonicity of stochastic rounding, which is problematic if $\sqrt{b^2 - ac}$ must be computed.

Under round to nearest (in base 2) we have for floating-point numbers x and y that $\mathrm{err}(x,y) = x + y - \mathrm{fl}(x+y)$ satisfies $|\mathrm{err}(x,y)| \leqslant \min(|x|,|y|)$ [15, Prob. 4.6], [35]. We show this to be false under stochastic rounding by counterexample. For $x = 4$ and

$y = \varepsilon$ we have $\mathrm{fl}(x+y) \in \{4, 4+4\varepsilon\}$ as the spacing of the floating-point numbers in the interval $[4,8]$ is 4ε. The bound is satisfied for $\mathrm{fl}(x+y) = 4$ but for $\mathrm{fl}(x+y) = 4+4\varepsilon$, $|\mathrm{err}(x,y)| = |4+\varepsilon-(4+4\varepsilon)| = 3\varepsilon > \min(|x|,|y|)$.

Vital to compensated summation algorithms is the fact that for floating-point numbers a and b, if $s = \mathrm{fl}(a+b)$ with round to nearest then $t = a+b-s$ is a floating-point number, which can be computed by the following algorithm.

Algorithm 3.1 (FastTwoSum) Given floating-point numbers a, b such that $|a| \geqslant |b|$, compute (with round to nearest) s and t such that $s = \mathrm{fl}(a+b)$ and $s+t = a+b$ exactly.

1: $s \leftarrow a+b$
2: $z \leftarrow s-a$
3: $t \leftarrow b-z$

Under stochastic rounding, the computed $\widehat{t}$ from Algorithm 3.1 is not exact, but we can bound the error. From [13, Prop. 4.3], we have

$$|\widehat{t}-t| \leqslant 2u|t| \tag{3.3.4}$$

if each arithmetic operation is performed with a directed rounding mode and hence also for stochastic rounding. Based on this argument, error bounds are provided in [13], [12] for compensated summation algorithms under directed rounding schemes, and these bounds therefore hold under stochastic rounding. We note that while the computation of t is no longer exact, compensated summation algorithms still prove accurate under stochastic rounding.

While the collection of properties analyzed above is by no means exhaustive, it demonstrates that it would be dangerous to simply replace round to nearest by stochastic rounding in a given computation. One should carefully consider whether the computation is dependent on properties of round to nearest beyond the model (3.2.3) and, if they are, check whether they remain true for stochastic rounding.

3.4 PROBABILISTIC BACKWARD ERROR ANALYSIS

We wish to exploit the properties of stochastic rounding in backward error analysis. Standard backward error analysis based on the model (3.2.3) remains valid with $u \leftarrow 2u$ by (3.2.4b), but we wish to take advantage of the statistical properties of stochastic rounding. In this section we develop probabilistic backward error bounds, which we apply to stochastic rounding in the next section.

3.4.1 Summary of probabilistic backward error bounds under independence

It is standard practice to express backward error results in terms of the constant $\gamma_n = nu/(1-nu)$. This constant arises when rounding error terms $1+\delta_i$ with $|\delta_i| \leqslant u$ are collected in a product and the distance of the product from 1 is bounded using the following lemma [15, Lem 3.1].

Lemma 3.6. *If $|\delta_i| \leqslant u$ and $\rho_i = \pm 1$ for $i = 1:n$, and $nu < 1$, then*

$$\prod_{i=1}^{n}(1+\delta_i)^{\rho_i} = 1+\theta_n, \quad |\theta_n| \leqslant \gamma_n. \tag{3.4.1}$$

The inequality $|\theta_n| \leqslant \gamma_n$ is a worst-case bound that is often pessimistic in practice and so it can fail to provide a good indication of the size of the error of a typical computation. This weakness is especially relevant in the context of large scale and/or low precision computations, since for large values of n or u, γ_n can exceed 1, in which case the worst-case bound is not able to guarantee even a single correct digit.[2] For example, with the half-precision arithmetics fp16 and bfloat16, $nu > 1$ for $n > 2048$ and $n > 256$, respectively.

2 Indeed once $nu > 1$, the bound (4.1.5) is not valid. By exploiting the round to nearest property it is possible to relax the condition $nu < 1$ at the cost of more complicated proofs [24], [33], but the bound will still be large for $nu > 1$.

These observations have generated a renewed interest in analyzing rounding errors from a probabilistic point of view. In particular, a systematic backward error analysis based on a probabilistic model that assumes rounding errors to be independent random variables of mean zero has recently been developed by Higham and Mary [16].

We state the following result, which is a minor rewriting of [16, Thm. 2.4] with the change of variable $\lambda \leftarrow \lambda/(1-u)$. Define

$$\widetilde{\gamma}_n(\lambda) = \exp\left(\frac{\lambda\sqrt{n}u + nu^2}{1-u}\right) - 1 = \lambda\sqrt{n}u + O(u^2). \tag{3.4.2}$$

Lemma 3.7. *Let $\delta_1, \delta_2, \ldots, \delta_n$ be independent random variables of mean zero such that $|\delta_i| \leqslant u$ for all i, and let $\rho_i = \pm 1$, $i = 1:n$. Then for any constant $\lambda > 0$,*

$$\prod_{i=1}^{n}(1+\delta_i)^{\rho_i} = 1 + \theta_n, \quad |\theta_n| \leqslant \widetilde{\gamma}_n(\lambda)$$

holds with probability at least $P(\lambda) = 1 - 2\exp(-\lambda^2/2)$.

The significance of the lemma is that it shows that if the rounding errors are assumed to be independent random variables of mean zero then $\gamma_n = nu + O(u^2)$ can be replaced by the relaxed constant $\widetilde{\gamma}_n(\lambda) = \lambda\sqrt{n}u + O(u^2)$ with a probability that is high even for modest λ. It justifies the long-standing rule of thumb that one can take the square root of an error constant because of statistical effects in rounding error propagation.

As an example of what can be proved using Lemma 3.7 we state the following result for inner products from [16, Thm. 3.1]. We define

$$Q(\lambda, n) = 1 - n(1 - P(\lambda)) = 1 - 2n\exp(-\lambda^2/2).$$

Theorem 3.8 (inner products). *Let $y = a^T b$, where $a, b \in \mathbb{R}^n$, be evaluated in floating-point arithmetic. If the rounding errors are independent random variables of mean zero then no matter what the order of evaluation the computed $\widehat{y}$ satisfies*

$$\widehat{y} = (a + \Delta a)^T b = a^T (b + \Delta b), \quad |\Delta a| \leqslant \widetilde{\gamma}_n(\lambda)|a|, \ |\Delta b| \leqslant \widetilde{\gamma}_n(\lambda)|b| \tag{3.4.3}$$

with probability at least $Q(\lambda, n)$.

Lemma 3.7 and Theorem 3.8 rely, however, on the two key assumptions that rounding errors are independent and have zero mean. With deterministic rounding modes these assumptions do not always hold and indeed examples where the probabilistic bound is violated are provided in [16] and are used in our experiments in section 3.7.

3.4.2 Generalizing backward error bounds to mean independence

We now weaken the independence assumption in Lemma 3.7 to *mean independence*. A random variable X is said to be mean independent of another random variable Y if its conditional expectation given Y is equal to its unconditional expectation, that is, $\mathbb{E}(X \mid Y) = \mathbb{E}(X)$. Random variables $\delta_1, \delta_2, \ldots$ are mean independent if $\mathbb{E}(\delta_k \mid \delta_1, \ldots, \delta_{k-1}) = \mathbb{E}(\delta_k)$ for all k. Independent random variables are mean independent, but the converse is not true in general. We will show in the next section that the rounding errors from mode 1 stochastic rounding are mean independent.

The probabilistic error analyses of [17], [23] prove that the assumption of independence of rounding errors can be relaxed to mean independence in the special case of inner product-based computations. We now show that it is possible to do so for general linear algebra operations, by deriving a version of Lemma 3.7 that requires only mean independence. To do so, we need the concept of a martingale.

Definition 3.1 (martingale). A sequence of random variables $E_0, \dots, E_n$ is a martingale if, for all k, $\mathbb{E}(|E_k|) < \infty$ and

$$\mathbb{E}\big(E_k \mid E_0, \dots, E_{k-1}\big) = E_{k-1}.$$

We also need the following inequality [27, Thm. 13.4].

Lemma 3.9 (Azuma–Hoeffding inequality). *Let $E_0, \dots, E_n$ be a martingale such that $|E_k - E_{k-1}| \leqslant c_k$, for $k = 1\colon n$. Then for any $\lambda > 0$,*

$$\Pr\left(|E_n - E_0| \geqslant \lambda \left(\sum_{k=1}^{n} c_k^2\right)^{1/2}\right) \leqslant 2\exp\big(-\lambda^2/2\big).$$

We are ready for the main result, which is a version of Lemma 3.7 with the independence assumption replaced by the weaker assumption of mean independence.

Theorem 3.10. *Let $\delta_1, \delta_2, \dots, \delta_n$ be random variables of mean zero with $|\delta_k| \leqslant u$ for all k such that $\mathbb{E}(\delta_{k+1} \mid \delta_1, \dots, \delta_k) = \mathbb{E}(\delta_{k+1}) = 0$ for $k = 1\colon n-1$. Then for $\rho_i = \pm 1$, $i = 1\colon n$ and any constant $\lambda > 0$,*

$$\prod_{i=1}^{n}(1+\delta_i)^{\rho_i} = 1 + \theta_n, \quad |\theta_n| \leqslant \widetilde{\gamma}_n(\lambda) \tag{3.4.4}$$

holds with probability at least $1 - 2\exp(-\lambda^2/2)$.

Proof. Let $E_k = \sum_{i=1}^{k} \rho_i \delta_i$ for $k = 1\colon n$ and $E_0 = 0$. Since $|E_k| \leqslant ku$, clearly $\mathbb{E}(|E_k|) < \infty$. Moreover, since $E_{k+1} = E_k + \rho_{k+1}\delta_{k+1}$,

$$\mathbb{E}(E_{k+1} \mid E_1, \dots, E_k) = E_k + \rho_{k+1}\mathbb{E}(\delta_{k+1} \mid \delta_1, \dots, \delta_k) = E_k.$$

Therefore $E_0, \dots, E_n$ is a martingale. Since $|E_{k+1} - E_k| \leqslant u$, Lemma 3.9 yields

$$|E_n - E_0| = |E_n| \leqslant \lambda\sqrt{n}u \tag{3.4.5}$$

with probability at least $1-2\exp(-\lambda^2/2)$. By a Taylor expansion it can be shown that [16, eqn. (2.3)]

$$\delta_i - \frac{u^2}{1-u} \leqslant \log(1+\delta_i) \leqslant \delta_i + \frac{u^2}{1-u}.$$

Hence, for $\rho_i = \pm 1$,

$$\rho_i\delta_i - \frac{u^2}{1-u} \leqslant \rho_i \log(1+\delta_i) \leqslant \rho_i\delta_i + \frac{u^2}{1-u}.$$

Summing gives

$$E_n - \frac{nu^2}{1-u} \leqslant \log \prod_{i=1}^{n}(1+\delta_i)^{\rho_i} \leqslant E_n + \frac{nu^2}{1-u},$$

which by (3.4.5) can be weakened to

$$-\left(\lambda\sqrt{n}u + \frac{nu^2}{1-u}\right) \leqslant \log \prod_{i=1}^{n}(1+\delta_i)^{\rho_i} \leqslant \lambda\sqrt{n}u + \frac{nu^2}{1-u}.$$

We slightly weaken this bound further by dividing the $\lambda\sqrt{n}u$ terms by $1-u$ on each side, and then we exponentiate to obtain

$$1 - \frac{\widetilde{\gamma}_n(\lambda)}{1+\widetilde{\gamma}_n(\lambda)} = \frac{1}{1+\widetilde{\gamma}_n(\lambda)} \leqslant \prod_{i=1}^{n}(1+\delta_i)^{\rho_i} \leqslant 1 + \widetilde{\gamma}_n(\lambda).$$

From the definition of θ_n, we therefore have $|\theta_n| \leqslant \widetilde{\gamma}_n(\lambda)$. □

Theorem 3.10 can now be used to derive analogues of the probabilistic backward error results from [16] for inner products, matrix–vector and matrix–matrix products, LU factorization, Cholesky factorization, solution of triangular systems, and solution of linear systems by LU factorization or Cholesky factorization. In all cases the assumption that the rounding errors are independent random variables can be weakened to an assumption of mean independence. To be precise, we define the following model of rounding errors in a given computation.

Model 3.11 (probabilistic model of rounding errors)**.** *Let the computation of interest generate rounding errors $\delta_1, \delta_2, \ldots$ in that order. The δ_k are random variables of mean zero such that $\mathbb{E}(\delta_k \mid \delta_1, \ldots, \delta_{k-1}) = \mathbb{E}(\delta_k)$ $(= 0)$.*

As an example, we write down the results for inner products, matrix–matrix products, and solution of linear systems.

Theorem 3.12 (inner products). *Let $y = a^T b$, where $a, b \in \mathbb{R}^n$, be evaluated in floating-point arithmetic. Under Model 3.11, no matter what the order of evaluation the computed $\widehat{y}$ satisfies*

$$\widehat{y} = (a + \Delta a)^T b = a^T (b + \Delta b), \quad |\Delta a| \leqslant \widetilde{\gamma}_n(\lambda)|a|, \ |\Delta b| \leqslant \widetilde{\gamma}_n(\lambda)|b| \tag{3.4.6}$$

with probability at least $Q(\lambda, n)$.

Proof. The proof is almost identical to that of [16, Thm. 3.1], the difference being that we invoke Theorem 3.10 instead of Lemma 3.7. □

The next two results are analogues of [16, Thms. 3.4, 3.7].

Theorem 3.13 (matrix–matrix products). *Let $C = AB$ with $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times p}$. Under Model 3.11, the jth column of the computed $\widehat{C}$ satisfies*

$$\widehat{c}_j = (A + \Delta A_j) b_j, \quad |\Delta A_j| \leqslant \widetilde{\gamma}_n(\lambda)|A|, \quad j = 1\colon n, \tag{3.4.7}$$

with probability at least $Q(\lambda, mn)$, and hence

$$|C - \widehat{C}| \leqslant \widetilde{\gamma}_n(\lambda)|A||B| \tag{3.4.8}$$

with probability at least $Q(\lambda, mnp)$.

Theorem 3.14 (linear system). *Let $A \in \mathbb{R}^{n\times n}$ and suppose that LU factorization and substitution produce computed factors $\widehat{L}$ and $\widehat{U}$ and a computed solution $\widehat{x}$ to $Ax = b$. Then, under Model 3.11,*

$$(A + \Delta A)\widehat{x} = b, \quad |\Delta A| \leqslant \left(3\widetilde{\gamma}_n(\lambda) + \widetilde{\gamma}_n(\lambda)^2\right)|\widehat{L}||\widehat{U}| \tag{3.4.9}$$

holds with probability at least $Q(\lambda, n^3/3 + 3n^2/2 + 7n/6)$.

3.5 BACKWARD ERROR ANALYSIS FOR STOCHASTIC ROUNDING

Now we focus on stochastic rounding, with the aim of showing that the analysis of the previous section is applicable, that is, that stochastic rounding satisfies Model 3.11. Throughout this section stochastic rounding means mode 1 stochastic rounding. In all our analysis the data is assumed to be deterministic.

We first show that stochastic rounding forces the rounding errors to be random variables with zero mean.

Lemma 3.15. *For* $x \in \mathbb{R}$, *if* $y = \mathrm{fl}(x) = x(1+\delta)$ *is produced by stochastic rounding then* δ *is a random variable with* $\mathbb{E}(\delta) = 0$.

Proof. Recall the definition (3.1.1) of stochastic rounding:

$$\mathrm{fl}(x) = \begin{cases} \lceil x \rceil \text{ with probability } p = (x - \lfloor x \rfloor)/(\lceil x \rceil - \lfloor x \rfloor), \\ \lfloor x \rfloor \text{ with probability } 1 - p. \end{cases}$$

Clearly, $\mathrm{fl}(x)$ and $\delta = (\mathrm{fl}(x) - x)/x$ are random variables. We have

$$\mathbb{E}(\mathrm{fl}(x)) = p\lceil x \rceil + (1-p)\lfloor x \rfloor = \frac{(x - \lfloor x \rfloor)\lceil x \rceil + (\lceil x \rceil - x)\lfloor x \rfloor}{\lceil x \rceil - \lfloor x \rfloor} = x.$$

Then $\mathbb{E}(\delta) = \mathbb{E}((\mathrm{fl}(x) - x)/x) = 0$. □

It is important to note that mode 2 stochastic rounding does not produce a zero mean: (3.1.2) implies that $\mathbb{E}(\mathrm{fl}(x)) = (\lceil x \rceil + \lfloor x \rfloor)/2$, which is in general not equal to x.

Lemma 3.7, and the analysis of [16], require independence of rounding errors. The question therefore arises: does stochastic rounding enforce independence of rounding errors? The answer is negative. Indeed, successive rounding errors are still dependent on each other since they affect the computed values. Consider for example the computation of $(a+b)+c$. We have

$$\mathrm{fl}(\mathrm{fl}(a+b)+c) = ((a+b)(1+\delta_1)+c)(1+\delta_2).$$

Clearly, δ_2 depends on the addends $(a+b)(1+\delta_1)$ and c and hence on δ_1. This simple example shows that independence of rounding errors is not enforced by stochastic rounding. However, stochastic rounding does enforce mean independence of the rounding errors.

Lemma 3.16. *Let the computation of interest generate rounding errors $\delta_1, \delta_2, \ldots,$ in that order. If stochastic rounding is used then the δ_k satisfy Model 3.11.*

Proof. We know by Lemma 3.15 that the rounding errors have mean zero. It suffices to consider quantities a and b resulting from the computation of $k-1$ scalar operations that have produced rounding errors $\delta_1, \ldots, \delta_{k-1}$. Consider now the computation of $c = a \,\mathrm{op}\, b$ for any scalar operation $\mathrm{op} \in \{+, -, *, /, \sqrt{}\}$, resulting in $\widehat{c} = \mathrm{fl}(a \,\mathrm{op}\, b) = (a \,\mathrm{op}\, b)(1+\delta_k)$. The rounding error $\delta_k = (\widehat{c}-c)/c$ is a random variable that depends on $\delta_1, \ldots, \delta_{k-1}$ and is given by

$$\delta_k = \begin{cases} (\lceil c \rceil - c)/c \text{ with probability } p = (c - \lfloor c \rfloor)/(\lceil c \rceil - \lfloor c \rfloor), \\ (\lfloor c \rfloor - c)/c \text{ with probability } 1-p. \end{cases} \tag{3.5.1}$$

Moreover, $(\lceil c \rceil - c)/c$ and $(\lfloor c \rfloor - c)/c$ are themselves random variables that are entirely determined by $\delta_1, \ldots, \delta_{k-1}$ and so the conditional expectation of each given $\delta_1, \ldots, \delta_{k-1}$ is itself. Therefore we obtain

$$\begin{aligned}\mathbb{E}(\delta_k \mid \delta_1, \ldots, \delta_{k-1}) &= p\mathbb{E}\left(\frac{\lceil c \rceil - c}{c} \mid \delta_1, \ldots, \delta_{k-1}\right) + (1-p)\mathbb{E}\left(\frac{\lfloor c \rfloor - c}{c} \mid \delta_1, \ldots, \delta_{k-1}\right) \\ &= p\left(\frac{\lceil c \rceil - c}{c}\right) + (1-p)\left(\frac{\lfloor c \rfloor - c}{c}\right) = 0.\end{aligned}$$

□

Since we have proven in Lemmas 3.15 and 3.16 that the rounding errors δ_i produced by stochastic rounding satisfy the assumptions of Theorem 3.10, we conclude that the probabilistic bound (3.4.4) holds unconditionally for them (with $u \leftarrow 2u$ in view of (3.2.4)), without exception. Hence for stochastic rounding the rule of thumb that one can replace nu in a worst-case error bound by $\sqrt{n}u$ to obtain a more realistic (probabilistic) error bound is *unconditionally true*. Furthermore, the backward error bounds in Theorems 3.12– 3.14 hold unconditionally for stochastic rounding as long as we replace u by $2u$ in $\widetilde{\gamma}(\lambda)$ in (3.4.2).

3.6 THE MEAN OF THE ERROR FOR STOCHASTIC ROUNDING

We now ask what is the expected value of the computed result for stochastic rounding. Since the result from a computation with stochastic rounding has a random error, which is generally different each time the computation is repeated, it is intuitively desirable that the expected value of the computed result is the true result. We focus on mode 1 stochastic rounding since, as we noted in the previous section, for mode 2 this property does not hold.

For a single floating-point operation we know that the expected value is the true value by Lemma 3.15, because $\mathbb{E}(1+\delta) = 1$ for a single rounding error δ. In the next result we show that a product of rounding error terms also has expected value 1. The key property needed is mean independence.

Lemma 3.17. *Let $\delta_1, \delta_2, \dots, \delta_n$ be random variables of mean zero such that $\mathbb{E}(\delta_{k+1} \mid \delta_1, \dots, \delta_k) = \mathbb{E}(\delta_{k+1}) = 0$ for $k = 1\colon n-1$. Then*

$$\mathbb{E}\left(\prod_{i=1}^{n}(1+\delta_i)\right) = 1.$$

Proof. Define $P_n = \prod_{i=1}^{n}(1+\delta_i)$. We prove $\mathbb{E}(P_n) = 1$ by induction. The result clearly holds for P_1 since $E(1+\delta_1) = 1$. Assume it holds for P_{n-1}. Using the law of total expectation (or tower property) $\mathbb{E}(X) = \mathbb{E}\big(\mathbb{E}(X \mid Y)\big)$ [3, p. 448], [45, p. 401], we have

$$\begin{aligned}
\mathbb{E}(P_n) &= \mathbb{E}(\mathbb{E}(P_n \mid \delta_1, \dots, \delta_{n-1})) \\
&= \mathbb{E}(\mathbb{E}(P_{n-1}(1+\delta_n) \mid \delta_1, \dots, \delta_{n-1})) \\
&= \mathbb{E}(P_{n-1}\mathbb{E}(1+\delta_n \mid \delta_1, \dots, \delta_{n-1})) \\
&= \mathbb{E}(P_{n-1}) = 1,
\end{aligned}$$

and the result follows by induction. □

We note that Lemma 3.17 does not generalize to the product $\prod_{i=1}^{n}(1+\delta_i)^{\rho_i}$ with $\rho_i = \pm 1$. We apply the lemma to inner products.

Theorem 3.18 (inner products). *Let $y = a^T b$, where $a, b \in \mathbb{R}^n$, be evaluated in floating-point arithmetic. Under stochastic rounding, no matter what the order of evaluation the computed $\widehat{y}$ satisfies $\mathbb{E}(\widehat{y}) = y$.*

Proof. Standard backward error analysis [15, sect. 3.1] shows that $\widehat{y}$ can be written as

$$\widehat{y} = \sum_{i=1}^{n} a_i b_i \prod_{k=1}^{n}(1+\delta_{k_i}),$$

where the δ_{k_i} satisfy (3.2.4b). (Some of the δ_{k_i} will be zero, depending on the order in which the inner product is evaluated). Taking the mean and using Lemma 3.17, along with the fact that the rounding errors from stochastic rounding are mean independent with zero mean by Lemma 3.16, we obtain $\mathbb{E}(\widehat{y}) = \sum_{i=1}^{n} a_i b_i = y$. □

As a special case of Theorem 3.18 we have that the expected value of a sum is the exact sum under stochastic rounding. We have a similar result for matrix multiplication (and, as a special case, matrix–vector products).

Theorem 3.19 (matrix multiplication). *Let $C = AB$, where $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times p}$, be evaluated in floating-point arithmetic. Under stochastic rounding, no matter what the order of evaluation the computed $\widehat{C}$ satisfies $\mathbb{E}(\widehat{C}) = C$.*

Proof. The result is obtained by applying Theorem 3.18 to the inner products $c_{ij} = A(i,:)B(:,j)$. □

Theorems 3.18 and 3.19 do not, of course, hold for round to nearest, because it is deterministic.

This argument extends to the solution of triangular systems, as we now show. We need an extension of Lemma 3.17.

Lemma 3.20. *Let $\delta_{-m}, \dots, \delta_0, \delta_1, \delta_2, \dots, \delta_n$ be random variables of mean zero such that $\mathbb{E}(\delta_k \mid \delta_{-m}, \dots, \delta_{k-1}) = \mathbb{E}(\delta_k) = 0$ for $k = 1\colon n$. Then*

$$\mathbb{E}\left(\prod_{i=1}^{n}(1+\delta_i) \;\middle|\; \delta_0, \dots, \delta_{-m} \right) = 1.$$

Proof. Define

$$p_n = \mathbb{E}\left(\prod_{i=1}^{n}(1+\delta_i) \;\middle|\; \delta_{-m}, \dots, \delta_0 \right).$$

We prove by induction that $p_n = 1$. We have

$$p_1 = \mathbb{E}\big(1 + \delta_1 \mid \delta_{-m}, \dots, \delta_0\big) = 1 + \mathbb{E}\big(\delta_1 \mid \delta_{-m}, \dots, \delta_0\big) = 1.$$

Assume that $p_{n-1} = 1$. Using the general form of the law of total expectation, $\mathbb{E}(X \mid Y) = \mathbb{E}(\mathbb{E}(X \mid Z) \mid Y)$ where "$Y \subseteq Z$" [3, Thm. 34.4], we have

$$\begin{aligned}
p_n &= \mathbb{E}\left(\prod_{i=1}^{n}(1+\delta_i) \;\middle|\; \delta_{-m}, \dots, \delta_0\right) \\
&= \mathbb{E}\left(\mathbb{E}\left(\prod_{i=1}^{n}(1+\delta_i) \;\middle|\; \delta_{-m}, \dots, \delta_{n-1}\right) \;\middle|\; \delta_{-m}, \dots, \delta_0\right) \\
&= \mathbb{E}\left(\prod_{i=1}^{n-1}(1+\delta_i)\mathbb{E}\big(1+\delta_n \mid \delta_{-m}, \dots, \delta_{n-1}\big) \;\middle|\; \delta_{-m}, \dots, \delta_0\right) \\
&= p_{n-1} = 1,
\end{aligned}$$

so the result follows by induction. □

Theorem 3.21. *Let the triangular system $Tx = b$, where $T \in \mathbb{R}^{n\times n}$ is nonsingular, be solved by substitution with stochastic rounding. The computed solution $\widehat{x}$ satisfies $\mathbb{E}(\widehat{x}) = x$.*

Proof. Assume that T is lower triangular without loss of generality. We prove that $\mathbb{E}(\widehat{x}_i) = x_i$ by induction. From (3.2.4b) and Lemma 3.15, we have

$$\mathbb{E}(\widehat{x}_1) = \mathbb{E}\left(\frac{b_1}{t_{11}}(1+\delta_1^{(1)})\right) = \frac{b_1}{t_{11}} = x_1.$$

Assume that $\mathbb{E}(\widehat{x}_j) = x_j$ for all $j < i$. We compute x_i from

$$x_i = \Big(b_i - \sum_{j=1}^{i-1} t_{ij}x_j\Big)/t_{ii}. \tag{3.6.1}$$

There are $2i-1$ rounding errors in total and no term in (3.6.1) is involved in more than $i+1$ rounding errors. Using (3.2.4b), and proceeding as in standard backward

error analysis [15, sect. 8.1], we find that no matter what the order of evaluation of (3.6.1), we can write

$$t_{ii}\widehat{x}_i = b_i \prod_{k=1}^{i+1}(1+\delta_{i,k}^{(i)}) - \sum_{j=1}^{i-1} t_{ij}\widehat{x}_j \prod_{k=1}^{i+1}(1+\delta_{j,k}^{(i)}),$$

where the $\delta_{j,k}^{(i)}$ are drawn from the $2i-1$ rounding errors and some of the $\delta_{j,k}^{(i)}$ are zero, depending on the order of evaluation. Therefore we obtain

$$\mathbb{E}(t_{ii}\widehat{x}_i) = b_i \mathbb{E}\Bigl(\prod_{k=1}^{i+1}(1+\delta_{i,k}^{(i)})\Bigr) - \sum_{j=1}^{i-1} t_{ij}\mathbb{E}\Bigl(\widehat{x}_j \prod_{k=1}^{i+1}(1+\delta_{j,k}^{(i)})\Bigr). \tag{3.6.2}$$

The first expectation term in this equation is equal to 1 by Lemmas 3.16 and 3.17. We need to show that the second expectation term is also 1, which is not immediate because $\widehat{x}_j$ is not constant (it depends on the previous rounding errors, which are random). To prove the result, we use the law of total expectation to condition on all the rounding errors upon which $\widehat{x}_j$ depends. Let

$$S_j = \{\, (p,\ell,m) : p = 1\colon j,\ \ell = 1\colon p,\ m = 1\colon p+1 \,\}.$$

We have

$$\begin{aligned}
\mathbb{E}\Bigl(\widehat{x}_j \prod_{k=1}^{i+1}(1+\delta_{j,k}^{(i)})\Bigr) &= \mathbb{E}\Bigl(\mathbb{E}\Bigl(\widehat{x}_j \prod_{k=1}^{i+1}(1+\delta_{j,k}^{(i)}) \;\Big|\; \{\, \delta_{\ell,m}^{(p)} : (p,\ell,m) \in S_j \,\}\Bigr)\Bigr) \\
&= \mathbb{E}\Bigl(\widehat{x}_j \mathbb{E}\Bigl(\prod_{k=1}^{i+1}(1+\delta_{j,k}^{(i)}) \;\Big|\; \{\, \delta_{\ell,m}^{(p)} : (p,\ell,m) \in S_j \,\}\Bigr)\Bigr) \\
&= \mathbb{E}(\widehat{x}_j) = x_j,
\end{aligned}$$

where the penultimate equality follows from Lemma 3.20, which is applicable by Lemma 3.16 and since the rounding errors $\delta_{j,k}^{(i)}$ occur later than the rounding errors $\{\, \delta_{\ell,m}^{(p)} : (p,\ell,m) \in S_j \,\}$ for all $j < i$. Equation (3.6.2) now gives

$$t_{ii}\mathbb{E}(\widehat{x}_i) = b_i - \sum_{j=1}^{i-1} t_{ij}x_j = t_{ii}x_i,$$

so $\mathbb{E}(\widehat{x}_i) = x_i$. The result follows by induction. □

These results do not extend to matrix factorizations and the solution of general linear systems by LU factorization. The reason is that such kernels involve divisions by computed quantities, which leads to a nonzero mean error because $\mathbb{E}(1/X) \neq 1/\mathbb{E}(X)$. For example, the Doolittle form of LU factorization [15, sec. 9.2] gives the following recurrence for the lower triangular factor:

$$\ell_{ik} = \left(a_{ik} - \sum_{j=1}^{k-1} \widehat{\ell}_{ij}\widehat{u}_{jk} \right) / \widehat{u}_{kk},$$

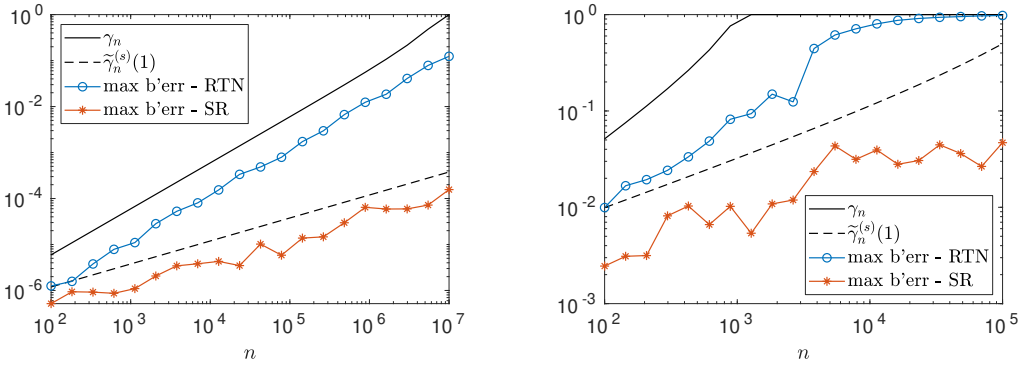
where the division by the computed $\widehat{u}_{kk}$ prevents the mean of the computed ℓ_{ik} equaling ℓ_{ik}.

3.7 NUMERICAL EXPERIMENTS

We present a set of numerical experiments to verify that mode 1 stochastic rounding obeys the probabilistic bound $\widetilde{\gamma}_n^{(s)}(\lambda)$ in (3.4.6) for inner products without fail, even when the bound does not hold for round to nearest. To that end, we revisit the numerical experiments of [16], which give two examples where the bound is violated with round to nearest.

We use the implementation of stochastic rounding provided in the MATLAB function `chop`[3] [18]. The computations are performed in MATLAB R2019b. The precisions used are half precision (fp16) and single precision (fp32). Reference solutions used in backward error formulas are computed in double precision (fp64).

3 https://github.com/higham/chop



(a) fp32. **(b)** fp16.

Figure 3.7.1: Computed backward errors of inner products for random constant vectors in (a) fp32 and (b) fp16. For each value of n we perform the computation 10 times and plot the maximum backward error for round to nearest (RTN) and stochastic rounding (SR).

In Figure 3.7.1 we plot the backward error for the inner product of two random vectors with constant entries, the two constants being sampled uniformly from $[0, 1]$. We also plot γ_n and

$$\widetilde{\gamma}_n^{(s)}(\lambda) = \exp\left(\frac{2\lambda\sqrt{n}u + 4nu^2}{1-2u}\right) - 1 = 2\lambda\sqrt{n}u + O(u^2),$$

which is $\widetilde{\gamma}_n(\lambda)$ in (3.4.2) with u replaced by $2u$. We take $\lambda = 1$, as in the experiments of [16]. With round to nearest, the error does not satisfy the bound (3.4.6), which is proportional to $\sqrt{n}u$, but rather grows proportionally to nu. As explained in [16], since the entries in each vector are constant, so too are the rounding errors within intervals of consecutive powers of 2. For round to nearest we thus have

$$\mathbb{E}(\delta_{k+1} \mid \delta_1, \ldots, \delta_k) = \delta_{k+1} \neq 0$$

for any δ_{k+1} unless it is the first rounding error incurred within the current interval of consecutive powers of two. Therefore the rounding errors are clearly not mean independent. However, stochastic rounding avoids producing constant rounding errors by randomizing them, and it thereby yields a much smaller error that satisfies the probabilistic bound (3.4.6).

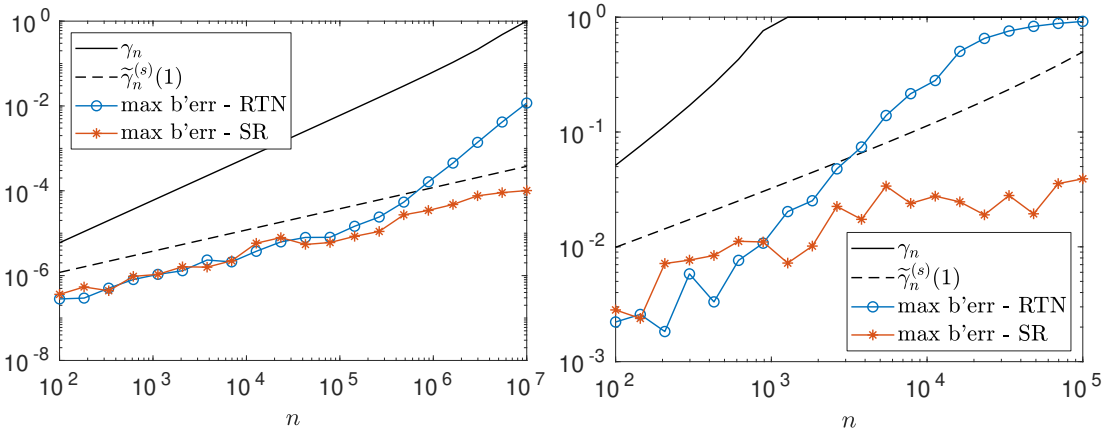
The second example displays the phenomenon of stagnation [4], [16]. It arises when summing a large number of terms of identical sign (positive, say). Consider, for example, the following recursive summation algorithm to compute the sum $s = \sum_{i=1}^{n} x_i$ of nonnegative x_i.

$s \leftarrow x_1$
for $i = 2 : n$ **do**
    $s \leftarrow s + x_i$
end for

Since the x_i are all nonnegative, the sum s grows monotonically with i. At some point, the sum becomes so large that the spacing ψ of floating-point numbers around s becomes larger than the x_i. Specifically, if the x_i are less than $\psi/2$, then with round to nearest the computed sum absorbs the x_i and no longer grows, that is, $\widehat{s}_{i+1} = \widehat{s}_i$. This leads to necessarily negative rounding errors, which therefore causes the error to start growing as nu rather than $\sqrt{n}u$. This stagnation is especially critical when using low precisions, since it can occur even for moderate values of n.

Figure 3.7.2 illustrates this phenomenon with the inner product of two vectors with random entries uniformly sampled in $[0, 1]$. With round to nearest, stagnation occurs for $n \gtrsim 10^6$ in single precision and for $n \gtrsim 10^4$ in half precision. Stochastic rounding does not suffer stagnation and is able to maintain an error growth bounded by $\sqrt{n}u$. This is because stochastic rounding allows the sum to continue growing: indeed, each increment has a small probability of increasing the sum to the next floating-point number, and statistically for a large number of increments the sum averages out to the exact sum. Theorem 3.18 makes this argument rigorous: the expected value of the computed inner product is the exact inner product.

In conclusion, these two examples illustrate that even in situations where round to nearest leads to rounding errors violating the assumptions required for the probabilistic bound (3.4.4) to hold, stochastic rounding still enforces these assumptions. Stochastic rounding can therefore produce significantly more accurate results than

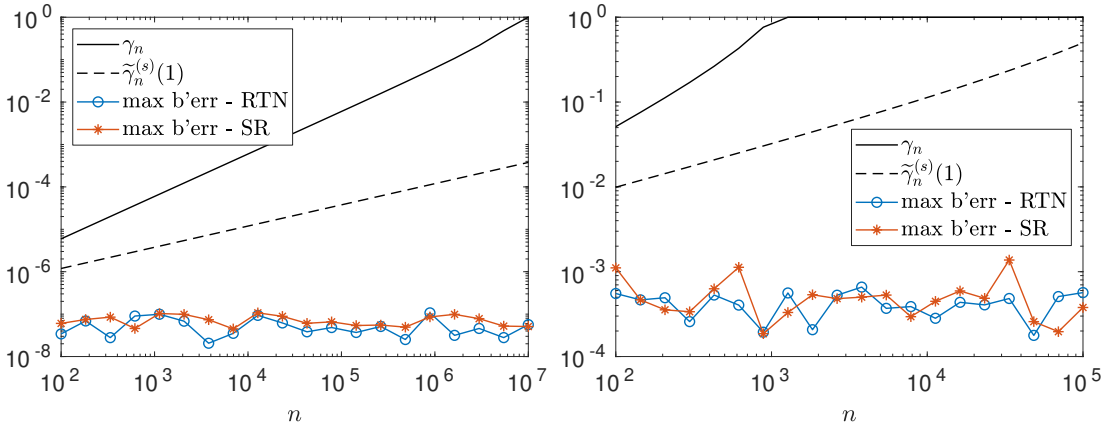


(a) fp32. **(b)** fp16.

Figure 3.7.2: Computed backward errors of inner products for data sampled from $[0, 1]$ in (a) fp32 and (b) fp16. For each value of n we perform the computation 10 times and plot the maximum backward error for each rounding mode.

round to nearest by reducing the error from nu to $\sqrt{n}u$. In particular, this explains the improvements from using stochastic rounding reported in deep learning applications.

The two examples above are bad cases for round to nearest. Figure 3.7.3 shows the results of an experiment with inner products of vectors x and y with elements from the uniform distribution on $[-1, 1]$. In this case the errors for stochastic rounding and round to nearest do not grow with n and so are both much less than the probabilistic error bound. The reason the errors do not grow is that the elements of x and y have mean zero, which means we obtain a backward error of $O(u)$ rather than $O(\sqrt{n}u)$. [17, Thm. 3.2]. Overall, round to nearest provides slightly more accurate results than stochastic rounding in this example, as might be expected in view of (3.2.3) and (3.2.4b).



(a) fp32. **(b)** fp16.

Figure 3.7.3: Computed backward errors of inner products for data sampled uniformly from $[-1, 1]$ in fp32 (a) and fp16 (b). For each value of n we perform the computation 10 times and plot the maximum backward error for each rounding mode.

3.8 CONCLUSIONS

Stochastic rounding is an old idea that is drawing renewed interest, notably in the context of deep learning. We have presented rounding error analyses applicable to a wide range of numerical linear algebra algorithms using floating-point arithmetic with stochastic rounding, and we expect our conclusions to extend to fixed-point arithmetic.

Stochastic rounding satisfies the basic model of floating-point arithmetic (3.2.3), provided that the unit roundoff u is replaced by $2u$; see (3.2.4b). However, we have identified several properties of round to nearest that no longer hold with stochastic rounding. Before replacing round to nearest by stochastic rounding in a computation one should therefore check whether these properties are needed.

Stochastic rounding has some attractive features compared with round to nearest, especially for large problems and low precisions. We have shown that stochastic rounding has the property that the rounding errors it produces are mean independent. We have also generalized the probabilistic error analysis result of [16] (Lemma 3.7 here) by weakening the independence assumption to mean indepen-

dence (Theorem 3.10). An important consequence of these results is that for stochastic rounding a worst-case error bound nu can be replaced by the more realistic probabilistic error bound $\sqrt{n}u$—that is, the long-standing rule of thumb is actually a *rule* for stochastic rounding.

Stochastic rounding can yield significantly more accurate results than round to nearest in the situations where the latter violates the probabilistic bounds, notably in certain sums and inner products. In particular, we have proved that stochastic rounding avoids stagnation and that the computed result has expected value equal to the exact sum. These findings are particularly important for deep learning applications, where stagnation can hamper parameter updates in neural networks.

REFERENCES

[1] *Arm A64 Instruction Set Architecture Armv8, for Armv8-A Architecture Profile*. Cambridge, UK: ARM Limited, 2019, p. 2742 (cited on p. 52).

[2] R. C. M. Barnes, E. H. Cooke-Yarborough, and D. G. A. Thomas. "An electronic digital computor using cold cathode counting tubes for storage." *Electronic Eng.* 23 (Aug. 1951), pp. 286–291 (cited on p. 48).

[3] P. Billingsley. *Probability and Measure*. Third. New York: Wiley, 1995 (cited on pp. 68, 70).

[4] P. Blanchard, N. J. Higham, and T. Mary. "A class of fast and accurate summation algorithms." *SIAM J. Sci. Comput.* 42.3 (Jan. 2020), A1541–A1557 (cited on p. 74).

[5] M.-C. Brunet and F. Chatelin. "CESTAC, a Tool for a Stochastic Round-off Error Analysis in Scientific Computing." In: *Numerical Mathematics and Applications*. Ed. by R. Vichnevetsky and J. Vignes. Elsevier Science Publishers B.V. (North-Holland), Amsterdam, The Netherlands, 1986, pp. 11–20 (cited on p. 49).

[6] M. Courbariaux, Y. Bengio, and J.-P. David. "BinaryConnect: Training Deep Neural Networks with Binary Weights During Propagations." In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. Curran Associates, Inc., 2015, pp. 3123–3131 (cited on p. 48).

[7] A. Edelman. "When is $x * (1/x) \neq 1$?" 1994. Manuscript (cited on p. 54).

[8] M. Essam, T. B. Tang, E. T. W. Ho, and H. Chen. "Dynamic Point Stochastic Rounding Algorithm for Limited Precision Arithmetic in Deep Belief Network Training." In: *2017 8th International IEEE/EMBS Conference on Neural Engineering (NER)*. May 2017, pp. 629–632 (cited on p. 48).

[9] G. E. Forsythe. "Reprint of a note on rounding-off errors." *SIAM Rev.* 1.1 (1959), pp. 66–67 (cited on p. 48).

[10] G. E. Forsythe. "Round-off errors in numerical integration on automatic machinery-preliminary report." *Bull. Amer. Math. Soc.* 56.1 (1950), pp. 61–62. Abstract (cited on p. 48).

[11] D. Goldberg. "What every computer scientist should know about floating-point arithmetic." *ACM Computing Surveys* 23.1 (Jan. 1991), pp. 5–48 (cited on pp. 50, 54, 55).

[12] S. Graillat, F. Jézéquel, and R. Picot. "Numerical validation of compensated algorithms with stochastic arithmetic." *Appl. Math. Comput.* 329 (2018), pp. 339–363 (cited on pp. 49, 58).

[13] S. Graillat, F. Jézéquel, and R. Picot. "Numerical validation of compensated summation algorithms with stochastic arithmetic." *Electron. Notes Theor. Comput. Sci.* 317 (2015), pp. 55–69 (cited on pp. 49, 58).

[14] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan. "Deep Learning with Limited Numerical Precision." In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France: JMLR.org, 2015, pp. 1737–1746 (cited on p. 48).

[15] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. Second. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2002, pp. xxx+680 (cited on pp. 50–54, 56, 57, 59, 68, 71, 72).

[16] N. J. Higham and T. Mary. "A new approach to probabilistic rounding error analysis." *SIAM J. Sci. Comput.* 41.5 (2019), A2815–A2835 (cited on pp. 49, 50, 60, 61, 63, 64, 66, 72–74, 76).

[17] N. J. Higham and T. Mary. "Sharper probabilistic backward error analysis for basic linear algebra kernels with random data." *SIAM J. Sci. Comput.* 42.5 (2020), A3427–A3446 (cited on pp. 61, 75).

[18] N. J. Higham and S. Pranesh. "Simulating low precision floating-point arithmetic." *SIAM J. Sci. Comput.* 41.5 (2019), pp. C585–C602 (cited on p. 72).

[19] M. Hopkins, M. Mikaitis, D. R. Lester, and S. Furber. "Stochastic rounding and reduced-precision fixed-point arithmetic for solving neural ordinary differential equations." *Phil. Trans. R. Soc. A* 378.2166 (2020), pp. 1–22 (cited on p. 49).

[20] T. E. Hull and J. R. Swenson. "Tests of probabilistic models for propagation of roundoff errors." *Comm. ACM* 9.3 (1966), pp. 108–113 (cited on p. 48).

[21] *IEEE Standard for Floating-Point Arithmetic, IEEE Std 754-2019 (Revision of IEEE 754-2008)*. New York: IEEE Computer Society, 2019, p. 82 (cited on pp. 47, 51).

[22] Intel Corporation. *BFLOAT16—Hardware Numerics Definition*. Nov. 2018. White paper. Document number 338302-001US (cited on p. 51).

[23] I. C. F. Ipsen and H. Zhou. "Probabilistic error analysis for inner products." *SIAM J. Matrix Anal. Appl.* 41.4 (Jan. 2020), pp. 1726–1741 (cited on p. 61).

[24] C.-P. Jeannerod and S. M. Rump. "Improved error bounds for inner products in floating-point arithmetic." *SIAM J. Matrix Anal. Appl.* 34.2 (2013), pp. 338–344 (cited on p. 59).

[25] F. Jézéquel and J.-M. Chesneaux. "CADNA: A library for estimating round-off error propagation." *Comput. Phys. Comm.* 178.12 (2008), pp. 933–955 (cited on p. 49).

[26] N. Mellempudi, S. Srinivasan, D. Das, and B. Kaul. *Mixed Precision Training With 8-bit Floating Point*. arXiv:1905.12334. May 2019 (cited on p. 48).

[27] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. 2nd. USA: Cambridge University Press, 2017 (cited on p. 62).

[28] J.-M. Muller, N. Brunie, F. de Dinechin, C.-P. Jeannerod, M. Joldes, V. Lefèvre, G. Melquiond, N. Revol, and S. Torres. *Handbook of Floating-Point Arithmetic*. Second. Boston, MA, USA: Birkhäuser, 2018, pp. xxv+627 (cited on p. 50).

[29] T. Na, J. H. Ko, J. Kung, and S. Mukhopadhyay. "On-Chip Training of Recurrent Neural Networks with Limited Numerical Precision." In: *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 3716–3723 (cited on p. 48).

[30] M. Ortiz, A. Cristal, E. Ayguadé, and M. Casas. *Low-Precision Floating-Point Schemes for Neural Network Training*. ArXiv:1804.05267. Apr. 2018 (cited on p. 48).

[31] D. S. Parker. *Monte Carlo Arithmetic: Exploiting Randomness in Floating-Point Arithmetic*. Technical Report CSD-970002. Computer Science Department, University of California (Los Angeles), 1997, p. 86 (cited on p. 49).

[32] D. S. Parker, B. Pierce, and P. R. Eggert. "Monte Carlo arithmetic: How to gamble with floating point and win." *Computing in Science and Engineering* 2.4 (July 2000), pp. 58–68 (cited on p. 49).

[33] S. M. Rump and C.-P. Jeannerod. "Improved backward error bounds for LU and Cholesky factorizations." *SIAM J. Matrix Anal. Appl.* 35.2 (2014), pp. 684–698 (cited on p. 59).

[34] N. S. Scott, F. Jézéquel, C. Denis, and J.-M. Chesneaux. "Numerical 'health check' for scientific codes: The CADNA approach." *Comput. Phys. Comm.* 176.8 (2007), pp. 507–521 (cited on p. 49).

[35] J. R. Shewchuk. "Adaptive precision floating-point arithmetic and fast robust geometric predicates". *Discrete Comput. Geom.* 18.3 (1997), pp. 305–363 (cited on p. 57).

[36] N. J. A. Sloane ed. *The On-Line Encyclopedia of Integer Sequences*. https://oeis.org (cited on p. 55).

[37] P. H. Sterbenz. *Floating-Point Computation*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1974, pp. xiv+316 (cited on p. 52).

[38] C. Su, S. Zhou, L. Feng, and W. Zhang. "Towards high performance low bitwidth training for deep neural networks." *Journal of Semiconductors* 41.2 (Feb. 2020), p. 022404 (cited on p. 48).

[39] J. Vignes. "New methods for evaluating the validity of the results of mathematical computations." *Math. Comput. Simulation* 20.4 (1978), pp. 227–249 (cited on p. 49).

[40] J. Vignes. "Discrete stochastic arithmetic for validating results of numerical software." *Numer. Algorithms* 37 (2004), pp. 377–390 (cited on p. 49).

[41] S. Vogel, C. Schorn, A. Guntoro, and G. Ascheid. *Efficient Stochastic Inference of Bitwise Deep Neural Networks*. ArXiv:1611.06539. Nov. 2016, p. 6 (cited on p. 48).

[42] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan. "Training Deep Neural Networks with 8-bit Floating Point Numbers." In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 7686–7695 (cited on p. 48).

[43] J. H. Wilkinson. "Error analysis of direct methods of matrix inversion." *J. ACM* 8.3 (July 1961), pp. 281–330 (cited on p. 49).

[44] J. H. Wilkinson. *Rounding Errors in Algebraic Processes*. London: Notes on Applied Science No. 32, Her Majesty's Stationery Office, 1963, pp. vi+161. Also published by Prentice-Hall, Englewood Cliffs, NJ, USA. Reprinted by Dover, New York, 1994 (cited on p. 49).

[45] D. Williams. *Weighing the Odds. A Course in Probability and Statistics*. Cambridge, UK: Cambridge University Press, 2001, pp. xvii+547 (cited on p. 68).

[46] S. Wu, G. Li, F. Chen, and L. Shi. "Training and Inference with Integers in Deep Neural Networks." In: *International Conference on Learning Representations*. 2018 (cited on p. 48).

4 PROBABILISTIC ROUNDING ERROR ANALYSIS OF HOUSEHOLDER QR FACTORIZATION

Abstract. When an $m \times n$ matrix is premultiplied by a product of n Householder matrices the worst-case normwise rounding error bound is proportional to mnu, where u is the unit roundoff. We prove that this bound can be replaced by one proportional to $\sqrt{mn}u$ that holds with high probability if the rounding errors are mean independent and of mean zero, under the assumption that a certain bound holds with probability 1. The proof makes use of a matrix concentration inequality. In particular, this result applies to Householder QR factorization. The same square rooting of the error constant applies to two-sided transformations by Householder matrices and hence to standard QR-type algorithms for computing eigenvalues and singular values. It also applies to Givens QR factorization. These results complement recent probabilistic rounding error analysis results for inner-product based algorithms and show that the square rooting effect is widespread in numerical linear algebra. Our numerical experiments, which make use of a new backward error formula for QR factorization, show that the probabilistic bounds give a much better indicator of the actual backward errors and their rate of growth than the worst-case bounds.

Keywords: floating-point arithmetic, backward error analysis, backward error, probabilistic rounding error analysis, Givens QR factorization, Householder QR factorization, matrix concentration inequality.

2010 MSC: 65G50, 65F05.

4.1 INTRODUCTION

It is well known that backward error bounds from worst-case rounding error analyses can greatly overestimate the backward error. Recently, it was proved that for inner product-based algorithms with backward error bounds of the form $f(n)u$, where n is the problem dimension and u is the unit roundoff, a bound proportional to $\sqrt{f(n)}u$ holds with high probability under suitable assumptions on the rounding errors [16]. These results apply to matrix multiplication, Cholesky factorization, LU factorization, and the solution of triangular systems, but not to algorithms based on orthogonal transformations.

A QR factorization of $A \in \mathbb{R}^{m\times n}$ ($m \geqslant n$) is a factorization $A = QR$ where $Q \in \mathbb{R}^{m\times m}$ is orthogonal and $R \in \mathbb{R}^{m\times n}$ is upper trapezoidal. A Householder matrix is a matrix of the form

$$P = I - \frac{2}{v^T v} vv^T, \quad 0 \neq v \in \mathbb{R}^m.$$

The vector v can be chosen so that in the product $y = Px$ all elements of y except the first are zero. By applying a sequence of such Householder matrices to A we can reduce it to upper trapezoidal form: $P_n \dots P_2 P_1 A = R$. We then have $Q = (P_n \dots P_2 P_1)^T$.

The standard rounding error analysis for Householder QR factorization is summarized in the following result [13, Thm. 19.4]. We define

$$\gamma_n = \frac{nu}{1-nu} \tag{4.1.1}$$

and use the 2-norm $\|x\|_2 = (x^T x)^{1/2}$, the corresponding subordinate matrix norm, and the Frobenius norm $\|A\|_F = \mathrm{trace}(A^T A)^{1/2}$. We denote the jth column of A by a_j. Throughout this work we express our bounds in terms of positive constants c_0, c_1, c_2, ..., which do not depend on u or any of the matrix dimensions.

Theorem 4.1. *Let $\widehat{R} \in \mathbb{R}^{m\times n}$ be the computed upper trapezoidal QR factor of $A \in \mathbb{R}^{m\times n}$ ($m \geqslant n$) obtained via the Householder QR algorithm, and assume that a condition of the form $c_0\gamma_{mn} < 1$ holds. There exists an orthogonal $Q \in \mathbb{R}^{m\times m}$ such that*

$$A + \Delta A = Q\widehat{R},$$

where

$$\|\Delta a_j\|_2 \leqslant c_1\gamma_{mn}\|a_j\|_2, \quad j = 1:n. \tag{4.1.2}$$

The columnwise bounds (4.1.2) yield the normwise backward error bound

$$\|\Delta A\|_F \leqslant c_1\gamma_{mn}\|A\|_F. \tag{4.1.3}$$

The purpose of this work is to show that the worst-case backward error bounds (4.1.2) and (4.1.3) for Householder QR factorization can be replaced by probabilistic bounds in which the dimension-dependent constants are proportional to the square roots of the original constants under suitable assumptions on the rounding errors; essentially, we can replace $c_1\gamma_{mn}$ by $c_1'\gamma_{\sqrt{mn}} + O(u^2)$. This is not an immediate consequence of the analysis for inner product-based algorithms, because the vector addition and scaling involved in applying Householder matrices need a different treatment.

We use the standard model of floating-point arithmetic, which assumes that the elementary operations are correctly rounded, so that

$$\mathrm{fl}(x \operatorname{op} y) = (x \operatorname{op} y)(1+\delta), \quad |\delta| \leqslant u. \tag{4.1.4}$$

In a rounding error analysis products of $1+\delta$ terms arise, and their distance from 1 can be bounded using the following lemma [13, Lem 3.1].

Lemma 4.2. *If $|\delta_i| \leqslant u$ and $\rho_i = \pm 1$ for $i = 1\!:\!n$, and $nu < 1$, then*

$$\prod_{i=1}^{n}(1+\delta_i)^{\rho_i} = 1+\theta_n, \quad |\theta_n| \leqslant \gamma_n. \tag{4.1.5}$$

An analogous probabilistic result [8, Thm. 4.6] is given in terms of the constant

$$\widetilde{\gamma}_n(\lambda) = \exp\left(\frac{\lambda\sqrt{n}u + nu^2}{1-u}\right) - 1 = \lambda\sqrt{n}u + O(u^2). \tag{4.1.6}$$

with $\lambda > 0$.

We need a definition.

Definition 4.1. Random variables $\delta_1, \delta_2, \ldots, \delta_n$ are *mean independent* if $\mathbb{E}(\delta_k \mid \delta_{k-1}, \ldots, \delta_1) = \mathbb{E}(\delta_k)$ for $k = 2\!:\!n$, where $\mathbb{E}$ denotes expectation.

Lemma 4.3. *Let $\delta_1, \delta_2, \ldots, \delta_n$ be mean independent random variables of mean zero such that $|\delta_i| \leqslant u$ for all i, and let $\rho_i = \pm 1$, $i = 1\!:\!n$. Then for any constant $\lambda > 0$,*

$$\prod_{i=1}^{n}(1+\delta_i)^{\rho_i} = 1+\theta_n, \quad |\theta_n| \leqslant \widetilde{\gamma}_n(\lambda), \tag{4.1.7}$$

holds with probability at least $p(\lambda) = 1 - 2\exp(-\lambda^2/2)$.

Throughout this work we will use the following model of rounding errors, which was also used in [8] and [17].

Model 4.4 (probabilistic model of rounding errors)**.** *Let the computation of interest generate rounding errors $\delta_1, \delta_2, \ldots$ (in that order) satisfying (4.1.4). The δ_k are mean independent random variables of mean zero.*

For the round to nearest rounding mode (the default in IEEE arithmetic) the error bounds obtained under Model 4.4 often reflect the behavior of the actual errors obtained, but in some circumstances the model is not valid because the rounding errors

are highly dependent or have nonzero mean, and the bound can then be violated [8, sec. 7], [9, sec. 8], [16, sec. 4.2]. For stochastic rounding [8], [9], the rounding errors *always* satisfy the model. For further discussion of the applicability of the model see [8], [16].

In our probabilistic rounding error analysis it is more difficult than in worst-case rounding error analysis to bound terms of all orders in u so we will state the bounds to first order in u. We will, however, show that the higher order terms do not change the conclusions of the analysis. We also give explicit bounds for the probabilities with which the error bounds hold, and these show that the analysis does not introduce exponential worsening of probabilities with the dimensions. All our probabilistic results can informally be stated as saying that the given backward error bound holds with λ a modest constant with high probability.

We begin in section 4.2 with a probabilistic analysis of the rounding errors in constructing a Householder vector. In section 4.3 we state a matrix concentration inequality that we use in section 4.4 in our probabilistic rounding error analysis of the application of products of Householder matrices to a matrix and of Householder QR factorization. In section 4.5 we discuss the implications of the analysis for two-sided transformations, Givens QR factorization, and other approaches. In section 4.6 we give a new backward error formula for QR factorization that we use in section 4.7, where we carry out numerical experiments to compare the results of the analysis with the practical behavior. Conclusions are given in section 4.8.

4.2 CONSTRUCTION OF A HOUSEHOLDER VECTOR

We first give a probabilistic error result for the construction of the Householder vectors needed in QR factorization. We begin by considering the evaluation of the vector 2-norm. We denote by $\mathrm{fl}(expr)$ the computed result of evaluating the expression *expr*.

Note that Model 4.4 assumes that (4.1.4) holds, so we are at liberty to use worst-case bounds within the probabilistic rounding error analysis, and we will do so for certain scalar operations.

Lemma 4.5. *Under Model 4.4, the computed 2-norm of $x \in \mathbb{R}^n$ satisfies*

$$\mathrm{fl}(\|x\|_2) = \|x\|_2(1+\eta), \quad |\eta| \leqslant c_1\widetilde{\gamma}_n(\lambda), \tag{4.2.1}$$

with probability at least $p_1(\lambda, n) = 1 - 2n\exp(-\lambda^2/2)$.

Proof. By standard inner-product analysis [8, Thm. 4.8], [16, Thm. 3.1],

$$\mathrm{fl}(x^Tx) = x^Tx(1+\theta_n), \quad |\theta_n| \leqslant \widetilde{\gamma}_n(\lambda), \tag{4.2.2}$$

holds with probability at least $p_1(\lambda, n)$. Then $\mathrm{fl}(\|x\|_2) = \|x\|_2\sqrt{1+\theta_n}(1+\delta)$, with $|\delta| \leqslant u$. For $\sqrt{1+\theta_n} = 1+\alpha$, we certainly have $|\alpha| \leqslant |\theta_n|$. Then $(1+\alpha)(1+\delta) = 1+\beta$, with $|\beta| \leqslant |\theta_n| + u + |\theta_n|u$. Hence provided $|\theta_n| \leqslant \widetilde{\gamma}_n(\lambda)$, we have $|\beta| \leqslant \widetilde{\gamma}_n(\lambda) + u + \widetilde{\gamma}_n(\lambda)u \leqslant c_1\widetilde{\gamma}_n(\lambda)$. The bound holds with probability at least the probability of (4.2.2) holding. □

We now derive a probabilistic version of the worst-case error bound for construction of a Householder vector [13, Lem. 19.1]. Here, $\mathrm{sign}(t) = -1$ if $t < 0$ or 1 if $t \geqslant 0$. We state the result for $x \in \mathbb{R}^n$, although strictly speaking $x \in F^n$, where $F \subset \mathbb{R}$ is a particular floating-point number system.

Lemma 4.6. *Let $x \in \mathbb{R}^n$. Consider the construction of $\beta \in \mathbb{R}$ and $v \in \mathbb{R}^n$ such that $Px = -\operatorname{sign}(x_1)\|x\|_2 e_1$, where $P = I - \beta vv^T$ is a Householder matrix and $\beta = 2/(v^T v)$, by*

$$
\begin{aligned}
v &= x,\\
s &= \operatorname{sign}(x_1)\|x\|_2,\\
w &= v_1 + s,\\
\beta &= 1/(sw),\\
v_1 &= w.
\end{aligned}
$$

Under Model 4.4 the computed $\widehat{\beta}$ and $\widehat{v}$ satisfy $\widehat{v}(2:n) = v(2:n)$ and

$$\widehat{v}_1 = v_1(1+\eta), \qquad \widehat{\beta} = \beta(1+\eta),$$

where $|\eta| \leqslant c_3\widetilde{\gamma}_n(\lambda) + O(u^2)$ with probability at least $p_1(\lambda, n)$.

Proof. From Lemma 4.5 we have that $\widehat{s} = s(1+\Delta s)$, where $|\Delta s| \leqslant c_1\widetilde{\gamma}_n(\lambda)$ holds with probability at least $p_1(\lambda, n)$. The rest of the computations, which determine w and β, are scalar operations and so we use worst-case bounds for these. We have

$$
\begin{aligned}
\widehat{w} &= (v_1 + \widehat{s})(1+\delta), \quad |\delta| \leqslant u\\
&= v_1 + s + \delta(v_1 + s) + s\Delta s(1+\delta) =: w + \Delta w_1.
\end{aligned}
$$

Because v_1 and s have the same sign, $|s| \leqslant |v_1 + s|$, and so

$$
\begin{aligned}
|\Delta w_1| &= \left|(v_1+s)\left(\delta + \frac{s}{v_1+s}\Delta s(1+\delta)\right)\right|\\
&\leqslant |w|\,(u + c_1\widetilde{\gamma}_n(\lambda) + c_1\widetilde{\gamma}_n(\lambda)u) \leqslant c_2\widetilde{\gamma}_n(\lambda)|w|.
\end{aligned}
$$

So $\widehat{w} = w(1+\Delta w)$, where $|\Delta w| \leqslant c_2\widetilde{\gamma}_n(\lambda)$ holds with probability at least $p_1(\lambda, n)$. We then have, using (4.1.4) and (4.1.5),

$$\widehat{\beta} = \operatorname{fl}(1/(\widehat{s}\widehat{w})) = \frac{1+\theta_2}{s(1+\Delta s)w(1+\Delta w)} = \frac{1}{sw}(1+\xi),$$

where $|\xi| \leqslant c_3\widetilde{\gamma}_n(\lambda) + O(u^2)$. □

If we redefine the Householder matrix as $P = I - vv^T$ with $\|v\|_2 = \sqrt{2}$ then Lemma 4.6 amounts to the result

$$\widehat{v} = v + \Delta v, \quad |\Delta v| \leqslant c_4\widetilde{\gamma}_n(\lambda)|v|, \tag{4.2.3}$$

where the bound holds with probability at least $p_1(\lambda, n)$.

4.3 MATRIX CONCENTRATION INEQUALITIES

The proof of Lemma 4.3 in [8] makes use of the scalar Azuma–Hoeffding inequality [18, Thm. 13.4]. Here, we will use a matrix version of that result due to Tropp [19, Thm. 7.1]. Our notation is as follows: we write (in this section only), $X \leqslant Y$, where X and Y are symmetric, to mean that $Y - X$ is positive semidefinite; the expectation of a matrix is defined componentwise; and we denote by $\lambda_{\max}$ the largest eigenvalue of a symmetric matrix. Note that "with probability 1" is often expressed as "almost surely".

Theorem 4.7. *Let $X_1, \dots, X_n$ be a sequence of random symmetric $d \times d$ matrices. If $\mathbb{E}(X_k \mid X_{k-1}, \dots, X_1) = 0$ and $X_k^2 \leqslant A_k^2$ with probability 1 for all k, where $A_1, \dots, A_n$ is a fixed sequence of symmetric $d \times d$ matrices, then for all $t \geqslant 0$,*

$$\mathbb{P}\left(\lambda_{\max}\left(\sum_{k=1}^n X_k\right) \geqslant t\right) \leqslant d\exp\left(-t^2/(8\sigma^2)\right), \tag{4.3.1}$$

where

$$\sigma^2 = \left\|\sum_{k=1}^n A_k^2\right\|_2. \tag{4.3.2}$$

We note briefly that the constant 1/8 in (4.3.1) can be improved to 1/2 as in our later invocation of Theorem 4.7, X_k commutes with A_k [19, Rem. 7.4]. We choose to leave the theorem in its most general form.

We need a version of Theorem 4.7 for nonsymmetric matrices. Define

$$\phi(B) = \begin{bmatrix} 0 & B \\ B^T & 0 \end{bmatrix} \tag{4.3.3}$$

to be the symmetric dilation of the rectangular matrix B. It is well known that

$$\lambda_{\max}(\phi(B)) = \|B\|_2. \tag{4.3.4}$$

The required result can be obtained by applying the theorem to the symmetric dilation of the sequence $X_1, \dots, X_n$ and using (4.3.4) [19, Rem. 7.3]. We also rescale by defining $\lambda = t/(2\sqrt{2}\sigma)$.

Theorem 4.8. *Let $X_1, \dots, X_n$ be a sequence of random $d_1 \times d_2$ matrices. If $\mathbb{E}(X_k \mid X_{k-1}, \dots, X_1) = 0$ for all k, and $X_k X_k^T \preccurlyeq A_k A_k^T$ and $X_k^T X_k \preccurlyeq A_k^T A_k$ with probability 1 for all k, where $A_1, \dots, A_n$ is a fixed sequence of $d_1 \times d_2$ matrices, then for all $\lambda \geqslant 0$,*

$$\mathbb{P}\left(\left\| \sum_{k=1}^n X_k \right\|_2 \geqslant 2\sqrt{2}\sigma\lambda \right) \leqslant (d_1 + d_2) \exp(-\lambda^2), \tag{4.3.5}$$

where

$$\sigma^2 = \max\left(\left\| \sum_{k=1}^n A_k A_k^T \right\|_2, \left\| \sum_{k=1}^n A_k^T A_k \right\|_2 \right) \leqslant \sum_{k=1}^n \|A_k\|_2^2. \tag{4.3.6}$$

This is the key result that we need in the next section.

4.4 APPLICATION OF A SEQUENCE OF HOUSEHOLDER MATRICES

Our probabilistic rounding error analysis for Householder QR factorization follows a similar strategy as for the worst-case analysis in [13, sec. 19.3]: we analyze the application of a sequence of general Householder matrices to a vector, then the application of the same sequence to a matrix, and finally specialize to the Householder matrices used in QR factorization.

For a single application of an $m \times m$ Householder matrix our backward error analysis gives an error constant proportional to $m^{1/2}$. A straightforward inductive argument for the application of n Householder matrices would lead to a backward error bound with a constant proportional to $nm^{1/2}$, which is unsatisfactory as it is linear in n, as for the worst-case analysis. Our key observation is that by applying the matrix concentration inequality in Theorem 4.8 we can obtain a bound with a constant of order $n^{1/2}m^{1/2}$. In the first three lemmas we take the Householder vectors to be exact.

Lemma 4.9. *Consider the (given) Householder matrices*

$$P_i = I - v_i v_i^T \in \mathbb{R}^{m\times m}, \quad v_i^T v_i = 2, \quad i = 1\colon r,$$

and the product

$$a_{r+1} = P_r \dots P_2 P_1 a, \quad a = a_1 \in \mathbb{R}^m,$$

computed as $a_{j+1} = P_j a_j = a_j - (v_j^T a_j) v_j$, $j = 1\colon r$. *Under Model 4.4, the computed* $\widehat{a}_{r+1}$ *satisfies*

$$\widehat{a}_{r+1} = (P_r + \Delta P_r) \dots (P_1 + \Delta P_1) a, \tag{4.4.1}$$

where

$$\|\Delta P_j\|_2 \leqslant c_5 \widetilde{\gamma}_m(\lambda) \tag{4.4.2}$$

holds for all j with probability at least $p_2(\lambda, m, r) = 1 - 2rm\exp(-\lambda^2/2)$.

Proof. Define the scalars $s_j = v_j^T a_j$. From standard rounding error analysis [13, sec. 3.1], [16, Proof of Thm. 3.1], the computed s_j satisfy $\widehat{s}_j = v_j^T D_j \widehat{a}_j$, where

$$D_j = \operatorname{diag}(d_k^{(j)}), \quad d_k^{(j)} = \prod_{i=1}^{m}(1+\delta_{k,i}^{(j)}), \quad |\delta_{k,i}^{(j)}| \leqslant u, \quad i,k = 1:m.$$

Then we form

$$\widehat{a}_{j+1} = \mathrm{fl}(\widehat{a}_j - \widehat{s}_j v_j) = \left(I + \Lambda_1^{(j)}\right)\left(\widehat{a}_j - \left(I + \Lambda_2^{(j)}\right)\widehat{s}_j v_j\right),$$

where

$$\Lambda_k^{(j)} = \operatorname{diag}(\varepsilon_{k,i}^{(j)}), \quad |\varepsilon_{k,i}^{(j)}| \leqslant u, \quad k = 1,2.$$

Note that $\Lambda_1^{(j)}$ depends on $\Lambda_2^{(j)}$, and that since we are using Model 4.4 the $\delta_{k,i}^{(j)}$ and $\varepsilon_{k,i}^{(j)}$ are mean independent random variables of mean zero. Hence

$$\begin{aligned}
\widehat{a}_{j+1} &= \left(I + \Lambda_1^{(j)}\right)\left(\widehat{a}_j - \left(I + \Lambda_2^{(j)}\right)v_j^T D_j \widehat{a}_j \cdot v_j\right) \\
&= \left(I + \Lambda_1^{(j)}\right)\left(I - \left(I + \Lambda_2^{(j)}\right)v_j v_j^T D_j\right)\widehat{a}_j \\
&= \left[I - \left(I + \Lambda_2^{(j)}\right)\left(v_j v_j^T + v_j v_j^T (D_j - I)\right) + \Lambda_1^{(j)}\left(I - \left(I + \Lambda_2^{(j)}\right)v_j v_j^T D_j\right)\right]\widehat{a}_j \\
&= (P_j + \Delta P_j)\widehat{a}_j,
\end{aligned} \tag{4.4.3}$$

where

$$\Delta P_j = -\Lambda_2^{(j)} v_j v_j^T - \underbrace{\left(I + \Lambda_2^{(j)}\right)v_j v_j^T (D_j - I)}_{(*)} + \Lambda_1^{(j)}\left(I - \left(I + \Lambda_2^{(j)}\right)v_j v_j^T D_j\right), \tag{4.4.4}$$

and so

$$\begin{aligned}\|\Delta P_j\|_2 &\leqslant 2u + 2(1+u)\|D_j - I\|_2 + u\big(1 + 2(1+u)\|D_j\|_2\big)\\ &= 3u + 2(1+u)\|D_j - I\|_2 + 2u(1+u)\|D_j\|_2.\end{aligned}$$

From Lemma 4.3 we have that for any constant $\lambda > 0$ and any particular j,

$$d_k^{(j)} = 1 + \psi_j^{(k)}, \quad |\psi_j^{(k)}| \leqslant \widetilde{\gamma}_m(\lambda), \tag{4.4.5}$$

holds for any particular k and j with probability at least $p(\lambda) = 1 - 2\exp(-\lambda^2/2)$, or equivalently it fails to hold with probability at most $1 - p(\lambda)$. By the inclusion–exclusion principle [22, p. 39] (4.4.5) fails to hold for at least one of the pairs (j,k) with probability at most $rm(1-p(\lambda))$, which means that it holds for all j and k with probability at least $p_2(\lambda, m, r) = 1 - rm(1-p(\lambda)) = 1 - 2rm\exp(-\lambda^2/2)$.

Hence $\|I - D_j\|_2 \leqslant \widetilde{\gamma}_m(\lambda)$ holds for all j with probability at least $p_2(\lambda, m, r)$ and hence

$$\|\Delta P_j\|_2 \leqslant 3u + 2(1+u)\widetilde{\gamma}_m(\lambda) + 2u(1+u)(1+\widetilde{\gamma}_m(\lambda)) \leqslant c_5\widetilde{\gamma}_m(\lambda) \tag{4.4.6}$$

holds for all j with probability at least $p_2(\lambda, m, r)$. □

Lemma 4.10. *With the same notation as in Lemma 4.9, assume that the bound (4.4.2) holds with probability 1 for all j. Then the computed $\widehat{a}_{r+1}$ satisfies*

$$\widehat{a}_{r+1} = Q^T(a + \Delta a), \quad \|\Delta a\|_2 \leqslant c_6\lambda\sqrt{r}\,\widetilde{\gamma}_m(\lambda)\|a\|_2 + O(u^2) \tag{4.4.7}$$

with probability at least $p_3(\lambda, m) = 1 - 2m\exp(-\lambda^2)$, where $Q = (P_r \dots P_2 P_1)^T$.

Proof. We can rewrite (4.4.1) as

$$\begin{aligned}\widehat{a}_{r+1} &= a_{r+1} + \left(\sum_{j=1}^{r} P_r \dots P_{j+1} \Delta P_j P_{j-1} \dots P_1 + O(u^2)\right) a \\ &= a_{r+1} + Q^T \left(\sum_{j=1}^{r} F_j + O(u^2)\right) a, \end{aligned} \tag{4.4.8}$$

where

$$F_j = P_1 \dots P_j \Delta P_j P_{j-1} \dots P_1. \tag{4.4.9}$$

Our aim is to show that $\mathbb{E}(F_j \mid F_{j-1}, \dots, F_1) = 0$, so that we can apply Theorem 4.8 with $X_k = F_k$. When we substitute the expressions for F_j and then ΔP_j into this expression and use the linearity of the expectation we obtain a sum of expectations, each of which we need to show is zero. We will just consider two of the terms, which come from $(*)$ in (4.4.4), as the other terms are treated similarly.

The matrix D_j contains the rounding errors from the computation of $v_j^T a_j$, whereas all the other terms in F_j contain rounding errors from later computations.

For the first part of the term $(*)$ in (4.4.4) we have

$$\begin{aligned}&\mathbb{E}(P_1 \dots P_j v_j v_j^T (I - D_j) P_{j-1} \dots P_1 \mid F_{j-1}, \dots, F_1) \\ &\qquad = P_1 \dots P_j v_j v_j^T \mathbb{E}(I - D_j \mid F_{j-1}, \dots, F_1) P_{j-1} \dots P_1, \end{aligned} \tag{4.4.10}$$

since the v_j and P_j are constant. We need the general form of the law of total expectation (LTE), $\mathbb{E}(\mathcal{X} \mid \mathcal{Y}) = \mathbb{E}(\mathbb{E}(\mathcal{X} \mid \mathcal{Z}) \mid \mathcal{Y})$, for any pair of measurable sets $\mathcal{Y}$ and $\mathcal{Z}$ such that $\mathcal{Y} \subseteq \mathcal{Z}$ [3, Thm. 34.4]. We define the set

$$\mathcal{Z}_s = \left\{ \left\{ \delta_{k,i}^{(\ell)} : i, k = 1 : m,\ \ell = 1 : s \right\}, F_{j-1}, \dots, F_1 \right\}. \tag{4.4.11}$$

By the LTE

$$\mathbb{E}(D_j \mid F_{j-1}, \dots, F_1) = \mathbb{E}\big(\mathbb{E}(D_j \mid \mathcal{Z}_{j-1}) \mid F_{j-1}, \dots, F_1\big) = I,$$

by [8, Lem. 6.4], in view of Model 4.4. Hence the expectation (4.4.10) is zero.

For the second part of the term $(*)$ in (4.4.4) we have

$$\begin{aligned}&\mathbb{E}(P_1 \dots P_j \Lambda_2^{(k)} v_j v_j^T (I - D_j) P_{j-1} \dots P_1 \mid F_{j-1}, \dots, F_1)\\ &\quad = P_1 \dots P_j \mathbb{E}\Big(\Lambda_2^{(j)} v_j v_j^T (I - D_j) \mid F_{j-1}, \dots, F_1\Big) P_{j-1} \dots P_1.\end{aligned}$$

By the LTE,

$$\begin{aligned}\mathbb{E}\Big(\Lambda_2^{(j)} v_j v_j^T (I - D_j) \mid F_{j-1}, \dots, F_1\Big) &= \mathbb{E}\Big(\mathbb{E}\Big(\Lambda_2^{(j)} v_j v_j^T (I - D_j) \mid \mathcal{Z}_j\Big) \mid F_{j-1}, \dots, F_1\Big)\\ &= \mathbb{E}\Big(\mathbb{E}\Big(\Lambda_2^{(j)} \mid \mathcal{Z}_j\Big) v_j v_j^T (I - D_j) \mid F_{j-1}, \dots, F_1\Big)\\ &= 0,\end{aligned}$$

since $\mathbb{E}\big(\Lambda_2^{(j)} \mid \mathcal{Z}_j\big) = 0$ by Model 4.4.

We now bound $\|F_j\|_2$, so that we can apply Theorem 4.8. By (4.4.2), (4.4.9), and the assumption of the lemma,

$$\|F_j\|_2^2 = \|\Delta P_j\|_2^2 \leqslant c_5^2 \widetilde{\gamma}_m(\lambda)^2, \quad j = 1 : r,$$

with probability 1. Hence we can take $X_j = F_j$, $A_j = c_5 \widetilde{\gamma}_m(\lambda) I_{d_1, d_2}$, and $\sigma^2 = r c_5^2 \widetilde{\gamma}_m(\lambda)^2$ in Theorem 4.8 and set $E = \sum_{j=1}^r F_j$, to obtain

$$\mathbb{P}\big(\|E\|_2 \geqslant 2\sqrt{2} c_5 \lambda \sqrt{r}\, \widetilde{\gamma}_m(\lambda)\big) \leqslant 2m \exp(-\lambda^2), \tag{4.4.12}$$

or equivalently

$$\|E\|_2 \leqslant c_6 \lambda \sqrt{r}\, \widetilde{\gamma}_m(\lambda), \tag{4.4.13}$$

holding with probability at least $1 - 2m\exp(-\lambda^2)$.

From (4.4.8) we have

$$\widehat{a}_{r+1} = Q^T a + Q^T E a + O(u^2) = Q^T(a + \Delta a)$$

with $\Delta a = Ea + O(u^2)$, and so

$$\|\Delta a\|_2 \leqslant \|Ea\|_2 + O(u^2) \leqslant c_6 \lambda \sqrt{r}\, \widetilde{\gamma}_m(\lambda)\|a\|_2 + O(u^2). \tag{4.4.14}$$

□

Now we consider the backward error in applying a sequence of Householder matrices to a matrix.

Lemma 4.11. *Consider the sequence of transformations*

$$A_{k+1} = P_k A_k, \quad k = 1\!:\!r.$$

where $A_1 = A \in \mathbb{R}^{m\times n}$, each $P_k \in \mathbb{R}^{m\times m}$ is a Householder matrix. Under Model 4.4 and the assumption of Lemma 4.10, the computed matrix $\widehat{A}_{r+1}$ satisfies

$$\widehat{A}_{r+1} = Q^T(A + \Delta A),$$

where $Q^T = P_r P_{r-1} \dots P_1$ and where

$$\|\Delta a_j\|_2 \leqslant c_6 \lambda \sqrt{r} \widetilde{\gamma}_m(\lambda)\|a_j\|_2 + O(u^2), \quad j = 1\!:\!n, \tag{4.4.15}$$

holds with probability at least $p_4(\lambda, m, n) = 1 - 2mn\exp(-\lambda^2)$.

Proof. Lemma 4.10 shows that for each j the bound (4.4.7) holds with $a = a_j$ with probability at least $p_3(\lambda, m)$. The probability that it holds for all columns is given

by 1 minus the probability that it fails for at least one, which is at least $1 - n(1 - p_3(\lambda, m)) = 1 - 2mn\exp(-\lambda^2)$. □

The probabilistic result for Householder QR factorization, analogous to the worst-case rounding error result Theorem 4.1, now follows.

Theorem 4.12. *Let $\widehat{R} \in \mathbb{R}^{m\times n}$ be the computed upper trapezoidal QR factor of $A \in \mathbb{R}^{m\times n}$ obtained via the Householder QR algorithm. Under Model 4.4 and the assumption of Lemma 4.10, there exists an orthogonal $Q \in \mathbb{R}^{m\times m}$ such that*

$$A + \Delta A = Q\widehat{R}, \tag{4.4.16}$$

where

$$\|\Delta a_j\|_2 \leqslant c_6\lambda\sqrt{n}\,\widetilde{\gamma}_m(\lambda)\|a_j\|_2 + O(u^2), \quad j = 1:n, \tag{4.4.17}$$

holds with probability at least $p_4(\lambda, m, n) = 1 - 2mn\exp(-\lambda^2)$.

Proof. The theorem follows from Lemma 4.11 as long as we note two subtleties. First, the Householder matrices in the lemma are completely general, yet for QR factorization they are chosen to introduce zeros into vectors and we explicitly set those elements to zero rather than compute them. This essentially forces rows of ΔP_i in (4.4.4) to be zero, which does not increase $\|\Delta P_i\|_2$, so the bounds still hold.

The second subtlety is that for Householder QR factorization the Householder vector v_j in Lemma 4.9 depends on previous computed quantities and is computed itself, so is subject to rounding error. The fact that v_j is no longer a constant as regards the conditional probabilities can be dealt with by adding "$v_j, v_{j-1}, \ldots, v_1$" to $\mathcal{Z}_s$ in (4.4.11). The key point is that D_j, $\Lambda_1^{(j)}$, and $\Lambda_2^{(j)}$ depend on rounding errors that occur later than those on which v_j depends. The result (4.2.3) for the construction of a Householder vector, together with (4.4.4), shows that the error in v_j adds a

perturbation of order $uc_4\widetilde{\gamma}_m(\lambda) = O(u^2)$ to ΔP_j, so it does not change the bound (4.4.6) and hence does not change (4.4.17). □

Theorem 4.12 bounds the backward error to first order, while the analogous worst-case result, Theorem 4.1, bounds all orders. We argue that the exclusion of higher order terms is inconsequential. Set $r = n$ in Lemma 4.10. The $O(u^2)$ term that we dropped from the analysis after the expansion of (4.4.1) at the beginning of the proof of Lemma 4.10 comprises $\binom{n}{2}$ terms of order u^2 and in view of (4.4.2) is bounded by (omitting constants c_i and λ) $n^2\widetilde{\gamma}_m(\lambda)^2\|a\|_2 \approx n^2mu^2\|a\|_2$. The first-order term has $\sqrt{mn}u$ dependence, while the second-order term has mn^2u^2 dependence. Thus for problem sizes with $m^{1/2}n^{3/2}u > 1$, second-order terms could dominate. This, in effect, imposes the requirement

$$m^{1/2}n^{3/2}u < 1 \tag{4.4.18}$$

for our analysis to imply that the error grows like $\sqrt{mn}u$. The higher order terms do not affect this argument. Indeed the ratio of the bound for the $(k+1)$st-order terms divided by the bound for the kth-order terms is

$$\frac{\binom{n}{k+1}(m^{1/2}u)^{k+1}}{\binom{n}{k}(m^{1/2}u)^{k}} = \frac{\frac{n!}{(n-k-1)!(k+1)!}m^{(k+1)/2}u^{k+1}}{\frac{n!}{(n-k)!k!}m^{k/2}u^{k}} = \frac{n-k}{k+1}m^{1/2}u,$$

and this ratio is less than 1 for $k \geqslant 2$ given (4.4.18). However, (4.4.18) is no more restrictive than existing assumptions about the worst-case analysis, as Theorem 4.1 has the assumption

$$mnu < 1. \tag{4.4.19}$$

Since we have $m \geqslant n$, and in many applications $m \gg n$, (4.4.18) can be significantly less restrictive than (4.4.19). While we ideally would have probabilistic bounds for the higher order terms, our analysis has still shown that we can significantly tighten the error bounds for the same problem sizes for which the worst-case analysis holds.

Ideally, we would remove the assumption in Lemmas 4.10 and 4.11 and Theorem 4.12 that the bound (4.4.2) holds with probability 1 for all j. We have effectively replaced a probability of $1 - 2rm\exp(-\lambda^2/2)$ with a probability of 1. To remove this assumption we need a version of Theorem 4.8 that allows $X_k X_k^T \leqslant A_k A_k^T$, and $X_k^T X_k \leqslant A_k^T A_k$ to hold with a certain probability less than 1 rather than almost surely. Such a result is not, to our knowledge, available in the literature on concentration inequalities, and deriving one requires further research that is beyond the scope of this paper.

The columnwise bound (4.4.17) implies a normwise one:

$$\|\Delta A\|_F \leqslant c_6 \lambda \sqrt{n}\, \widetilde{\gamma}_m(\lambda) \|A\|_F + O(u^2). \tag{4.4.20}$$

The equivalent bound for the worst-case analysis is (4.1.3).

The conclusion from our analysis is that the constant mnu in the worst-case backward error bound for Householder QR factorization reduces to $\sqrt{mn}u$ in the probabilistic bound.

Finally, we derive a probabilistic error bound on the loss of orthogonality in the computed factor $\widehat{Q}$ formed as the product of the Householder matrices, computed in the more efficient right to left order.

Theorem 4.13. *Let $\widehat{Q} \in \mathbb{R}^{m\times m}$ be the computed orthogonal QR factor of $A \in \mathbb{R}^{m\times n}$ obtained by forming $Q = P_1 P_2 \dots P_n$ in the right to left order. Under Model 4.4 and the assumption of Lemma 4.10, $\widehat{Q} = Q + \Delta Q$, where the jth column of ΔQ is bounded by*

$$\|\Delta q_j\|_2 \leqslant c_6 \lambda \sqrt{n}\, \widetilde{\gamma}_m(\lambda) + O(u^2), \quad j = 1 : m, \tag{4.4.21}$$

with probability at least $p_4(\lambda, m, n) = 1 - 2mn\exp(-\lambda^2)$. Moreover,

$$\|\widehat{Q}^T \widehat{Q} - I\|_F \leqslant c_7 \lambda \sqrt{mn}\, \widetilde{\gamma}_m(\lambda) + O(u^2) \tag{4.4.22}$$

holds with the same probability.

Proof. Applying Lemma 4.10 with the P_i taken in the reverse order, with $r = n$ and a the jth column of the $m \times m$ identity matrix, gives (4.4.21), where the probability follows by the same argument as in the proof of Lemma 4.11.

By [14, Thm. 8.17], with U denoting the orthogonal polar factor of $\widehat{Q}$, which is the nearest orthogonal matrix to $\widehat{Q}$ in the Frobenius norm [14, Thm. 8.4], we have

$$\begin{aligned}\|\widehat{Q}^T\widehat{Q} - I\|_F &\leqslant (1 + \|\widehat{Q}\|_2)\|\widehat{Q} - U\|_F \\ &\leqslant (1 + \|\widehat{Q}\|_2)\|\widehat{Q} - Q\|_F \\ &\leqslant c_7\lambda\sqrt{mn}\,\widetilde{\gamma}_m(\lambda) + O(u^2).\end{aligned}$$

□

The worst-case bound corresponding to (4.4.21) has constant mnu [13, p. 360], so again the dimensions are square-rooted.

4.5 DISCUSSION

We now discuss several aspects and implications of the error analysis.

4.5.1 Choice of λ

We begin with a brief discussion of the probabilities associated with the bounds in Lemma 4.9 and Theorem 4.12. As noted in previous work [8], [16], [17], these probabilities are typically pessimistic and setting $\lambda = 1$ almost always provides bounds

Table 4.5.1: The value of $1 - p_5(\lambda, m, m)$ for various choices of λ and m. The underlined entries correspond to negative probabilities.

	m			
λ	10^2	10^4	10^6	10^8
6.0	3.0460e−04	3.0460e+00	3.0460e+04	3.0460e+08
7.0	4.5795e−07	4.5795e−03	4.5795e+01	4.5795e+05
8.0	2.5328e−10	2.5328e−06	2.5328e−02	2.5328e+02
9.0	5.1535e−14	5.1535e−10	5.1535e−06	5.1535e−02
10.0	3.8575e−18	3.8575e−14	3.8575e−10	3.8575e−06
11.0	1.0622e−22	1.0622e−18	1.0622e−14	1.0622e−10
12.0	1.0760e−27	1.0760e−23	1.0760e−19	1.0760e−15

that hold in practice. Define $1 - p_5(\lambda, m, n) = 2mn\big(\exp(-\lambda^2) + \exp(-\lambda^2/2)\big)$, which is an upper bound on the probabilities in Lemma 4.9 and Theorem 4.12 that the respective bounds do not hold. Table 4.5.1 shows values of $1 - p_5(\lambda, m, m)$. For certain values the upper bounds are negative. However, for $\lambda = 10$ and all the problem sizes shown in Table 4.5.1, p_5 is within 4×10^{-6} of it.

4.5.2 Aggregated Householder transformations

In practice, Householder transformations are usually aggregated in order to express the computation primarily in terms of matrix multiplication. A common form of aggregation is the WY representation [4], [13, sec. 19.5]. We represent the product $Q_r = P_r P_{r-1} \dots P_1$ of b Householder matrices $P_i = I - v_i v_i^T$ as

$$Q_r = I + W_b Y_b^T, \quad W_b, Y_b \in \mathbb{R}^{m \times b}.$$

This is done through the recurrence

$$W_1 = -v_1, \quad Y_1 = v_1, \qquad W_i = \begin{bmatrix} W_{i-1} & -v_i \end{bmatrix}, \quad Y_i = \begin{bmatrix} Y_{i-1} & Q_{i-1}^T v_i \end{bmatrix}.$$

We partition A as

$$A = \begin{bmatrix} A_1 & B \end{bmatrix}, \quad A_1 \in \mathbb{R}^{m \times r}$$

and transform A_1 to upper trapezoidal form by forming $P_r \dots P_1 A_1$, accumulating the product $P_r \dots P_1 = I + W_r Y_r^T$. The matrix B is then updated via $B \leftarrow B + W_r(Y_r^T B)$, and we repeat this process on the remaining rows of B.

We do not perform a full analysis of the aggregated algorithm. We simply note that the two core operations of the aggregated Householder QR factorization are the application of a sequence of Householder matrices, in forming $P_r \dots P_1 A_1$, and matrix multiplication in forming $B \leftarrow B + W_r(Y_r^T B)$. Both of these operations have been shown to have probabilistic bounds whose constants are the square roots of those in the worst-case bounds, so we can expect the same to be true for the aggregated algorithm.

4.5.3 Mixed-precision QR factorization

Yang, Fox, and Sanders [23, Thm. 4.1] consider a mixed precision Householder QR factorization algorithm in which the working precision is u_{low} and the inner products are computed at precision u_{high}. They obtain a normwise backward error bound of order $n(u_{\text{low}} + mu_{\text{high}})$. Our analysis is readily adapted for this algorithm by replacing $\widetilde{\gamma}_m = \widetilde{\gamma}_m(u)$ in (4.4.2) by $\widetilde{\gamma}_m(u_{\text{high}}) + O(u_{\text{low}})$, and the resulting probabilistic error bound is of order $n^{1/2}(u_{\text{low}} + m^{1/2}u_{\text{high}})$.

4.5.4 Two-sided transformations

Householder matrices and Householder QR factorization are tools in many algorithms, including the reduction of matrices to tridiagonal or Hessenberg form for the QR algorithm for eigenvalues and to bidiagonal form for the QR algorithm for singular values [12]. Can we use our results to obtain probabilistic backward error bounds with reduced constants for these reductions? For the reduction of $A \in \mathbb{R}^{n\times n}$

to Hessenberg form H we have $H = P_{n-2} \dots P_1 A P_1^T \dots P_{n-2}^T$. The reduction is carried out in the order

$$H = P_{n-2}(\dots(P_1 A P_1^T)\dots)P_{n-2}^T, \tag{4.5.1}$$

because P_{j+1} depends on $P_j \dots P_1 A P_1^T \dots P_j^T$. Our analysis does not directly apply to this two-sided case. However, consider applying all the transformations on the left before applying those on the right:

$$H = (P_{n-2} \dots P_1 A)P_1^T \dots P_{n-2}^T \equiv (PA)P^T. \tag{4.5.2}$$

Two applications of Lemma 4.11 give

$$\begin{aligned} \widehat{H} &= \big(P(A + \Delta A_1) + \Delta A_2\big)P^T \\ &= P(A + \Delta A)P^T, \qquad \Delta A = \Delta A_1 + P^T \Delta A_2. \end{aligned}$$

The probabilistic bounds on $\|\Delta A_1\|_F$ and $\|\Delta A_2\|_F$ from the lemma are proportional to nu and hence so is the bound for $\|\Delta A\|_F$. The usual worst-case bound obtained by Wilkinson is proportional to $n^2 u$ [21, pp. 160–161] (we have accounted for the fact that Wilkinson assumes that inner products are accumulated at twice the working precision).

There are two reasons why the probabilistic backward error bound for (4.5.2) should be indicative of the backward error for (4.5.1). First, (4.5.2) does many more operations, since it incurs substantial fill-in; in particular $P_{n-2} \dots P_1 A$ is full, apart from the first column. We have carried out extensive numerical experiments, in which the backward error for (4.5.2) was always of the same, or smaller, order of magnitude as that for (4.5.1). The second reason is that in practice the Householder matrices are aggregated, so that the actual computation lies somewhere between (4.5.1) and (4.5.2).

For the multishift Hessenberg QR iteration the bulge chasing is implemented using Householder matrices [6], [7], so Lemma 4.11 can be applied again for sufficiently large bulges, and for small bulges one can apply the worst-case bounds.

4.5.5 Other forms of QR factorization

The modified Gram–Schmidt algorithm applied to $A \in \mathbb{R}^{m\times n}$ with $m \geqslant n$ is known to be equivalent both mathematically and numerically to Householder QR factorization applied to the augmented matrix $\left[\begin{smallmatrix} 0 \\ A \end{smallmatrix}\right]$ [5], [13, sec. 19.8]. Hence we can apply Theorem 4.12, and since the augmented matrix is $(m+n)\times n$ the constant obtained will be proportional to $\sqrt{(m+n)n}u \approx \sqrt{mn}u$ since $m \geqslant n$. (As explained in the above references, some additional work is needed to obtain a backward error result for modified Gram–Schmidt and the constant will be increased.)

In [1], a randomized process for computing the QR factorization of $A \in \mathbb{R}^{m\times n}$ is presented. It uses a randomized Gram-Schmidt process, with the approach based on the dimension reduction technique of random sketching. The authors give a rounding error analysis under the assumption that the rounding errors are mean independent random variables of zero mean. They exploit mixed-precision arithmetic with two precisions $u_{\text{fine}} < u_{\text{crs}}$, where u_{crs} is used for computing the projections and u_{fine} for everything else. The analogous result to Theorem 4.12 is [1, Thm. 3.2] $\|A - \widehat{Q}\widehat{R}\|_F \leqslant cu_{\text{crs}}n^{3/2}\|A\|_F$, where $\widehat{Q} \in \mathbb{R}^{m\times n}$ is the computed orthonormal QR factor.

Finally, we consider Givens QR factorization. Givens transformations operate on vectors of length 2, so there is no benefit to a probabilistic approach in analyzing the application of a single rotation. The backward error analysis [13, sec. 19.6] shows that each individual rotation introduces a backward error bounded by a small constant

and that the $m+n-2$ products of disjoint rotations needed for a QR factorization lead to a result of the same form as Theorem 4.1 but with

$$\|\Delta a_j\|_2 \leqslant c_7 \gamma_{m+n-2} \|a_j\|_2, \quad j = 1\colon n.$$

A probabilistic analysis analogous to that in the proof of Lemma 4.9, using the matrix concentration inequality, leads to a result of the same form as Theorem 4.12 but with probabilistic bound

$$\|\Delta a_j\|_2 \leqslant c_8 (m+n)^{1/2} u \, \|a_j\|_2 + O(u^2), \quad j = 1\colon n.$$

4.6 BACKWARD ERROR FOR QR FACTORIZATION

In our numerical experiments in the next section we need to compare the probabilistic backward error bounds with the actual backward errors. How to compute the backward error matrix $\Delta A = A - Q\widehat{R}$ for a given $\widehat{R}$, though, is not clear, since the orthogonal matrix Q in Theorems 4.1 and 4.12 is unknown. We will focus on the backward error measure

$$\begin{aligned} \mu(\widehat{R}) &= \min\left\{ \left(\sum_{j=1}^{n} \|d_j \Delta a_j\|_2^2 \right)^{1/2} : A + \Delta A = Q\widehat{R},\ Q \in \mathbb{R}^{m\times m},\ Q^T Q = I_m \right\} \quad (4.6.1) \\ &= \min\bigl(\|(A - Q\widehat{R})D\|_F : Q \in \mathbb{R}^{m\times m},\ Q^T Q = I_m,\ D = \operatorname{diag}(d_j) \bigr). \end{aligned}$$

For $D = \operatorname{diag}(\|a_j\|_2^{-1})$ we have a columnwise backward error and for $D = \|A\|_F^{-1} I$ the normwise relative backward error.

The next result shows how to compute $\mu(\widehat{R})$. Recall that the polar decomposition of $A \in \mathbb{R}^{n\times n}$ is a factorization $A = UH$, where U is orthogonal and H is symmetric positive semidefinite.

Theorem 4.14. *Let $A, B \in \mathbb{R}^{m\times n}$ and let $D = \operatorname{diag}(d_i) \in \mathbb{R}^{n\times n}$ be nonsingular. Then $\min\{\, \|(A - QB)D\|_F : Q \in \mathbb{R}^{m\times m},\ Q^TQ = I_m \,\}$ is obtained for $Q = U^T$, where U is the orthogonal polar factor of the matrix BD^2A^T.*

Proof. For $F, G \in \mathbb{R}^{m\times n}$, the orthogonal Procrustes problem has the form $\min\{\, \|F - GW\|_F : W \in \mathbb{R}^{n\times n},\ W^TW = I_n \,\}$ and any orthogonal polar factor of G^TF is a solution [14, Thm. 8.6]. Writing $\|(A - QB)D\|_F = \|AD - QBD\|_F = \|DA^T - DB^TQ^T\|_F$ therefore gives the result on taking $F = DA^T$ and $G = DB^T$. □

By Theorem 4.14, $\mu(\widehat{R}) = \|(A - Q\widehat{R})D\|_F$, where Q^T is an orthogonal polar factor of $\widehat{R}D^2A^T$. If $\widehat{R}D^2A^T = U\Sigma V^T$ is a singular value decomposition then we can take $Q^T = UV^T$.

We note that one can express $\mu(\widehat{R})^2 = \|AD\|_F^2 + \|\widehat{R}D\|_F^2 - 2\sum_{i=1}^n \sigma_i(\widehat{R}D^2A^T)$, where σ_i denotes the ith largest singular value. However, this formula suffers from severe cancellation, and rounding errors can cause it to evaluate as negative, so it is better to use the expression $\|(A - Q\widehat{R})D\|_F$. In fact, even the latter expression does not necessarily give a result of the correct order of magnitude, so it is best to compute the backward error at twice the working precision. Hence in our experiments we take single precision as the working precision and compute the backward error in double precision.

4.7 NUMERICAL EXPERIMENTS

For all our all numerical experiments we set the parameter $\lambda = 1$ and set all constants c_i in the error bounds (worst-case or probabilistic) to be 1. We use MATLAB R2021b and take IEEE single precision as the working precision. Normwise relative

backward errors, as defined by (4.6.1) with $D = \|A\|_F^{-1} I$, are computed as described in section 4.6, in double precision.

We use round to nearest in all the experiments. We have tried stochastic rounding, which ensures that the assumptions of Model 4.4 are satisfied [8], and found that the numerical results reported are virtually identical to those for round to nearest in these experiments.

In sections 4.7.1, 4.7.3, and 4.7.4 we use random $m \times n$ matrices with entries drawn uniformly from the interval $[0, 1]$ and we sample 10 different matrices for each pair of dimensions. In section 4.7.2 we use real-life matrices from the SuiteSparse collection.

4.7.1 Householder QR factorization for random matrices

In this experiment we test the backward error bound in Theorem 4.12. In order to study how the error grows with n and m independently, we first fix a value of n and vary m and then fix m and vary n. For each pair of dimensions we plot the maximum and mean normwise backward errors for Householder QR factorization along with associated worst-case and probabilistic bounds obtained from (4.1.3) and (4.4.20). The results are given in Figures 4.7.1 and 4.7.2. We see that the probabilistic bound $\sqrt{mn}u$ proves a much better indicator than the worst-case bound mnu of the size of the error and its rate of growth.

4.7.2 Householder QR factorization for SuiteSparse matrices

We also consider Householder QR factorization of some matrices from the SuiteSparse Matrix Collection [10], [11]. We select all matrices from the collection with

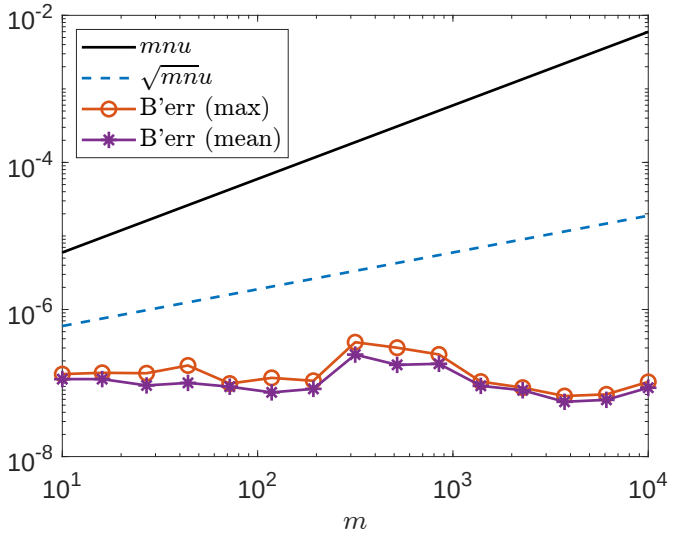


Figure 4.7.1: Normwise backward errors and bounds for Householder QR factorization for $n = 10$ and various m, for $m \times n$ matrices with elements sampled uniformly from $[0, 1]$.

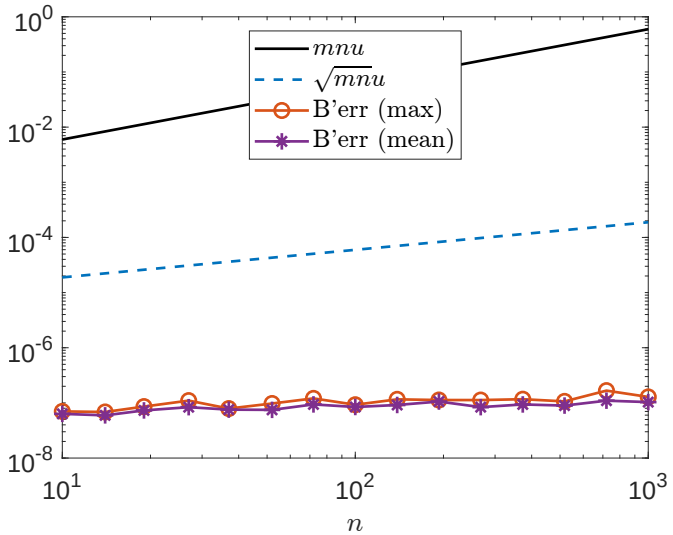


Figure 4.7.2: Normwise backward errors and bounds for Householder QR factorization for $m = 10^4$ and various n, for $m \times n$ matrices with elements sampled uniformly from $[0, 1]$.

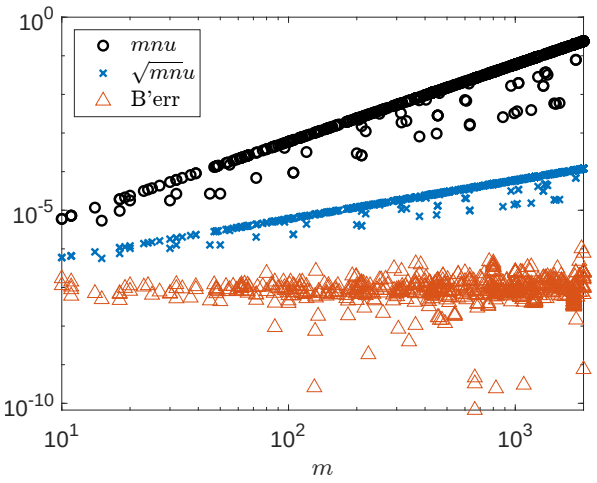


Figure 4.7.3: Normwise backward errors and bounds for 774 $m \times n$ matrices from the SuiteSparse collection.

$10 \leqslant m, n \leqslant 2 \times 10^3$ and $m \geqslant n$. This results in 842 matrices. We plot the same error quantities as for the random matrices. Some of the matrices in the collection have zero columns, so we filter these out. There are a few cases where the reported error exceeds even the deterministic bound, which we suspect is an underflow issue; a similar observation is made in [16, sec. 4.5]. We also filter out these cases from the reported results, which results in 774 matrices. We show the observed backward errors in Figure 4.7.3. Again, the probabilistic bound is satisfied and proves closer to the observed error than the worst-case bound by several orders of magnitude.

4.7.3 Givens rotations

Givens QR factorization is typically used for structured matrices, such as tridiagonal or upper Hessenberg matrices, but here we wish to see how the backward error of the factorization behaves for dense matrices. For random $n \times n$ matrices we plot in Figure 4.7.4 the maximum and mean normwise backward errors, the worst-case error bound $2nu$, and the probabilistic error bound $\sqrt{2n}\,u$. The backward error grows at a rate very similar to that of the probabilistic bound.

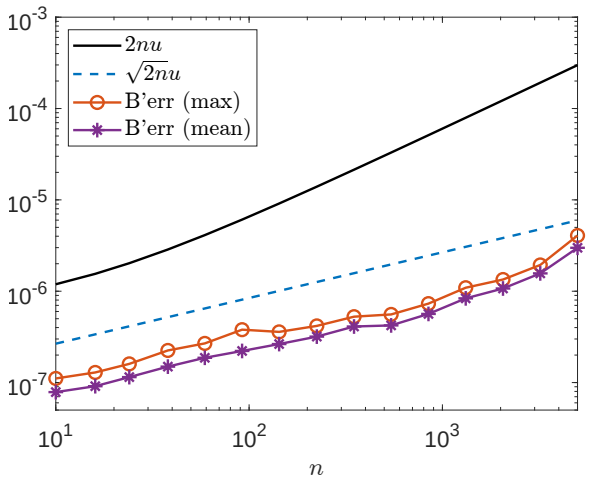


Figure 4.7.4: Normwise backward error and backward error bound for Givens QR factorization for $n \times n$ matrices with elements sampled uniformly from $[0, 1]$.

4.7.4 Reduction to Hessenberg form

Finally, we consider Householder reduction matrix to Hessenberg form: $A = QHQ^T$. Figure 4.7.5 shows normwise backward errors, computed as $\|A - \widehat{Q}H\widehat{Q}^T\|_F/\|A\|_F$, where $\widehat{Q}$ is the computed product of Householder matrices (we do not have an explicit backward error formula such as that in Theorem 4.14 in this two-sided case), and the worst-case and probabilistic bounds, n^2u and nu respectively. We see that the backward error satisfies the probabilistic bound, and again the probabilistic bound is a better indicator than the worst-case bound of the rate of growth of the backward error with n.

Throughout Section 4.7 we presented various comparisons of worst-case and probabilistic bounds and observed errors. In some cases, the probabilistic bounds not only bound the observed error, but also provide good estimates for how the error grows. In other cases, the probabilistic bound can be several orders of magnitude larger than the observed error. While it is important to emphasise that they are *bounds* and not predictions of the error, an interesting area for further research is to narrow this gap between bound and error for some of the cases. This is done in [17] for the case of summation, where some underlying assumptions are made about the data.

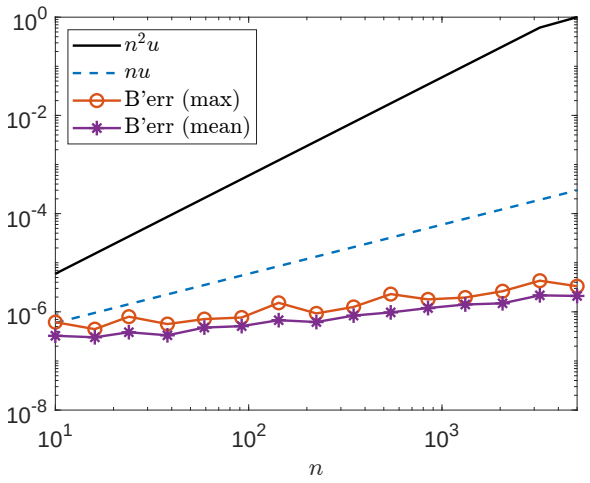


Figure 4.7.5: Normwise backward errors and bounds for reduction to Hessenberg form for $n \times n$ matrices with elements sampled uniformly from $[0, 1]$.

4.8 CONCLUSIONS

In a classic 1961 paper, Wilkinson [20, p. 318] carries out rounding error analyses of LU factorization, Givens QR factorization, and Householder QR factorization. He notes that "The bounds we have obtained are in all cases strict upper bounds. In general, the statistical distribution of the rounding errors will reduce considerably the function of n occurring in the relative errors. We might expect in each case that this function should be replaced by something which is no bigger than its square root and is usually appreciably smaller." Recent probabilistic rounding error analysis has provided a rigorous foundation for Wilkinson's statement for LU factorization and other inner product-based computations. Our work does the same for Householder QR factorization-based methods, as well as for Givens QR factorization, under the technical assumption in Lemma 4.10. The assumption we make is that a certain bound holds with probability 1, when in actuality it holds with at least some probability that we can make arbitrarily close to, but not equal to 1. In order to avoid making this assumption we require a version of Theorem 4.7 which bounds $X_k^2 \leqslant A_k^2$ with a certain probability. A potential alternative route is a theorem involving bounds on the moments of X_k. A related result in the scalar case is given by [2, Thm. 3.14].

A significant feature of the probabilistic backward error bounds is that they bound the likely rate of growth of the backward error as the problem dimensions increase. The rate of growth, along with blocking, exploiting architectural features of the hardware, and using other techniques to improve the accuracy of the computations, is what determines our ability to solve problems at extreme scale and possibly low precision in a numerically stable way [15].

REFERENCES

[1] O. Balabanov and L. Grigori. *Randomized Gram-Schmidt Process with Application to GMRES*. ArXiv:2011.05090. Nov. 2020. Revised January 2022 (cited on p. 105).

[2] B. Bercu, B. Delyon, and E. Rio. *Concentration Inequalities for Sums and Martingales*. Springer International Publishing, 2015 (cited on p. 112).

[3] P. Billingsley. *Probability and Measure*. Third. New York: Wiley, 1995 (cited on p. 95).

[4] C. H. Bischof and C. F. V. Loan. "The WY representation for products of Householder matrices". *SIAM J. Sci. Statist. Comput.* 8.1 (1987), s2–s13 (cited on p. 102).

[5] Å. Björck and C. C. Paige. "Loss and recapture of orthogonality in the modified Gram–Schmidt algorithm." *SIAM J. Matrix Anal. Appl.* 13.1 (1992), pp. 176–190 (cited on p. 105).

[6] K. Braman, R. Byers, and R. Mathias. "The multishift *QR* algorithm. Part I: Maintaining well-focused shifts and level 3 performance." *SIAM J. Matrix Anal. Appl.* 23.4 (2002), pp. 929–947 (cited on p. 105).

[7] K. Braman, R. Byers, and R. Mathias. "The multishift *QR* algorithm. Part II: Aggressive early deflation." *SIAM J. Matrix Anal. Appl.* 23.4 (2002), pp. 948–973 (cited on p. 105).

[8] M. P. Connolly, N. J. Higham, and T. Mary. "Stochastic rounding and its probabilistic backward error analysis." *SIAM J. Sci. Comput.* 43.1 (Jan. 2021), A566–A585 (cited on pp. 86–88, 90, 96, 101, 108).

[9] M. Croci, M. Fasi, N. J. Higham, T. Mary, and M. Mikaitis. "Stochastic rounding: Implementation, error analysis and applications." *Roy. Soc. Open Sci.* 9.3 (2022), pp. 1–25 (cited on p. 87).

[10] T. A. Davis. *SuiteSparse Matrix Collection*. https://sparse.tamu.edu/ (cited on p. 108).

[11] T. A. Davis and Y. Hu. "The University of Florida Sparse Matrix Collection." *ACM Trans. Math. Software* 38.1 (2011), 1:1–1:25 (cited on p. 108).

[12] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Fourth. Baltimore, MD, USA: Johns Hopkins University Press, 2013, pp. xxi+756 (cited on p. 103).

[13] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. Second. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2002, pp. xxx+680 (cited on pp. 84, 85, 88, 92, 93, 101, 102, 105).

[14] N. J. Higham. *Functions of Matrices: Theory and Computation*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2008, pp. xx+425 (cited on pp. 101, 107).

[15] N. J. Higham. *Numerical Stability of Algorithms at Extreme Scale and Low Precisions*. MIMS EPrint 2021.14. UK: Manchester Institute for Mathematical Sciences, The University of Manchester, Sept. 2021, p. 21. To appear in Proc. Int. Cong. Math. (cited on p. 113).

[16] N. J. Higham and T. Mary. "A new approach to probabilistic rounding error analysis." *SIAM J. Sci. Comput.* 41.5 (2019), A2815–A2835 (cited on pp. 84, 87, 88, 93, 101, 110).

[17] N. J. Higham and T. Mary. "Sharper probabilistic backward error analysis for basic linear algebra kernels with random data." *SIAM J. Sci. Comput.* 42.5 (2020), A3427–A3446 (cited on pp. 86, 101, 111).

[18] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. 2nd. USA: Cambridge University Press, 2017 (cited on p. 90).

[19] J. A. Tropp. "User-friendly tail bounds for sums of random matrices." *Found. Comput. Math.* 12.4 (Aug. 2012), pp. 389–434 (cited on pp. 90, 91).

[20] J. H. Wilkinson. "Error analysis of direct methods of matrix inversion." *J. ACM* 8.3 (July 1961), pp. 281–330 (cited on p. 112).

[21] J. H. Wilkinson. *The Algebraic Eigenvalue Problem*. Oxford, UK: Oxford University Press, 1965, pp. xviii+662 (cited on p. 104).

[22] D. Williams. *Weighing the Odds. A Course in Probability and Statistics*. Cambridge, UK: Cambridge University Press, 2001, pp. xvii+547 (cited on p. 94).

[23] L. M. Yang, A. Fox, and G. Sanders. "Rounding error analysis of mixed precision block Householder QR algorithms." *SIAM J. Sci. Comput.* 43.3 (2021), A1723–A1753 (cited on p. 103).

5 RANDOMIZED LOW RANK MATRIX APPROXIMATION: ROUNDING ERROR ANALYSIS AND A MIXED PRECISION ALGORITHM

Abstract. The available error bounds for randomized algorithms for computing a low rank approximation to a matrix assume exact arithmetic. Rounding errors potentially dominate the approximation error, though, especially when the algorithms are run in low precision arithmetic. We give a rounding error analysis of the method that computes a randomized rangefinder and then computes an approximate singular value decomposition approximation. Our analysis covers the basic method and the power iteration for the fixed-rank problem, as well as the power iteration for the fixed-precision problem. We see that for the fixed-rank problem, the bound for the power iteration is favourable in terms of simplicity and rounding error contribution. We give both worst-case and probabilistic rounding error bounds as functions of the problem dimensions and the rank. The worst-case bounds are pessimistic, but the probabilistic bounds are reasonably tight and still reliably bound the error in practice. We also propose a mixed precision version of the algorithm that offers potential speedups by gradually decreasing the precision during the execution of the algorithm.

Keywords: randomized algorithms, low rank matrix approximation, singular value decomposition, rounding error analysis, probabilistic rounding error analysis, mixed precision algorithm.

2010 MSC: 65G50, 65F05.

5.1 INTRODUCTION

Randomized algorithms for low rank matrix approximation, a problem with many applications in scientific computing, first began to appear two decades ago. Early, influential works, including [1], [6], [7], [16], [19], and [24], showed that for certain problems randomized algorithms can be significantly more computationally efficient than classical numerical linear algebra algorithms. Comprehensive surveys of randomized algorithms and low-rank approximation are given in [15] and [17]. Given a matrix $A \in \mathbb{R}^{m \times n}$ with $m \geqslant n$ and low numerical rank, the goal is to compute cheaply some useful factorization that approximates A well. The basic computational procedure can be split into two steps [6], [17, sect. 11].

- **Rangefinder:** Compute an orthonormal Q such that $A \approx QQ^T A$ and Q has as few columns as possible.

- **Factorization:** Use Q to help construct an approximate factorization of A.

The two steps are described in more detail in Algorithms 5.1 and 5.2. Here we focus specifically on computing the singular value decomposition (SVD), but other factorizations can also be computed using Q from Algorithm 5.1 (see, for example, [18, sect. 3]). Algorithm 5.1 describes the case of drawing a Gaussian matrix Ω. By a Gaussian matrix we mean a random matrix whose elements are independently drawn from the standard normal distribution (mean 0, variance 1). A thin QR factorization means one with a rectangular Q and an upper triangular R.

Algorithm 5.1 Given $A \in \mathbb{R}^{m \times n}$ with $m \geqslant n$ and an integer $k < n$, compute a matrix $Q \in \mathbb{R}^{m \times k}$ with orthonormal columns such that $A \approx QQ^T A$.

1: Draw a Gaussian matrix $\Omega \in \mathbb{R}^{n \times k}$.
2: $Y = A\Omega$
3: Compute an orthonormal basis $Q \in \mathbb{R}^{m \times k}$ for Y via a thin QR factorization $Y = QR$.

Algorithm 5.2 Given $A \in \mathbb{R}^{m\times n}$ with $m \geqslant n$ and Q from Algorithm 5.1, compute the approximate SVD $A \approx U\Sigma V^T$, where $U \in \mathbb{R}^{m\times k}$, $\Sigma \in \mathbb{R}^{k\times k}$, $V \in \mathbb{R}^{n\times k}$.

1: $B = Q^T A$
2: Compute the economy size SVD $B = \widetilde{U}\Sigma V^T$.
3: $U = Q\widetilde{U}$

Note that line 2 of Algorithm 5.1 requires about a factor n/k more flops than line 3, so most of the work is in the matrix multiplication of line 2.

We wish to bound the normwise absolute error in the approximate SVD from Algorithm 5.2. In exact arithmetic,

$$\|A - U\Sigma V^T\| = \|A - Q\widetilde{U}\Sigma V^T\| = \|A - QB\| = \|A - QQ^TA\|, \tag{5.1.1}$$

so it is only Algorithm 5.1, the rangefinder process, that introduces errors. Previous error analyses of randomized algorithms that provide bounds on the right-hand side of (5.1.1) (see, e.g., [17, sect. 11]) are for exact arithmetic, so to obtain practical error bounds one has to assume that floating-point arithmetic errors are swamped by the errors introduced by the randomization process, or account for their effect on (5.1.1). This is done in [2] for the Nyström approximation of a positive semidefinite matrix, with the goal of balancing the two sources of error and exploiting low-precision arithmetic.

Our goal is to perform a rounding error analysis of Algorithms 5.1 and 5.2. The algorithms as presented refer to the fixed-rank problem, where we have a user-specified value of k, the rank of Ω. In the other approach, the fixed-precision problem, we iteratively construct Q until the error (5.1.1) is less than some specified tolerance. We analyze both problems, beginning with the fixed-rank problem. Since the fixed-precision problem is essentially solved as a sequence of fixed-rank problems, our fixed-rank analysis proves crucial for our fixed-precision analysis.

In section 5.2 we give a rounding error analysis for Algorithms 5.1 and 5.2, which treat the fixed-rank problem. We then extend the analysis to a power iteration gen-

eralization of Algorithm 5.1. The power iteration analysis has a favourable rounding error term and avoids having to make complicated statements about the probability of the bounds holding, as unfortunately is the case for the basic method. This analysis is a usual worst-case analysis, but in section 5.2.2 we give a probabilistic rounding error analysis. In section 5.3 we extend our analysis to an algorithm for the fixed-precision problem, and we propose a mixed precision version of the algorithm. We give numerical experiments to assess the quality of the rounding error bounds and to indicate the possible benefits of the mixed precision algorithm. Concluding remarks are given in section 5.4.

5.2 ROUNDING ERROR ANALYSIS

If there are no rounding errors then the error in the computed SVD from Algorithm 5.2 is given by (5.1.1). We wish to derive an error bound that accounts for rounding errors. We will make the simplifying assumption that the SVD in line 2 of Algorithm 5.2 is computed exactly. This assumption has little effect on the final error bounds assuming that the SVD is computed by a numerically stable algorithm.

We will use the standard model of floating-point arithmetic [8, sect. 2.2]:

$$\mathrm{fl}(x \mathrel{\mathrm{op}} y) = (x \mathrel{\mathrm{op}} y)(1+\delta), \quad |\delta| \leqslant u, \tag{5.2.1}$$

with $\mathrm{op} \in \{+, -, \times, /\}$. Throughout we use the quantities

$$\gamma_n = \frac{nu}{1-nu}, \qquad \widetilde{\gamma}_n = \frac{cnu}{1-cnu}, \tag{5.2.2}$$

where c is a small integer constant and u is the unit roundoff.

We will use the 2-norm, $\|A\|_2 = \max_{x\neq 0} \|Ax\|_2/\|x\|_2$ and the Frobenius norm, $\|A\|_F = (\sum_{i,j} |a_{ij}|^2)^{1/2}$. We will also use the inequality, for $A \in \mathbb{C}^{r\times m}$, $B \in \mathbb{C}^{m\times n}$, $C \in \mathbb{C}^{n\times s}$ [13, 1991, Cor. 3.5.10],

$$\|ABC\|_F \leqslant \|A\|_2 \|B\|_F \|C\|_2. \tag{5.2.3}$$

We need the result that for $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times p}$, and $C = AB$ the computed product $\widehat{C}$ satisfies [8, sect. 3.5].

$$\widehat{C} = C + \Delta C, \quad |\Delta C| \leqslant \gamma_n |A||B|. \tag{5.2.4}$$

Let $\widehat{R} \in \mathbb{R}^{k\times k}$ be the computed upper triangular factor of $A \in \mathbb{R}^{m\times k}$ obtained by Householder QR factorization. Then there exists a matrix $\widetilde{Q} \in \mathbb{R}^{m\times k}$ with orthonormal columns such that [8, Thm. 19.4]

$$A + \Delta A = \widetilde{Q}\widehat{R}, \quad \|\Delta a_j\|_2 \leqslant \widetilde{\gamma}_{mk}\|a_j\|_2, \quad j = 1:k. \tag{5.2.5}$$

We can now obtain an error bound for step 2 of Algorithm 5.1

Lemma 5.1. *Let $Y = A\Omega$, where $A \in \mathbb{R}^{m\times n}$ with $m \geqslant n$ and $\Omega \in \mathbb{R}^{n\times k}$, where $k \leqslant n$. Householder QR factorization of the computed $\widehat{Y}$ produces a computed upper triangular $\widehat{R}$ satisfying*

$$Y + \Delta Y = \widetilde{Q}\widehat{R}, \quad \|\Delta y_j\|_2 \leqslant (\gamma_n + \widetilde{\gamma}_{mk} + \gamma_n\widetilde{\gamma}_{mk})\|A\|_F\|\omega_j\|_2, \quad j = 1:k, \tag{5.2.6}$$

where $\widetilde{Q} \in \mathbb{R}^{m\times k}$ has orthonormal columns and ω_j is the jth column of Ω.

Proof. From (5.2.4) we have $\widehat{Y} = Y + \Delta Y^{(1)}$, where

$$\|\Delta y_j^{(1)}\|_2 \leqslant \gamma_n \| \, |A||\omega_j| \, \|_2 \leqslant \gamma_n \|A\|_F \|\omega_j\|_2. \tag{5.2.7}$$

By (5.2.5), Householder QR factorization of $\widehat{Y}$ yields $\widehat{Y} + \Delta Y^{(2)} = \widetilde{Q}\widehat{R}$, where $\widetilde{Q}$ has orthonormal columns and

$$\|\Delta y_j^{(2)}\|_2 \leqslant \widetilde{\gamma}_{mk}\|\widehat{y}_j\|_2 \leqslant \widetilde{\gamma}_{mk}\big(\|y_j\|_2 + \gamma_n\|A\|_F\|\omega_j\|_2\big).$$

Writing $\Delta Y = \Delta Y^{(1)} + \Delta Y^{(2)}$ and using $\|y_j\|_2 \leqslant \|A\|_F\|w_j\|_2$ gives (5.2.6). □

We could explicitly form the matrix Q after computing the QR factorization in Algorithm 5.1 and then explicitly form the matrix product $B = Q^T A$ on step 1 of Algorithm 5.2. However, it is normal practice to keep Q in factored form and apply it to A in factored form. We will assume that this is how B is evaluated.

The error analysis of Householder QR factorization shows that the matrix $\widetilde{Q}$ in Lemma 5.1 is a product of exact Householder matrices that are defined in terms of the computed quantities that appear during the factorization. Moreover, if Q^T is applied in factored form to a matrix $H \in \mathbb{R}^{m\times n}$ to form $G = Q^T H$ then [8, Lem. 19.3] implies that the computed $\widehat{G}$ satisfies

$$\widehat{G} = \widetilde{Q}^T H + \Delta H, \quad \|\Delta h_j\|_2 \leqslant \widetilde{\gamma}_{mk}\|h_j\|_2, \quad j = 1:n. \tag{5.2.8}$$

Therefore the matrix $\widehat{B}$ computed on line 1 of Algorithm 5.2 using the factored form of Q satisfies

$$\widehat{B} = \widetilde{Q}^T A + \Delta A, \quad \|\Delta A\|_F \leqslant \widetilde{\gamma}_{mk}\|A\|_F. \tag{5.2.9}$$

The matrix $\widehat{U}$ from line 3 of Algorithm 5.2 computed using the factored form of Q satisfies

$$\widehat{U} = \widetilde{Q}\widetilde{U} + \Delta U, \quad \|\Delta U\|_F \leqslant \widetilde{\gamma}_{mk}\|\widetilde{U}\|_F = k^{1/2}\widetilde{\gamma}_{mk}. \tag{5.2.10}$$

Hence

$$\begin{aligned} A - \widehat{U}\Sigma V^T &= A - \widetilde{Q}\widetilde{U}\Sigma V^T - \Delta U\Sigma V^T \\ &= A - \widetilde{Q}\widehat{B} - \Delta U\Sigma V^T \\ &= (I - \widetilde{Q}\widetilde{Q}^T)A - \widetilde{Q}\Delta A - \Delta U\Sigma V^T \\ &= (I - \widetilde{Q}\widetilde{Q}^T)A + E, \end{aligned} \tag{5.2.11}$$

where

$$\begin{aligned} \|E\|_F &\leqslant \widetilde{\gamma}_{mk}\|A\|_F + k^{1/2}\widetilde{\gamma}_{mk}\|\Sigma V^T\|_F \\ &= \widetilde{\gamma}_{mk}\|A\|_F + k^{1/2}\widetilde{\gamma}_{mk}\|\widehat{B}\|_F \\ &\leqslant \widetilde{\gamma}_{mk}\|A\|_F + k^{1/2}\widetilde{\gamma}_{mk}(1+\widetilde{\gamma}_{mk})\|A\|_F \\ &= \widetilde{\gamma}_{mk}\big(1 + k^{1/2}(1+\widetilde{\gamma}_{mk})\big)\|A\|_F. \end{aligned} \tag{5.2.12}$$

We now have

$$A - \widehat{U}\Sigma V^T = (I - \widetilde{Q}\widetilde{Q}^T)A + E,$$

To bound $(I - \widetilde{Q}\widetilde{Q}^T)A$, we note that for any X of suitable size,

$$\|(I - \widetilde{Q}\widetilde{Q}^T)A\|_F \leqslant \|(I - \widetilde{Q}\widetilde{Q}^T)A\Omega X\|_F + \|(I - \widetilde{Q}\widetilde{Q}^T)A(I - \Omega X)\|_F. \tag{5.2.13}$$

For the first term we have $(I - \widetilde{Q}\widetilde{Q}^T)A\Omega = -(I - \widetilde{Q}\widetilde{Q}^T)\Delta Y$ by (5.2.6). Using (5.2.3), together with $\|I - \widetilde{Q}\widetilde{Q}^T\|_2 \leqslant 1$ we then have

$$\|(I - \widetilde{Q}\widetilde{Q}^T)A\Omega\|_F \leqslant \|\Delta Y\|_F, \tag{5.2.14}$$

and so

$$\|(I - \widetilde{Q}\widetilde{Q}^T)A\Omega X\|_F = \| - (I - \widetilde{Q}\widetilde{Q}^T)\Delta Y X\|_F \leqslant \|\Delta Y\|_F\|X\|_F.$$

For the second term, choose $X = (V^T\Omega)^\dagger V^T$, where V contains the first r right singular vectors of A such that $r + 4 \leqslant k$. We can then obtain, assuming $V^T\Omega$ has full rank and using $V^T\Omega(V^T\Omega)^\dagger = I_r$,

$$\begin{aligned}\|(I - \widetilde{Q}\widetilde{Q}^T)A(I - \Omega X)\|_F^2 &\leqslant \|A(I - \Omega(V^T\Omega)^\dagger V^T)\|_F^2 \\ &= \|A(I - VV^T)(I - \Omega(V^T\Omega)^\dagger V^T)\|_F^2 \\ &\leqslant \|A(I - VV^T)\|_F^2 + \|A(I - VV^T)\Omega(V^T\Omega)^\dagger\|_F^2 \\ &= \|\Sigma_2\|_F^2 + \|\Sigma_2(V_\perp^T\Omega)(V^T\Omega)^\dagger\|_F^2,\end{aligned}$$

with Σ_2 containing singular values $\sigma_{r+1}, \sigma_{r+2}, \ldots$ and $V_\perp$ the corresponding singular vectors. We are left with

$$\|(I - \widetilde{Q}\widetilde{Q}^T)A\|_F \leqslant \|\Delta Y\|_F\|(V^T\Omega)^\dagger\|_F + \sqrt{\|\Sigma_2\|_F^2 + \|\Sigma_2(V_\perp^T\Omega)(V^T\Omega)^\dagger\|_F^2}. \tag{5.2.15}$$

To proceed, two key observations from [6, sec. 10] are needed. Firstly, we can bound, both in expectation and probabilistically, the 2 and Frobenius norms of the pseudoinverse of a Gaussian matrix. Secondly, as the Gaussian distribution is rotationally invariant, the matrices $V^T\Omega$ and $V_\perp^T\Omega$ are also Gaussian matrices, and as such their norms can be bounded by those results on pseudoinverses.

The second term in (5.2.15) is independent of u and corresponds to the exact bounds present in [6, Thm. 9.1] and, on incorporating the bounds on $V_\perp^T\Omega$ and $(V^T\Omega)^\dagger$, will be equal to those results up to a constant. The first term, which bounds the rounding errors, is what we are interested in. Note that $(V^T\Omega)^\dagger \in \mathbb{R}^{r\times k}$ and so by [6, Prop. 10.4], it is bounded probabilistically in the Frobenius norm by a constant of order $\sqrt{k}$. We know from (5.2.6) that $\|\Delta Y\|_F \leqslant (\gamma_n + \widetilde{\gamma}_{mk} + \gamma_n\widetilde{\gamma}_{mk})\|A\|_F\|\Omega\|_F$. From [21, Thm. 4.1.1], we can bound $\|\Omega\|_2 \leqslant t\sqrt{n}$, where the bound holds with at least some probability that depends on t, k and n. This probability is extremely close to 1 for any realistic choice of n and k and a very modest t. Using norm relations we have

$\|\Omega\|_F \leqslant \sqrt{k}\|\Omega\|_2 = O(\sqrt{kn})$. Combining all this we find that the final rounding error contribution can be bounded by

$$\|\Delta Y\|_F \|(V^T\Omega)^\dagger\|_F \leqslant O(mk^{3/2}u)\|A\|_F\|\Omega\|_F \leqslant O(m\sqrt{n}k^2u)\|A\|_F,$$

As the leading order term of E is $O(mku)$, we have from the above that

$$\|A - \widehat{U}\widehat{\Sigma}\widehat{V}^T\|_F \leqslant O(m\sqrt{n}k^2u)\|A\|_F + \sqrt{\|\Sigma_2\|_F^2 + \|\Sigma_2(V_\perp^T\Omega)(V^T\Omega)^\dagger\|_F^2}, \tag{5.2.16}$$

the left term corresponding to the rounding error contribution and the right hand term the randomization errors. For a more precise bound, one would have to incorporate statements of probability from the bounded random matrices. We avoid doing this here because, as we will see in the next section, the analysis of the power method is more interpretable while also providing a favourable rounding error contribution, meaning it is the analysis we focus on for the remainder of this chapter.

In practice, structured random matrices are often used instead of a Gaussian Ω. As an example we consider the commonly used subsampled random Fourier transform (SRFT) [6, sect. 4.6]. An SRFT matrix has the form

$$\Omega = \sqrt{\frac{n}{k}}DFR, \tag{5.2.17}$$

where

- $D = \operatorname{diag}(z_1, z_2, \ldots, z_n) \in \mathbb{C}^{n\times n}$, with each $z_i \in \mathbb{C}$ an independent random variable uniformly distributed on the complex unit circle,
- the unitary matrix $F \in \mathbb{C}^{n\times n}$ is the discrete Fourier matrix with entries

$$f_{jk} = n^{-1/2}\exp(-2\pi\mathrm{i}(j-1)(k-1)/n),$$

- R is an $n \times k$ matrix whose columns are sampled uniformly at random from the $n \times n$ identity matrix without replacement.

We expect rounding error bounds for the basic method using SRFT or other structured matrices to be attainable, using a similar method to that for obtaining the Gaussian matrix bound, but requiring results about the particular random matrix, such as in [6, sec. 11]. We will see in the next section that another advantage of our analysis of the power method is that the results will hold for any choice of random Ω.

5.2.1 Extension to power iteration

If the singular values of A decay slowly then the accuracy of Algorithm 5.1 in exact arithmetic degrades. To address this issue a power scheme that generalizes Algorithm 5.1 was proposed in [6, sect. 4.5], which we reproduce in Algorithm 5.3.

Algorithm 5.3 Given $A \in \mathbb{R}^{m\times n}$ with $m \geqslant n$, and integers $k < n$ and q, compute a matrix $Q \in \mathbb{R}^{m\times k}$ with orthonormal columns such that $A \approx QQ^TA$.

1: Draw a Gaussian matrix $\Omega \in \mathbb{R}^{n\times k}$.
2: $Y = (AA^T)^qA\Omega$
3: Compute an orthonormal basis $Q \in \mathbb{R}^{m\times k}$ for Y via a thin QR factorization $Y = QR$.

In floating-point arithmetic, the power scheme given in Algorithm 5.3 fails to capture singular vectors associated with singular values that are small relative to $\|A\|_2$, so a modified variant of this algorithm given in Algorithm 5.4 is usually considered.

Algorithm 5.4 Given $A \in \mathbb{R}^{m\times n}$ with $m \geqslant n$, and integers $k < n$ and q, compute a matrix $Q \in \mathbb{R}^{m\times k}$ with orthonormal columns such that $A \approx QQ^TA$. All the QR factorizations used are thin QR factorizations.

1: Draw a Gaussian matrix $\Omega \in \mathbb{R}^{n\times k}$.
2: Compute $Y_1 = A\Omega$ and factorize $Y_1 = Q_1R_1$.
3: **for** $i = 1 : q$ **do**
4: Compute $Y_{2i} = A^TQ_{2i-1}$ and factorize $Y_{2i} = Q_{2i}R_{2i}$.
5: Compute $Y_{2i+1} = AQ_{2i}$ and factorize $Y_{2i+1} = Q_{2i+1}R_{2i+1}$.
6: **end for**
7: $Q = Q_{2q+1}$

In Algorithm 5.4, $q = 1$ or $q = 2$ is usually sufficient for most practical problems [6, sect. 1.6]. However, here we perform analysis for the general case, and extend the analysis of section 5.2 for Algorithm 5.4, to bound $\|A - \widehat{U}\widehat{\Sigma}\widehat{V}^T\|_F$. Introducing the exact Q, we can write

$$A - \widehat{U}\widehat{\Sigma}\widehat{V}^T = (I - QQ^T)A + (QQ^T - \widetilde{Q}\widetilde{Q}^T)A + E \tag{5.2.18}$$

and so

$$\|A - \widehat{U}\widehat{\Sigma}\widehat{V}^T\|_F \leqslant \|(I - QQ^T)A\|_F + \|(QQ^T - \widetilde{Q}\widetilde{Q}^T)A\|_F + \|E\|_F, \tag{5.2.19}$$

where $\|(I - QQ^T)A\|_F$ is bounded by the analysis in [6, sect. 9.3], and $\|E\|_F$, which is the error term that arises from Algorithm 5.2 once we have computed Q via Algorithm 5.4, is bounded by (5.2.12). The matrix $\widetilde{Q}$ is the exact Q that arises, as defined in (5.2.5), from the final QR factorization of Algorithm 5.4.

The key observation is that we can identify the factorization $Q_{2q+1}R_{2q+1} = Y_{2q+1} = AQ_{2q}$ in Algorithm 5.4 with the factorization $QR = Y = A\Omega$ in Algorithm 5.1. By a similar proof to that of Lemma 5.1 we have

$$Y_{2q+1} + \Delta Y_{2q+1} = \widetilde{Q}\widehat{R}_{2q+1}, \quad \|\Delta Y_{2q+1}\|_F \leqslant k^{1/2}(\gamma_n + \widetilde{\gamma}_{mk} + \gamma_n\widetilde{\gamma}_{mk})\|A\|_F \tag{5.2.20}$$

for a matrix $\widetilde{Q} \in \mathbb{R}^{m\times k}$ with orthonormal columns.

Purely for the purposes of the error analysis, we now assume that all the QR factorizations in Algorithm 5.4 are full QR factorizations with $Q_j \in \mathbb{R}^{m\times m}$ and $R_j = \left[\begin{smallmatrix} R'_j \\ 0 \end{smallmatrix}\right]$ with $R'_j \in \mathbb{R}^{k\times k}$, and we set $Q = Q_{2q+1}I_{m,k} \in \mathbb{R}^{m\times k}$, where $I_{m,k} = I(:, 1:k) = \left[\begin{smallmatrix} I_k \\ 0 \end{smallmatrix}\right] \in \mathbb{R}^{m\times k}$. We have

$$\begin{aligned} QQ^T Y_{2q+1} &= Q_{2q+1}I_{m,k}I_{m,k}^T Q_{2q+1}^T Q_{2q+1}R_{2q+1} \\ &= Q_{2q+1}I_{m,k}I_{m,k}^T \begin{bmatrix} R'_{2q+1} \\ 0 \end{bmatrix} \\ &= Q_{2q+1}\begin{bmatrix} R'_{2q+1} \\ 0 \end{bmatrix} \\ &= Q_{2q+1}R_{2q+1} = Y_{2q+1}. \end{aligned} \tag{5.2.21}$$

Then, with $F = QQ^T - \widetilde{Q}\widetilde{Q}^T$, using (5.2.20) and (5.2.21) we have

$$\begin{aligned} FY_{2q+1} &= QQ^T Y_{2q+1} - \widetilde{Q}\widetilde{Q}^T Y_{2q+1} = (I - \widetilde{Q}\widetilde{Q}^T)Y_{2q+1} \\ &= -(I - \widetilde{Q}\widetilde{Q}^T)\Delta Y_{2q+1}. \end{aligned} \tag{5.2.22}$$

Combining (5.2.20) and (5.2.22) gives

$$\begin{aligned} \|(QQ^T - \widetilde{Q}\widetilde{Q}^T)A\|_F &= \|FA\|_F = \|FAQ_{2q}\|_F = \|FY_{2q+1}\|_F \\ &\leqslant \|\Delta Y_{2q+1}\|_F \leqslant k^{1/2}(\gamma_n + \widetilde{\gamma}_{mk} + \gamma_n\widetilde{\gamma}_{mk})\|A\|_F. \end{aligned} \tag{5.2.23}$$

Using this bound in (5.2.19) gives the next result.

Theorem 5.2. *Let $A \in \mathbb{R}^{m\times n}$, where $m \geqslant n$, and assume that the SVD on line 2 of Algorithm 5.2 is computed exactly. Algorithm 5.4, with $q \geqslant 1$, and Algorithm 5.2 produce a computed SVD $A \approx \widehat{U}\widehat{\Sigma}\widehat{V}^T$ satisfying*

$$\begin{aligned} \|A - \widehat{U}\widehat{\Sigma}\widehat{V}^T\|_F &\leqslant \|(I - QQ^T)A\|_F \\ &\quad + \left((1 + k^{1/2})\widetilde{\gamma}_{mk} + k^{1/2}\gamma_n(1 + \widetilde{\gamma}_{mk}) + k^{1/2}\widetilde{\gamma}_{mk}^2\right)\|A\|_F. \end{aligned} \tag{5.2.24}$$

We note that results are given in [6, sect. 10.4] bounding the expected value of the 2-norm approximation error $\|(I - QQ^T)A\|_2$, with deviation bounds easy to obtain. Results for the Frobenius norm can be obtained by using $\|B\|_F \leqslant \sqrt{\text{rank}(B)}\|B\|_2$.

Note the leading order rounding error contribution is $mk^{3/2}u$ compared with $mn^{1/2}k^2u$ in (5.2.16). The smaller constant in (5.2.24) compared with (5.2.16) is attributable to the fact that the final QR factorization in Algorithm 5.4 is of AQ_{2q+1} rather than of $A\Omega$ as in Algorithm 5.1, and we have exploited the orthogonality of Q_{2q+1}.

5.2.2 Probabilistic rounding error analysis

Recent results in probabilistic error analysis provide error bounds that are both tighter and more indicative of the typical error growth than worst-case bounds [3], [4], [10], [11], [14]. Worst-case error bounds of the form $f(n)u$ translate to probabilistic bounds of the form $\sqrt{f(n)}u$ under the assumption that the rounding errors are mean independent random variables of mean zero. Results of this form hold for inner products, matrix-vector and matrix-matrix products, LU factorization, triangular systems, Cholesky factorization, and QR factorization. Crucially for this work, these results hold for the two central kernels of Algorithms 5.1 and 5.2.

For matrix multiplication the probabilistic analogue of (5.2.4) is given by [4, Thm. 5.9], [10, Thm. 3.4]

$$\widehat{C} = C + \Delta C, \quad |\Delta C| \leqslant \overline{\gamma}_n(\lambda)|A||B|, \tag{5.2.25}$$

where

$$\overline{\gamma}_n(\lambda) := \exp\left(\frac{\lambda\sqrt{n}u + nu^2}{1-u}\right) - 1 = \lambda\sqrt{n}u + O(u^2). \tag{5.2.26}$$

Here, and below, $\lambda > 0$ is a constant that can be freely chosen and controls the probability of failure of the bound, which is a monotonically decreasing function of λ. We do not state probabilities of failure, as they are very close to 1 even for modest λ and they are also pessimistic; see the cited references for the details.

Similar probabilistic analogues apply for Householder QR factorization and the multiplication of a matrix by a sequence of Householder matrices. The analogue to (5.2.5) is given by [3, Thm. 4.4]

$$A + \Delta A = \widetilde{Q}\widehat{R}, \quad \|\Delta a_j\|_2 \leqslant c\lambda\sqrt{k}\overline{\gamma}_m(\lambda)\|a_j\|_2 + O(u^2), \quad j = 1:n, \tag{5.2.27}$$

where c is a modest integer constant. The same reduction in the error constant applies to (5.2.8); see [3, Lem. 4.3].

Using these results, the worst-case analysis extends straightforwardly to the probabilistic case, giving the following analogue of Theorem 5.3. For the sake of readability, we include a more detailed analysis than the outline given here in Appendix 5.A at the end of this chapter. We write "under the assumptions of probabilistic error analysis" to mean that rounding errors are mean independent random variables of mean zero and that a technical assumption in [3, Lem. 4.2], required for QR factorization, holds.

Theorem 5.3. *Let $A \in \mathbb{R}^{m\times n}$, where $m \geqslant n$, and assume that the SVD on line 2 of Algorithm 5.2 is computed exactly. In floating-point arithmetic and under the assumptions of probabilistic error analysis, Algorithm 5.4 and Algorithm 5.2 produce a computed SVD $A \approx \widehat{U}\widehat{\Sigma}\widehat{V}^T$ satisfying*

$$\begin{aligned}\|A - \widehat{U}\widehat{\Sigma}\widehat{V}^T A\|_F \leqslant \|A - QQ^T A\|_F \\ + \left((k + k^{1/2})\overline{\gamma}_m(\lambda) + k^{1/2}\overline{\gamma}_n(\lambda)\right)\|A\|_F + O(u^2).\end{aligned}$$

Table 5.2.1: Leading order rounding error terms in the bounds.

	Worst case	Probabilistic
Algorithms 5.2 and 5.1 (standard, Gaussian Ω)	$mn^{1/2}k^2u$	$(mn)^{1/2}k^{3/2}u$
Algorithms 5.2 and 5.4 (power iteration)	$mk^{3/2}u$	$m^{1/2}ku$

We note that the leading rounding error contribution is $\sqrt{mk}u$ for the probabilistic case, in comparison to $mk^{3/2}u$ in Theorem 5.2.

Probabilistic error analysis can also be applied to the analysis of the basic method, reducing the leading order rounding error contribution from $mn^{1/2}k^2u$ to $(mn)^{1/2}k^{3/2}u$. In Table 5.2.1, we compare the leading order error terms of the two methods and the two types of analysis. The more favorable algorithm in terms of rounding error bound is the power iteration.

5.2.3 Numerical experiments

We present some numerical experiments to test the sharpness of the worst-case error bound (5.2.16) and its probabilistic variant. The leading order rounding error contributions for each are given in Table 5.2.1. We first need the result [6, Thm. 10.7] that for Gaussian Ω the bound

$$\|A - QQ^TA\|_F \leqslant \psi_1(k,p,t)\left(\sum_{j>k}\sigma_j^2\right)^{1/2} + \psi_2(k,p,t,s)\sigma_{k+1}, \tag{5.2.28}$$

holds with probability at least $1-(2t^{-p}+\mathrm{e}^{-s^2/2})$, where k is the target rank, and the functions ψ_1 and ψ_2 are given by

$$\psi_1(k,p,t) = 1 + t\left(\frac{3k}{p+1}\right)^{1/2}, \quad \psi_2(k,p,t,s) = st\frac{\mathrm{e}\sqrt{k+p}}{p+1}. \tag{5.2.29}$$

For even modest choices of these parameters the probability of failure is negligible. Choosing for example $p = t = s = 5$ results in a probability of failure of 6×10^{-4}.p We simply choose to set $p = t = s = 1$, as we have observed that the probabilities associated with these parameters are pessimistic. We use (5.2.28) for the exact error term in (5.2.16). For the rounding error terms contained in (5.2.16), we simply plot the leading order terms and set any non-dimensional constants to 1.

Note the parameter p above is an oversampling parameter, so in the bounds given in (5.2.16) we should strictly make the substitution $k \leftarrow k + p$. In Figure 5.2.1 we do not include the contribution from the oversampling parameter because the p we have chosen is small enough to make no material effect to the displayed bounds.

We use two types of test matrix, which were also used in [23]. In this section we just consider Algorithms 5.1 and 5.2. We do not consider the power iteration of section 5.2.1 as an error bound analogous to (5.2.28) is not available [6, sect. 10.4]. All matrices used in the tests are square ($m = n$); we have run tests with $m \gg n$ and obtained similar results. The types are as follows.

- Type 1: low rank plus noise. Define $D = \mathrm{diag}(I_r, 0) \in \mathbb{R}^{n \times n}$. Then

$$A = D + (\xi/n)GG^T,$$

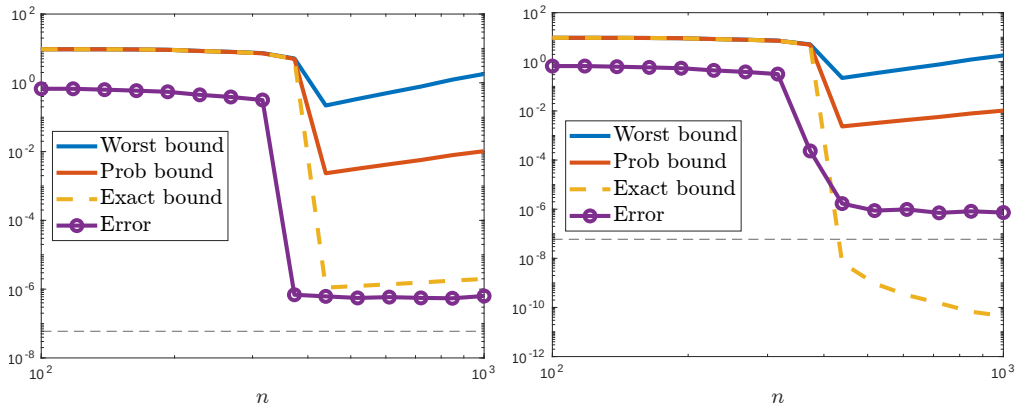
  with G a Gaussian matrix. We fix $\xi = 10^{-4}$ and $r = 20$.

- Type 2: polynomial decay. We generate random orthogonal matrices $U, V \in \mathbb{R}^{n \times n}$ from the Haar distribution [20] and define

$$A = UDV^T, \qquad D = \mathrm{diag}(\phi I_r, 2^{-\alpha}, 3^{-\alpha}, \ldots, (n - r + 1)^{-\alpha}). \tag{5.2.30}$$

  Throughout $r = 20$, $\alpha = 3$, and $\phi = 10^6$.

The experiments are run in MATLAB R2021a. All steps of the algorithms are performed in IEEE single precision, with reference quantities computed in IEEE double



(a) Type 1 matrix with $k = \sqrt{n}$.

(b) Type 2 matrix with $k = \sqrt{n}$.

Figure 5.2.1: Numerical experiments performed in fp32. The dashed line in both figures is $u = 2^{-24} \approx 6 \times 10^{-8}$, the unit roundoff for single precision.

precision. Test matrices are also rounded to single precision. The results are plotted in Figure 5.2.1, in which "Worst bound" denotes the worst-case rounding error bound from (5.2.16), "Prob bound" denotes the probabilistic bound from Table 5.2.1, "Exact bound" denotes the bound in (5.2.28), and "Error" is the approximation error $\|A - \widehat{U}\widehat{\Sigma}\widehat{V}^T\|_F/\|A\|_F$.

All of the error curves in Figure 5.2.1 exhibit the same phenomenon: at a certain value of n the error decreases drastically. This is due to the experimental setup, where we have set $k = \sqrt{n}$. Each of the test matrices has a singular value spectrum that is flat for the largest singular values, but then drops off quickly past a certain point. The large decrease in the error curves corresponds to the point when k has reached a value large enough that the singular values we are approximating are past this dropoff point in the singular value spectrum of the test matrices.

We see that for Type 1 matrices the error satisfies all the bounds, with the probabilistic bound being significantly closer to the exact bound than the worst-case bound. In the case of Type 2 matrices, however, as the exact bound becomes less than the unit roundoff, it no longer becomes a reliable indicator for the computed error. In this case our probabilistic bound again bounds the error.

5.3 FIXED-PRECISION ALGORITHMS

Up to now, we have discussed fixed-rank problems, where we specify a target rank a priori. Perhaps a more common situation computationally is the fixed-precision problem, in which we specify a tolerance to which we want our computed approximation to be accurate. In Algorithm 5.5 we display the basic fixed-precision rangefinder algorithm proposed by Martinsson and Voronin [18, Fig. 4]. Here we essentially solve the fixed-rank problem multiple times with the rank k chosen to be some block size b and iteratively construct a basis matrix until the specified tolerance has been met. We consider the specified tolerance ε to be a *relative* tolerance.

Algorithm 5.5 Given $A \in \mathbb{R}^{m\times n}$ with $m \geqslant n$ and a tolerance $\varepsilon > 0$, this algorithm computes a matrix $Q \in \mathbb{R}^{m\times b}$ with orthonormal columns and $B \in \mathbb{R}^{b\times n}$ such that $\|A - QB\|_F/\|A\|_F \leqslant \varepsilon$. The parameter b is a block size and q determines the number of power iterations in the inner loop.

$Q = [\,]$, $B = [\,]$, $A_1 = A$, $\rho_1 = 1$
for $i = 1 : \texttt{its}_{\texttt{max}}$ **do**
  Draw a Gaussian matrix $\Omega \in \mathbb{R}^{n\times b}$.
  $Y = A_i\Omega$
  $Q_i = \texttt{qr}(Y, 0)$
  for $j = 1 : q$ **do**
    Compute $Y = A_i^T Q_i$ and $Q_i = \texttt{qr}(Y, 0)$.
    Compute $Y = A_i Q_i$ and $Q_i = \texttt{qr}(Y, 0)$.
  end for
  $Q_i = \texttt{qr}(Q_i - \sum_{j=1}^{i-1} Q_j Q_j^T Q_i, 0)$
  $B_i = Q_i^T A_i$
  $A_{i+1} = A_i - Q_i B_i$
  $\rho_i = \|A_{i+1}\|_F/\|A\|_F$
  If $\rho_i \leqslant \varepsilon$ then quit.
end for
$Q = [Q_1 \ \cdots \ Q_i]$, $B = [B_1^T \ \cdots \ B_i^T]^T$

In the algorithms of this section $\texttt{qr}(\cdot, 0)$ returns the orthonormal factor from the thin QR factorization. We compute the factorization $A \approx QB$, with $B = Q^T A$. Once Q and B are available, further factorizations, such as the SVD or low rank QR, are easily computed [18, Rem. 1]. The main focus of this section is on the iterative construction of Q and B.

An alternative algorithm is proposed in [25, Alg. 2]. This is the algorithm implemented by the MATLAB `svdsketch` function[1]. We know from [25, Prop. 1] that when executed in exact arithmetic [25, Alg. 2] and Algorithm 5.5 are identical. In floating-point arithmetic, there are two main differences. The algorithm [25, Alg. 2] performs the same operations for computing Q_i as Algorithm 5.5, but in a different order. To analyse [25, Alg. 2] we would need to modify the analysis of Section 5.2 to account for this changed order. The error ρ_i is also calculated differently, which allows one to avoid retaining and computing the Frobenius norm of $\|A_{i+1}\|_F$ at each iteration, as is done in Algorithm 5.5. This new method of calculating the error introduces a limitation on the accuracy of the computation [25, sec. 3.3]. In the remainder of this work we focus on Algorithm 5.5, with the expectation that our analysis can be adapted to [25, Alg. 2].

5.3.1 Error analysis of fixed-precision algorithm

Here we present a framework for analyzing the effect of rounding errors on Algorithm 5.5. In the next section we use this analysis to motivate mixed precision algorithms for these problems. Our primary interest is how the influence of rounding errors limits the tolerance that we can set in these algorithms.

On a given iteration, we compute $\widehat{Q}_i$ and $\widehat{B}_i = \widehat{Q}_i^T A$ and we are interested in the error $\|A - \widehat{Q}_i \widehat{B}_i\|_F$. Here, we assume for the sake of the analysis that the fixed-precision iterations have $q > 0$, meaning we incorporate the power iteration. The reasoning we will apply is also valid for the basic method, but we make this choice as our power iteration error analysis results are more formal and it is what we use in the experiments to follow. From either Theorem 5.2 or Theorem 5.3 , we know that, to first order in u,

$$\|A - \widehat{Q}_i \widehat{Q}_i^T A\|_F \leqslant \|A - Q_i B_i\|_F + u f(m,n,b)\|A\|_F, \tag{5.3.1}$$

1 https://uk.mathworks.com/help/matlab/ref/svdsketch.html

where Q_i and B_i are the exact matrices and the form of f depends on whether the bound is worst-case or probabilistic. As the computation proceeds, A is updated and at each iteration the quantity ρ_i (Line 13 of Algorithm 5.5) serves as our relative error. Define

$$\widehat{P}_i = I - \widehat{Q}_i \widehat{Q}_i^T, \qquad P_i = I - Q_i Q_i^T,$$

In exact arithmetic, the error after t iterations of the outer loop in Algorithm 5.5 is given by

$$\rho_t = \frac{\|P_t \dots P_2 P_1 A\|_F}{\|A\|_F}. \tag{5.3.2}$$

In floating-point arithmetic it is given by

$$\widehat{\rho}_t = \frac{\|\widehat{P}_t \dots \widehat{P}_2 \widehat{P}_1 A\|_F}{\|A\|_F}. \tag{5.3.3}$$

Bounding the difference between (5.3.2) and (5.3.3) will allow us to determine the impact of rounding errors on the iterative algorithm.

We have $\widehat{P}_i A - P_i A = (\widehat{Q}_i \widehat{Q}_i^T - Q_i Q_i^T)A$ which we can bound from the analysis of Section 5.2. This term amounts to the $\|FA\|_F$ term we bounded in (5.2.23). In the case of both Theorems 5.2 and 5.3, the contribution of the $\|FA\|_F$ term is what gives the leading order rounding error term. For the sake of simplicity then, we write

$$\widehat{P}_i A = P_i A + \Delta_i, \quad \|\Delta_i\|_F \leqslant u f(m,n,b) \|A\|_F, \tag{5.3.4}$$

where, as described in (5.3.1), the precise form of f depends on the specific analysis deployed. Then

$$\begin{aligned}
\widehat{P}_t \dots \widehat{P}_2 \widehat{P}_1 A &= \widehat{P}_t \dots \widehat{P}_2 (P_1 A + \Delta_1) \\
&= \widehat{P}_t \dots \widehat{P}_3 (P_2 (P_1 A + \Delta_1) + \Delta_2) \\
&= \dots = P_t \dots P_1 A + \sum_{i=1}^{t-1} P_t \dots P_{i+1} \Delta_i + \Delta_t,
\end{aligned}$$

so

$$\widehat{\rho}_t \leqslant \rho_t + \frac{\left\|\sum_{i=1}^{t-1} P_t \dots P_{i+1}\Delta_i + \Delta_t\right\|_F}{\|A\|_F} \leqslant \rho_t + \sum_{i=1}^{t} \|\Delta_i\|_F/\|A\|_F.$$

Finally, we have

$$\widehat{\rho}_t \leqslant \rho_t + tuf(m,n,b), \tag{5.3.5}$$

where the general form of (5.3.5) is the same as that of the bounds of section 5.2: our computed error is bounded by the exact error plus a term involving the unit roundoff, and some function of the problem dimensions. The dependence on $\|A\|_F$ has disappeared as we are now considering the relative error. If the exact error ρ_t is less than or equal to $tuf(m,n,b)$ it is possible that the overall error will be dominated by rounding errors. This could in turn cause the algorithm not to converge as expected.

Note that in exact arithmetic, the approximation error of Algorithm 5.5 is identical to that of Algorithm 5.1 [18, sect. 4]. Therefore, after t iterations of Algorithm 5.5 with block size b, we can identify ρ_t with the relative approximation error of Algorithm 5.1 with $\Omega \in \mathbb{R}^{n\times k}$ where $k = bt$.

5.3.2 Mixed precision rangefinder

We saw in the previous section how accuracy guarantees of fixed-precision problems can be affected by the choice of precision and the problem size. Here we motivate the use of low precision in these algorithms.

We use the probabilistic bounds given in section 5.2.2 to guide how to deploy low precisions. Algorithm 5.6 describes this approach. We update A_i, beginning in high precision, and switch to lower precisions once $\|A_i\|_F$ is sufficiently reduced, corresponding to a smaller relative error ρ_i.

Algorithm 5.6 Given $A \in \mathbb{R}^{m\times n}$ with $m \geqslant n$ and a relative tolerance $\varepsilon > 0$, this algorithm computes a matrix $Q \in \mathbb{R}^{m\times tb}$ with orthonormal columns such that $\|A - QQ^TA\|_F \leqslant \varepsilon$. The parameter b is a block size and q determines the number of power iterations in the inner loop. A sequences of precisions $u_1 < u_2 < \cdots < u_p$ and tolerances $1 > \varepsilon_1 > \cdots > \varepsilon_p = \varepsilon$ are given.

$Q = [\,]$, $B = [\,]$, $A_1 = A$, $\rho_1 = 1$
for $i = 1 : \mathtt{its}_{\max}$ **do**
  Draw a Gaussian matrix $\Omega \in \mathbb{R}^{n\times b}$.
  Find the largest j, $1 \leqslant j \leqslant p$ such that $\rho_i < \varepsilon_j$.
  $Y = A_i\Omega$ at precision u_j.
  $Q_i = \mathtt{qr}(Y, 0)$ at precision u_j
  for $j = 1 : q$ **do**
    Compute $Y = A_i^TQ_i$ and $Q_i = \mathtt{qr}(Y, 0)$ at precision u_j.
    Compute $Y = A_iQ_i$ and $Q_i = \mathtt{qr}(Y, 0)$ at precision u_j.
  end for
  Reorthonormalize Q_i at precision u_1.
  $B_i = Q_i^TA_i$ at precision u_j.
  $A_{i+1} = A_i - Q_iB_i$ at precision u_j.
  $\rho_i = \|A_{i+1}\|_F/\|A\|_F$
  If $\rho_i \leqslant \varepsilon$ then quit.
end for
$Q = [Q_1 \ \cdots \ Q_i]$, $B = [B_1^T \ \cdots \ B_i^T]^T$

The primary difficulty in this algorithm is determining when to switch to lower precision. We use a combination of the rounding error bounds from previous sections and a user-specified parameter to set these tolerances. In the problem setup we have a global tolerance ε, a sequence of available precisions $u_j, j = 1:p$, an $m \times n$ matrix A, and a block size b. Assume we perform power iterations, so $q > 0$. The basic idea is simple: for t iterations at precision u_j, if $t\sqrt{mb}u_j\rho_t$, from (5.3.5) with $f(m,n,b)$ chosen to be $\sqrt{mb}$ from Table 5.2.1, is significantly less than ε then we know that the contribution of rounding errors will be negligible compared with the algorithmic error, and we can safely use precision u_j. If we had a priori knowledge of the number of iterations that would be performed at precision u_j, we could compare these quantities to ε and decide whether the use of precision u_j is appropriate. As we do not know t, we must set a user-specified parameter θ, so our quantity of interest is now $\theta\sqrt{mb}u_j\rho_t$. The parameter θ helps to account for the role played by the unknown

number of iterations, but also allows the user to incorporate a degree of optimism or pessimism in the algorithm. We then make the simple choice

$$\varepsilon_j = \begin{cases} \varepsilon/(\theta\sqrt{m}bu_{j+1}), & j = 1:p-1, \\ \varepsilon, & j = p. \end{cases} \tag{5.3.6}$$

This choice means that if $\rho_t < \varepsilon_j$, when we switch to precision u_{j+1} we know that $\theta\sqrt{m}bu_{j+1}\rho_t < \varepsilon$. The parameter θ controls by how much we want to ensure that the leading rounding error contribution is less than ε. The larger the value of θ, the more certain we can be that rounding errors will not swamp the algorithmic error. The smaller the value of θ, the more optimistic we are about the impact of rounding errors, and the earlier the switch to lower precisions.

5.3.3 Reorthonormalization

The reorthonormalization steps in Line 10 of Algorithm 5.5 and Line 11 of Algorithm 5.6 are performed in order to maintain orthonormality among the columns of the computed Q. Preserving orthonormality is important as any subsequent uses for Q will have the assumption that Q has orthonormal columns. Taking for example the computation of the randomized SVD in Algorithm 5.2, if Q loses orthonormality then in Line 3 the resultant U will also lack orthonormality. To ensure orthonormality we reorthonormalize each Q at each iteration in the highest used precision.

In Algorithm 5.6 we have some specified accuracy tolerance ε, a sequence of available precisions $u_1 < u_2 < \cdots < u_p$, and we orthonormalize at the highest precision, u_1. The matrix $\widehat{Q}_i$ that we orthonormalize has been computed at some precision u_j and so $\|\widehat{Q}_i - Q_i\|$ will be of order u_j. Orthonormalizing at precision u_1, to obtain $\widetilde{Q}_i$, ensures that $\|\widetilde{Q}_i^T\widetilde{Q}_i - I\|$ is of order u_1, but $\|\widehat{Q}_i - \widetilde{Q}_i\|$ will still be of order u_j. This means that our accuracy is limited by the lowest precision used, as we always have a term of order u_j in the error bound (5.3.5). For this reason, we only allow precisions

to be used in Algorithm 5.6 which have a unit roundoff less than the specified relative tolerance.

From the point of view of the error analysis of section 5.2, the orthonormalization step simply changes the constant slightly in (5.2.10) and subsequent bounds. It does not change the form of the final bounds, so Theorems 5.2 and 5.3 remain valid with orthonormalization.

5.3.4 Numerical experiments

We now test the performance of Algorithm 5.6. For various test matrices we compute the matrices Q and B. We then compute the approximate SVD $A \approx \widehat{U}\widehat{\Sigma}\widehat{V}^T$ as described in Algorithm 5.2. We use the error measure $\|A - \widehat{U}\widehat{\Sigma}\widehat{V}^T\|_F/\|A\|_F$. The test matrices used are described below. Throughout we set the block size $b = 10$ and set $q = 1$ in the power iterations. We use an implementation of Algorithm 5.6 in which we have three available precisions: fp64 (IEEE double precision), fp32 (IEEE single precision), and fp16 (IEEE half precision), with respective unit roundoffs 2^{-53}, 2^{-24}, and 2^{-11}. We compare this to Algorithm 5.5 run entirely in fp64. The subsequent computation of the SVD is done in fp64 in both implementations.

For double and single precision we use the native MATLAB arithmetic. For half precision we use the `chop` function[2] of [12]. We use three types of matrices.

- Type 1. $A \in \mathbb{R}^{n\times n}$ is generated using the default mode in the MATLAB function `gallery('randsvd')`, which gives geometrically distributed singular values. We set $n = 500$ and $\kappa_2(A) = 10^{10}$.
- Type 2. Polynomial decay. These are the matrices (5.2.30) with $n = 500, r = 100$, $p = 2$, and $\phi = 1$.

2 https://github.com/higham/chop

Table 5.3.1: Iteration counts for experiments with Type 1 matrices with various relative tolerances ε and choices of θ.

	$\varepsilon = 10^{-1}$			$\varepsilon = 10^{-3}$			$\varepsilon = 10^{-5}$			$\varepsilon = 10^{-7}$		
θ	0.1	1	10	0.1	1	10	0.1	1	10	0.1	1	10
t_h	6	5	0	10	5	0	0	0	0	0	0	0
t_s	0	1	6	6	11	16	26	25	20	30	25	20
t_d	0	0	0	0	0	0	0	1	6	6	11	16
Cost	0.27	0.31	0.51	0.38	0.46	0.53	0.55	0.56	0.65	0.64	0.70	0.76

- Type 3. Exponential decay[22, Sec. 7.3.1]. We generate random orthogonal matrices $U, V \in \mathbb{R}^{n\times n}$ from the Haar distribution, as in (5.2.30), and define

$$A = UDV^T, \quad D = \operatorname{diag}(I_r, 10^{-p}, 10^{-2p}, \ldots, 10^{-(n-r)p}). \tag{5.3.7}$$

  We take $n = 500$, $r = 100$ and $p = 0.1$.

In Tables 5.3.1, 5.3.2, and 5.3.3, (t_h, t_s, t_d) denote the number of iterations in half, single, and double precision respectively. We use (5.3.6) for the choice of the ε_j, with the values for the global tolerance ε indicated in each table. For each matrix type we choose three θ values: 0.1, 1, and 10. In all experiments, the mixed precision and fp64 algorithms satisfy the global tolerance ε with a comparable final error. For the same values of θ, the mixed-precision and fp64 algorithms always require the same number of iterations.

We have also included the reduction in computational cost for each set of results, given as a number between 0 and 1, where we are taking the cost of an equivalent number of fp64 iterations to be 1. To work out this cost we assume a ratio of $1 : 2 : 4$ for the costs of fp16, fp32, and fp64 arithmetics. We take the operation count of computing $C = AB$ with $A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{n\times r}$ to be $2mnr$ flops, and the cost of computing a QR factorization to be $2n^2(m - n/3)$ flops for $A \in \mathbb{R}^{m\times n}$[5, Chap. 1], [9, App. C]. For iteration i of Algorithm 5.6 this gives $10mnb + 6b^2(m - b/3)$ flops at precision u_j, and $4(i-1)mbr + 2b^2(m - b/3)$ flops at precision u_1. The "Cost" values

Table 5.3.2: Iteration counts for experiments with Type 2 matrices with various relative tolerances ε and choices of θ.

	$\varepsilon = 10^{-1}$			$\varepsilon = 10^{-2}$			$\varepsilon = 10^{-3}$			$\varepsilon = 10^{-4}$		
θ	0.1	1	10	0.1	1	10	0.1	1	10	0.1	1	10
t_h	10	8	0	9	1	0	2	1	0	0	0	0
t_s	0	2	10	2	10	11	10	11	12	18	18	13
t_d	0	0	0	0	0	0	0	0	0	0	0	5
Cost	0.28	0.33	0.52	0.32	0.50	0.52	0.48	0.50	0.52	0.53	0.53	0.66

Table 5.3.3: Iteration counts for experiments with Type 3 matrices with various relative tolerances ε and choices of θ.

	$\varepsilon = 10^{-1}$			$\varepsilon = 10^{-3}$			$\varepsilon = 10^{-5}$			$\varepsilon = 10^{-7}$		
θ	0.1	1	10	0.1	1	10	0.1	1	10	0.1	1	10
t_h	11	9	0	2	1	0	0	0	0	0	0	0
t_s	0	2	11	11	12	13	15	10	4	6	5	4
t_d	0	0	0	0	0	0	0	5	11	11	12	13
Cost	0.28	0.32	0.52	0.49	0.51	0.52	0.53	0.69	0.87	0.83	0.86	0.89

in the table are then worked out using the specific iteration counts and assumed cost ratios .

We see that the mixed precision iterations can lead to computational gains. Smaller values of θ leads to better performance of Algorithm 5.6, as it performs a greater proportion of the operations in low precision while still satisfying the final tolerance. We have found $\theta = 0.1$ to be an appropriate choice in our experiments.

Under the assumed cost ratios for fp16, fp32, and fp64, for certain choices of θ and ε we can expect the mixed-precision algorithm to be at least twice as fast as the fp64 algorithm.

5.4 CONCLUDING REMARKS

We have addressed the question of how rounding errors affect the exact arithmetic error bounds for randomized low rank matrix approximating. Our key findings for the fixed rank problem are summarized by the leading order rounding error terms in Table 5.2.1. For the power iteration (Algorithms 5.2 and 5.4), the probabilistic error bound is proportional to $m^{1/2}ku$, so under the assumptions of probabilistic error analysis the effects of rounding errors will be negligible if $m^{1/2}ku$ is sufficiently smaller than the error $\|(I - QQ^T)A\|_F$ for exact arithmetic. For IEEE half precision arithmetic (fp16), for which $u \approx 4.88 \times 10^{-4}$, the rounding error bound could well dominate unless m is small.

We proposed in Algorithm 5.6 an algorithm that exploits arithmetics of different precisions. It gradually decreases the precision of the arithmetic as the algorithm proceeds, exploiting the fact that as the approximation error decreases we need less precision in the arithmetic. Our experiments showed potential benefits, since the low precision iterations will have lower arithmetic, energy, and memory costs than higher precision ones.

ACKNOWLEDGMENTS

We thank Theo Mary for helpful discussions on this work, and Daniel Kressner for suggesting the use of (5.2.13) and the analysis leading to (5.2.15). All data and codes supporting this work are available at https://github.com/michaelc100/Mixed-Precision-RandNLA.

APPENDIX

5.A PROBABILISTIC ERROR ANALYSIS

To extend the worst-case analysis to the probabilistic case, we simply need to replace any use of the worst-case versions of the rounding error results with their stated probabilistic versions. This analysis is reproduced below. There are no meaningful differences beyond the worst-case and probabilistic distinction and the reduction in error constant that follows. For this appendix, we operate as if all assumptions needed for probabilistic error analysis hold. There is still future work to be done refining the assumptions mentioned in Chapter 4. To avoid repetition these assumptions are not stated in any of the theorems that follow. We carry out a first-order analysis, so $O(u^2)$ terms are omitted.

We can first obtain an error bound for the QR factorization performed on the matrix $Y = A\Omega$. This is an analogue of Lemma 5.1.

Lemma 5.4. *Let $Y = A\Omega$, where $A \in \mathbb{R}^{m\times n}$ with $m \geqslant n$ and $\Omega \in \mathbb{R}^{n\times k}$, where $k \leqslant n$. Householder QR factorization of the computed $\widehat{Y}$ produces a computed upper triangular $\widehat{R}$ satisfying*

$$Y + \Delta Y = \widetilde{Q}\widehat{R}, \quad \|\Delta y_j\|_2 \leqslant (\overline{\gamma}_n(\lambda) + c_1\lambda\sqrt{k}\overline{\gamma}_m(\lambda))\|A\|_F\|\omega_j\|_2, \quad j = 1:k, \tag{5.A.1}$$

where $\widetilde{Q} \in \mathbb{R}^{m\times k}$ has orthonormal columns and ω_j is the jth column of Ω.

Proof. From (5.2.25) we have $\widehat{Y} = Y + \Delta Y^{(1)}$, where

$$\|\Delta y_j^{(1)}\|_2 \leqslant \overline{\gamma}_n(\lambda)\| \, |A||\omega_j| \, \|_2 \leqslant \overline{\gamma}_n(\lambda)\|A\|_F\|\omega_j\|_2. \tag{5.A.2}$$

By (5.2.27), Householder QR factorization of $\widehat{Y}$ yields $\widehat{Y} + \Delta Y^{(2)} = \widetilde{Q}\widehat{R}$, where $\widetilde{Q}$ has orthonormal columns and

$$\|\Delta y_j^{(2)}\|_2 \leqslant c_1\lambda\sqrt{k}\overline{\gamma}_m(\lambda)\|\widehat{y}_j\|_2 \leqslant c_1\lambda\sqrt{k}\overline{\gamma}_m(\lambda)\big(\|y_j\|_2 + \overline{\gamma}_n(\lambda)\|A\|_F\|\omega_j\|_2\big).$$

Writing $\Delta Y = \Delta Y^{(1)} + \Delta Y^{(2)}$ and using $\|y_j\|_2 \leqslant \|A\|_F\|w_j\|_2$ gives (5.A.1). □

From [3, Lem. 4.3] the matrix $\widehat{B}$ computed on line 1 of Algorithm 5.1 using the factored form of Q satisfies

$$\widehat{B} = \widetilde{Q}^T A + \Delta A, \quad \|\Delta A\|_F \leqslant c_2\lambda\sqrt{k}\overline{\gamma}_m(\lambda)\|A\|_F.$$

The matrix $\widehat{U}$ from line 3 of Algorithm 5.2 computed using the factored form of Q satisfies

$$\widehat{U} = \widetilde{Q}\widetilde{U} + \Delta U, \quad \|\Delta U\|_F \leqslant c_3\lambda\sqrt{k}\overline{\gamma}_m(\lambda)\|\widetilde{U}\|_F = c_3\lambda k\overline{\gamma}_m(\lambda).$$

Hence

$$\begin{aligned} A - \widehat{U}\Sigma V^T &= A - \widetilde{Q}\widetilde{U}\Sigma V^T - \Delta U\Sigma V^T \\ &= A - \widetilde{Q}\widehat{B} - \Delta U\Sigma V^T \\ &= (I - \widetilde{Q}\widetilde{Q}^T)A - \widetilde{Q}\Delta A - \Delta U\Sigma V^T \\ &= (I - \widetilde{Q}\widetilde{Q}^T)A + E, \end{aligned} \tag{5.A.3}$$

where

$$\begin{aligned}\|E\|_F &\leqslant c_2\lambda\sqrt{k}\overline{\gamma}_m(\lambda)\|A\|_F + c_3\lambda k\overline{\gamma}_m(\lambda)\|\Sigma V^T\|_F \\ &= c_2\lambda\sqrt{k}\overline{\gamma}_m(\lambda)\|A\|_F + c_3\lambda k\overline{\gamma}_m(\lambda)\|\widehat{B}\|_F \\ &\leqslant c_2\lambda\sqrt{k}\overline{\gamma}_m(\lambda)\|A\|_F + (c_3\lambda k\overline{\gamma}_m(\lambda) + O(u^2))\|A\|_F \\ &= (c_4\lambda\sqrt{k}\overline{\gamma}_m(\lambda)(1+\sqrt{k}) + O(u^2))\|A\|_F. \end{aligned} \tag{5.A.4}$$

For the case of the power iteration, the extension is simple. Introducing the exact Q, we rewrite this equation as

$$A - \widehat{U}\Sigma V^T = (I - QQ^T)A + (QQ^T - \widetilde{Q}\widetilde{Q}^T)A + E. \tag{5.A.5}$$

We have

$$\|A - \widehat{U}\widehat{\Sigma}\widehat{V}^T\|_F \leqslant \|(I - QQ^T)A\|_F + \|(QQ^T - \widetilde{Q}\widetilde{Q}^T)A\|_F + \|E\|_F, \tag{5.A.6}$$

where we have already bounded $\|E\|_F$ probabilistically in (5.A.4) and $\|(I - QQ^T)A\|_F$ is bounded by the analysis in [6, sect. 9.3].

We note the probabilistic analogue of (5.2.20).

$$\begin{aligned}&Y_{2q+1} + \Delta Y_{2q+1} = \widetilde{Q}\widehat{R}_{2q+1}, \\ &\|\Delta Y_{2q+1}\|_F \leqslant k^{1/2}(\widetilde{\gamma}_n(\lambda) + c_5\lambda\sqrt{k}\overline{\gamma}_m(\lambda) + O(u^2))\|A\|_F \end{aligned} \tag{5.A.7}$$

Following the analysis as in Section 5.2.1, we obtain

$$\begin{aligned}FY_{2q+1} &= QQ^TY_{2q+1} - \widetilde{Q}\widetilde{Q}^TY_{2q+1} = (I - \widetilde{Q}\widetilde{Q}^T)Y_{2q+1} \\ &= -(I - \widetilde{Q}\widetilde{Q}^T)\Delta Y_{2q+1}. \end{aligned} \tag{5.A.8}$$

Combining (5.A.7) and (5.A.8) gives

$$\begin{aligned}\|(QQ^T - \widetilde{Q}\widetilde{Q}^T)A\|_F &= \|FA\|_F = \|FAQ_{2q}\|_F = \|FY_{2q+1}\|_F \\ &\leqslant \|\Delta Y_{2q+1}\|_F \leqslant k^{1/2}(\overline{\gamma}_n(\lambda) + c_5\lambda\sqrt{k}\overline{\gamma}_m(\lambda) + O(u^2))\|A\|_F,\end{aligned}$$

combined with (5.A.4) gives Theorem 5.3

To see the reduction in leading rounding error constant for the basic method, note that the leading rounding error contribution in Lemma 5.4 is reduced to $\sqrt{mk}u$, so the only change is a reduction in the contribution of $\|\Delta Y\|_F$ in (5.2.15), which carries through to reducing the overall leading order contribution to $O((mn)^{1/2}k^{3/2}u)$ in (5.2.16).

REFERENCES

[1] H. Avron, P. Maymounkov, and S. Toledo. "Blendenpik: Supercharging LAPACK's least-squares solver." *SIAM J. Sci. Comput.* 32.3 (2010), pp. 1217–1236 (cited on p. 117).

[2] E. Carson and I. Daužickaitė. "Single-pass Nyström approximation in mixed precision." *arXiv e-prints* (2022). arXiv:2205.13355 (cited on p. 118).

[3] M. P. Connolly and N. J. Higham. *Probabilistic Rounding Error Analysis of Householder QR Factorization*. MIMS EPrint 2022.5. UK: Manchester Institute for Mathematical Sciences, The University of Manchester, Feb. 2022, p. 16 (cited on pp. 128, 129, 144).

[4] M. P. Connolly, N. J. Higham, and T. Mary. "Stochastic rounding and its probabilistic backward error analysis." *SIAM J. Sci. Comput.* 43.1 (Jan. 2021), A566–A585 (cited on p. 128).

[5] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Fourth. Baltimore, MD, USA: Johns Hopkins University Press, 2013, pp. xxi+756 (cited on p. 140).

[6] N. Halko, P. G. Martinsson, and J. A. Tropp. "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions." *SIAM Rev.* 53.2 (2011), pp. 217–288 (cited on pp. 117, 123–126, 128, 130, 131, 145).

[7] N. Halko, P.-G. Martinsson, Y. Shkolnisky, and M. Tygert. "An algorithm for the principal component analysis of large data sets." *SIAM J. Sci. Comput.* 33.5 (2011), pp. 2580–2594 (cited on p. 117).

[8] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. Second. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2002, pp. xxx+680 (cited on pp. 119–121).

[9] N. J. Higham. *Functions of Matrices: Theory and Computation*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2008, pp. xx+425 (cited on p. 140).

[10] N. J. Higham and T. Mary. "A new approach to probabilistic rounding error analysis." *SIAM J. Sci. Comput.* 41.5 (2019), A2815–A2835 (cited on p. 128).

[11] N. J. Higham and T. Mary. "Sharper probabilistic backward error analysis for basic linear algebra kernels with random data." *SIAM J. Sci. Comput.* 42.5 (2020), A3427–A3446 (cited on p. 128).

[12] N. J. Higham and S. Pranesh. "Simulating low precision floating-point arithmetic." *SIAM J. Sci. Comput.* 41.5 (2019), pp. C585–C602 (cited on p. 139).

[13] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge, UK: Cambridge University Press, 1991, pp. viii+607 (cited on p. 120).

[14] I. C. F. Ipsen and H. Zhou. "Probabilistic error analysis for inner products." *SIAM J. Matrix Anal. Appl.* 41.4 (Jan. 2020), pp. 1726–1741 (cited on p. 128).

[15] N. K. Kumar and J. Schneider. "Literature survey on low rank approximation of matrices." *Linear and Multilinear Algebra* 65.11 (2017), pp. 2212–2244 (cited on p. 117).

[16] E. Liberty, F. Woolfe, P.-G. Martinsson, V. Rokhlin, and M. Tygert. "Randomized algorithms for the low-rank approximation of matrices." *Proceedings of the National Academy of Sciences* 104.51 (2007), pp. 20167–20172 (cited on p. 117).

[17] P.-G. Martinsson and J. Tropp. "Randomized numerical linear algebra: foundations & algorithms." *Acta Numerica* 29 (2020), pp. 403–572 (cited on pp. 117, 118).

[18] P.-G. Martinsson and S. Voronin. "A randomized blocked algorithm for efficiently computing rank-revealing factorizations of matrices." *SIAM J. Sci. Comput.* 38.5 (2016), S485–S507 (cited on pp. 117, 133, 136).

[19] V. Rokhlin and M. Tygert. "A fast randomized algorithm for overdetermined linear least-squares regression." *Proc. Nat. Acad. Sci.* 105.36 (2008), pp. 13212–13217 (cited on p. 117).

[20] G. W. Stewart. "The efficient generation of random orthogonal matrices with an application to condition estimators." *SIAM J. Numer. Anal.* 17.3 (1980), pp. 403–409 (cited on p. 131).

[21] J. A. Tropp. "An introduction to matrix concentration inequalities." *Foundations and Trends in Machine Learning* 8.1-2 (2015), pp. 1–230 (cited on p. 123).

[22] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher. "Practical sketching algorithms for low-rank matrix approximation." *SIAM Journal on Matrix Analysis and Applications* 38.4 (2017), pp. 1454–1485 (cited on p. 140).

[23] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher. "Streaming low-rank matrix approximation with an application to scientific simulation." *SIAM J. Sci. Comput.* 41.4 (2019), A2430–A2463 (cited on p. 131).

[24] F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert. "A fast randomized algorithm for the approximation of matrices." *Applied and Computational Harmonic Analysis* 25.3 (2008), pp. 335–366 (cited on p. 117).

[25] W. Yu, Y. Gu, and Y. Li. "Efficient randomized algorithms for the fixed-precision low-rank matrix approximation." *SIAM J. Matrix Anal. Appl.* 39.3 (2018), pp. 1339–1359 (cited on p. 134).

6 CONCLUSION

It has been well known for some time that the dimensional constants in rounding error bounds can, in all likelihood, be replaced by something like their square root. However, the two trends of ever increasing problems sizes and seemingly ever decreasing precisions meant that it became vital to replace this rule of thumb by something more rigorous.

In Chapter 3, we provided a detailed error analysis of stochastic rounding. Crucially, we showed that the rounding errors produced by stochastic rounding are mean independent. By relaxing the assumption of independence of rounding errors from [2] to *mean independence*, we developed error bounds for inner product based algorithms where worst case bounds can be replaced by their square root. This showed that for stochastic rounding, the rule of thumb is a rule. This helps to highlight the benefits of using stochastic rounding in actual computation, and the references in Chapter 3 to its use and implementation in hardware show the scientific community are interested in it. More central to this thesis however, is its theoretical use in providing informative error bounds.

In Chapter 4, we used this same model of rounding errors to analyse Householder QR factorization. We again observed the same square rooting of dimensional constants. The analysis for this case was more involved than Chapter 3, in particular requiring the use of a *matrix* concentration inequality [3] where previously we had only required scalar results. Extending our results to other orthogonal transformation based computations, and having covered inner-product based computations in

Chapter 3, showed that the square rooting effect is widespread throughout numerical linear algebra.

In Chapter 5, we considered the problem of low rank matrix approximation. We first considered the randomized SVD procedure made famous by [1]. Typical error bounds for this procedure assume that any errors incurred by the floating-point arithmetic will be swamped by those of the randomization process. We were able to derive both worst-case and probabilistic bounds for this algorithm. Using our probabilistic bounds as guidance, we then devised a mixed-precision randomized SVD, demonstrating potential performance gains in speed and memory without any loss in accuracy.

The field of mixed-precision computation, and particularly mixed-precision numerical linear algebra, is truly thriving. There are endless possibilities for tuning and altering existing algorithms, as well as devising brand new ones that make the best use of the floating-point formats being offered by the state of the art hardware. Possibilities for future work include exploring more applications for stochastic rounding and identifying scenarios where its favourable rounding properties are particularly useful, as well as investigating the numerical consequences of different possible implementations of stochastic rounding. Extending our ideas about mixed-precision in the randomized SVD to other randomized algorithms is also an area of great interest, particularly as the inherent approximation error in these algorithms gives us a lot of leeway with the precision used. By devising rigorous yet informative probabilistic error bounds, we hope these will have a positive impact on numerical analysts when devising ever faster algorithms that are still, in all likelihood, accurate.

REFERENCES

[1] N. Halko, P. G. Martinsson, and J. A. Tropp. "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions." *SIAM Rev.* 53.2 (2011), pp. 217–288 (cited on p. 150).

[2] N. J. Higham and T. Mary. "A new approach to probabilistic rounding error analysis." *SIAM J. Sci. Comput.* 41.5 (2019), A2815–A2835 (cited on p. 149).

[3] J. A. Tropp. "User-friendly tail bounds for sums of random matrices." *Found. Comput. Math.* 12.4 (Aug. 2012), pp. 389–434 (cited on p. 149).