Randomized Low Rank Matrix Approximation: Rounding Error Analysis and a Mixed Precision Algorithm

Connolly, Michael P. and Higham,
Nicholas J. and Pranesh, Srikara

2022

MIMS EPrint: **2022.10**

Manchester Institute for Mathematical Sciences
School of Mathematics

The University of Manchester

Reports available from: http://eprints.maths.manchester.ac.uk/
And by contacting: The MIMS Secretary
School of Mathematics
The University of Manchester
Manchester, M13 9PL, UK

ISSN 1749-9097

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

MICHAEL P. CONNOLLY†, NICHOLAS J. HIGHAM†, AND SRIKARA PRANESH†

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 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.

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

AMS subject classifications. 65G50, 65F05

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 [23], 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 $m \geq n$ 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 $A \approx QQ^TA$ 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 1.1 and 1.2. Here we focus specifically on computing the singular value decomposition (SVD), but other factorizations can also be computed using Q from Algorithm 1.1 (see, for example, [18, sect. 3]). Algorithm 1.1 describes the general case of drawing a matrix Ω with isotropic columns. A random vector $x \in \mathbb{R}^n$ is isotropic if $\mathbb{E}(xx^T) = I_n$, where $\mathbb{E}$ denotes expectation. A thin QR factorization means one with a rectangular Q and an upper triangular R.

*Version of July 19, 2022.

Funding: This work was supported by by MathWorks, Engineering and Physical Sciences Research Council grant EP/P020720/1, the Royal Society, and the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

†Department of Mathematics, University of Manchester, Manchester, M13 9PL, UK (michael.connolly-3@manchester.ac.uk, nick.higham@manchester.ac.uk, psrikara@gmail.com).

Algorithm 1.1 Given $A \in \mathbb{R}^{m\times n}$ with $m \geq n$ and an integer $k < n$, compute a matrix $Q \in \mathbb{R}^{m\times k}$ with orthonormal columns such that $A \approx QQ^TA$.

1: Draw a matrix $\Omega \in \mathbb{R}^{n\times k}$ with isotropic columns.
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 1.2 Given $A \in \mathbb{R}^{m\times n}$ with $m \geq n$ and Q from Algorithm 1.1, 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: $B = Q^TA$
2: Compute the economy size SVD $B = \widetilde{U}\Sigma V^T$.
3: $U = Q\widetilde{U}$

Note that line 2 of Algorithm 1.1 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 1.2. In exact arithmetic,

$$\|A - U\Sigma V^T\| = \|A - Q\widetilde{U}\Sigma V^T\| = \|A - QB\| = \|A - QQ^TA\|, \tag{1.1}$$

so it is only Algorithm 1.1, the rangefinder process, that introduces errors. Previous error analyses of randomized algorithms that provide bounds on the right-hand side of (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 (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 1.1 and 1.2. 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 (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.

Our analysis focuses on random matrices whose columns are isotropic, so it includes as particular cases Gaussian matrices and other commonly used structured random matrices. By a Gaussian matrix we mean a random matrix whose elements are independently drawn from the standard normal distribution (mean 0, variance 1).

In section 2 we give a rounding error analysis for Algorithms 1.1 and 1.2, which treat the fixed-rank problem. We then extend the analysis to a power iteration generalization of Algorithm 1.1. This analysis is a usual worst-case analysis, but in section 2.2 we give a probabilistic rounding error analysis. In section 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 4.

2. Rounding error analysis. If there are no rounding errors then the error in the computed SVD from Algorithm 1.2 is given by (1.1). We wish to derive an error bound that accounts for rounding errors. The only simplifying assumption we will make is that the SVD in line 2 of Algorithm 1.2 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]:

$$\mathrm{fl}(x \text{ op } y) = (x \text{ op } y)(1+\delta), \quad |\delta| \le u, \tag{2.1}$$

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

$$\gamma_n = \frac{nu}{1-nu}, \qquad \widetilde{\gamma}_n = \frac{cnu}{1-cnu}, \tag{2.2}$$

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],

$$\|ABC\|_F \le \|A\|_2 \|B\|_F \|C\|_2. \tag{2.3}$$

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].

$$\widehat{C} = C + \Delta C, \quad |\Delta C| \le \gamma_n |A||B|. \tag{2.4}$$

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]

$$A + \Delta A = \widetilde{Q}\widehat{R}, \quad \|\Delta a_j\|_2 \le \widetilde{\gamma}_{mk}\|a_j\|_2, \quad j = 1\colon k. \tag{2.5}$$

We can now obtain an error bound for step 2 of Algorithm 1.1

LEMMA 2.1. *Let $Y = A\Omega$, where $A \in \mathbb{R}^{m\times n}$ with $m \ge n$ and $\Omega \in \mathbb{R}^{n\times k}$, where $k \le n$. Householder QR factorization of the computed $\widehat{Y}$ produces a computed upper triangular $\widehat{R}$ satisfying*

$$Y + \Delta Y = \widetilde{Q}\widehat{R}, \quad \|\Delta Y^{(j)}\|_2 \le (\gamma_n + \widetilde{\gamma}_{mk} + \gamma_n\widetilde{\gamma}_{mk})\|A\|_F\|\omega_j\|_2, \quad j = 1\colon k, \tag{2.6}$$

where $\widetilde{Q} \in \mathbb{R}^{m\times k}$ has orthonormal columns and ω_j is the jth column of Ω.

Proof. From (2.4) we have $\widehat{Y} = Y + \Delta Y_1$, where

$$\|\Delta Y_1^{(j)}\|_2 \le \gamma_n \| \, |A||\omega_j| \, \|_2 \le \gamma_n \|A\|_F \|\omega_j\|_2. \tag{2.7}$$

By (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

$$\|\Delta Y_2^{(j)}\|_2 \le \widetilde{\gamma}_{mk}\|\widehat{y}_j\|_2 \le \widetilde{\gamma}_{mk}\big(\|y_j\|_2 + \gamma_n\|A\|_F\|\omega_j\|_2\big).$$

Writing $\Delta Y = \Delta Y_1 + \Delta Y_2$ and using $\|y_j\|_2 \le \|A\|_F\|w_j\|_2$ gives (2.6). ∎

We could explicitly form the matrix Q after computing the QR factorization in Algorithm 1.1 and then explicitly form the matrix product $B = Q^T A$ on step 1 of Algorithm 1.2. 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 2.1 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

$$\widehat{G} = \widetilde{Q}^T H + \Delta H, \quad \|\Delta h_j\|_2 \le \widetilde{\gamma}_{mk}\|h_j\|_2, \quad j = 1:n. \tag{2.8}$$

Therefore the matrix $\widehat{B}$ computed on line 1 of Algorithm 1.2 using the factored form of Q satisfies

$$\widehat{B} = \widetilde{Q}^T A + \Delta A, \quad \|\Delta A\|_F \le \widetilde{\gamma}_{mk}\|A\|_F. \tag{2.9}$$

The matrix $\widehat{U}$ from line 3 of Algorithm 1.2 computed using the factored form of Q satisfies

$$\widehat{U} = \widetilde{Q}\widetilde{U} + \Delta U, \quad \|\Delta U\|_F \le \widetilde{\gamma}_{mk}\|\widetilde{U}\|_F = k^{1/2}\widetilde{\gamma}_{mk}. \tag{2.10}$$

Hence

$$\begin{aligned} A - \widehat{U}\Sigma V^T &= A - \widetilde{Q}\widetilde{U}\Sigma V^T - \Delta U \Sigma V^T \\ &= A - \widetilde{Q}\widehat{B} - \Delta U \Sigma V^T \\ &= (I - \widetilde{Q}\widetilde{Q}^T)A - \widetilde{Q}\Delta A - \Delta U \Sigma V^T \\ &= (I - \widetilde{Q}\widetilde{Q}^T)A + E, \end{aligned} \tag{2.11}$$

where

$$\begin{aligned} \|E\|_F &\le \widetilde{\gamma}_{mk}\|A\|_F + k^{1/2}\widetilde{\gamma}_{mk}\|\Sigma V^T\|_F \\ &= \widetilde{\gamma}_{mk}\|A\|_F + k^{1/2}\widetilde{\gamma}_{mk}\|\widehat{B}\|_F \\ &\le \widetilde{\gamma}_{mk}\|A\|_F + k^{1/2}\widetilde{\gamma}_{mk}(1+\widetilde{\gamma}_{mk})\|A\|_F \\ &= \widetilde{\gamma}_{mk}\bigl(1 + k^{1/2}(1+\widetilde{\gamma}_{mk})\bigr)\|A\|_F. \end{aligned} \tag{2.12}$$

Introducing the exact Q, we rewrite this equation as

$$A - \widehat{U}\Sigma V^T = (I - QQ^T)A + (QQ^T - \widetilde{Q}\widetilde{Q}^T)A + E. \tag{2.13}$$

In this expression, the first term is bounded by the existing exact arithmetic analysis and the last term is bounded by (2.12). We now focus on bounding the middle term, FA, where

$$F = QQ^T - \widetilde{Q}\widetilde{Q}^T.$$

Our aim is to bound $\|FA\|_F$, but first we will bound $\|FY\|_F$. Note that $QQ^TY = QQ^TQR = QR = Y$. Hence

$$FY = QQ^TY - \widetilde{Q}\widetilde{Q}^TY = (I - \widetilde{Q}\widetilde{Q}^T)Y = -(I - \widetilde{Q}\widetilde{Q}^T)\Delta Y,$$

by (2.6). Using (2.3), together with $\|I - \widetilde{Q}\widetilde{Q}^T\|_2 \le 1$ we then have

$$\|FY\|_F \le \|\Delta Y\|_F. \tag{2.14}$$

We now state an important result about matrices with isotropic columns. This is contained in [17, sects. 4.2, 4.8] but is not stated and proved as a self-contained result. We include a proof, for completeness.

LEMMA 2.2. *Let $\Omega \in \mathbb{R}^{n\times k}$ be a random matrix with isotropic columns. Then for $B \in \mathbb{R}^{m\times n}$ we have*

$$\mathbb{E}\big(\|B\Omega\|_F^2\big) = k\,\|B\|_F^2. \tag{2.15}$$

Proof. For any $A \in \mathbb{R}^{n\times n}$, define $x_i = \omega_i^T A\omega_i$, where ω_i is the ith column of Ω. By cyclicity of the trace and linearity of the expected value we have

$$\mathbb{E}(x_i) = \mathbb{E}\big(\mathrm{trace}(\omega_i\omega_i^T A)\big) = \mathrm{trace}\big(\mathbb{E}(\omega_i\omega_i^T A)\big) = \mathrm{trace}\big(\mathbb{E}(\omega_i\omega_i^T)A\big) = \mathrm{trace}(A).$$

Setting $A = B^TB$, we have $\|B\Omega\|_F^2 = \sum_{i=1}^k \omega_i^T(B^TB)\omega_i = \sum_{i=1}^k x_i$, hence

$$\mathbb{E}(\|B\Omega\|_F^2) = \mathbb{E}\left(\sum_{i=1}^k x_i\right) = k\,\mathrm{trace}(B^TB) = k\|B\|_F^2.$$

∎

Using Lemma 2.2 and (2.14), and since $Y = A\Omega$, we have

$$\|FA\|_F^2 = \frac{1}{k}\mathbb{E}\big(\|FA\Omega\|_F^2\big) = \frac{1}{k}\mathbb{E}\big(\|FY\|_F^2\big) \le \frac{1}{k}\mathbb{E}\big(\|\Delta Y\|_F^2\big),$$

by the monotonicity of the expected value. Hence

$$\|FA\|_F \le \frac{1}{\sqrt{k}}\mathbb{E}\big(\|\Delta Y\|_F^2\big)^{1/2}. \tag{2.16}$$

We finally need to bound $\mathbb{E}\big(\|\Delta Y\|_F^2\big)^{1/2}$. From (2.6) we obtain

$$\|\Delta Y\|_F^2 \le (\gamma_n + \widetilde{\gamma}_{mk} + \gamma_n\widetilde{\gamma}_{mk})^2\|A\|_F^2\|\Omega\|_F^2. \tag{2.17}$$

Using these bounds in (2.13) gives the following result.

THEOREM 2.3. *Let $A \in \mathbb{R}^{m\times n}$, where $m \ge n$, and assume that the SVD on line 2 of Algorithm 1.2 is computed exactly. In floating-point arithmetic, Algorithm 1.2 produces a computed SVD $A \approx \widehat{U}\widehat{\Sigma}\widehat{V}^T$ satisfying*

$$\begin{aligned}\|A - \widehat{U}\widehat{\Sigma}\widehat{V}^T\|_F \le \|(I - QQ^T)A\|_F + \big[&\widetilde{\gamma}_{mk}\big(1 + k^{1/2}(1 + \widetilde{\gamma}_{mk})\big)\\ &+ k^{-1/2}(\gamma_n + \widetilde{\gamma}_{mk} + \gamma_n\widetilde{\gamma}_{mk})\mathbb{E}\left(\|\Omega\|_F^2\right)^{1/2}\big]\|A\|_F.\end{aligned} \tag{2.18}$$

The first term in (2.18), $\|(I-QQ^T)A\|_F$, is bounded by [6, Thm. 9.1]. The second and third terms bound the rounding errors; the third is deterministic once a choice for Ω has been fixed.

Consider now the particular case of a Gaussian Ω. It is easy to see that Ω has isotropic columns. Using $\mathbb{E}(\|\Omega\|_F^2) = nk$ (which follows by direct calculation or by Lemma 2.2) in (2.18) gives the following result.

COROLLARY 2.4. *Let $A \in \mathbb{R}^{m\times n}$, where $m \geq n$, and assume that a Gaussian Ω is used in Algorithm 1.1 and that the SVD on line 2 of Algorithm 1.2 is computed exactly. In floating-point arithmetic, Algorithm 1.2 produces a computed SVD $A \approx \widehat{U}\widehat{\Sigma}\widehat{V}^T$ satisfying*

$$\begin{aligned}\|A - \widehat{U}\widehat{\Sigma}\widehat{V}^T\|_F &\leq \|(I - QQ^T)A\|_F \\ &\quad + \big((1 + k^{1/2} + n^{1/2})\widetilde{\gamma}_{mk} + n^{1/2}\gamma_n(1 + \widetilde{\gamma}_{mk}) + k^{1/2}\widetilde{\gamma}_{mk}^2\big)\|A\|_F. \end{aligned} \tag{2.19}$$

Now we have chosen Ω as Gaussian, a more descripitve bound for $\|(I-QQ^T)A\|_F$ is available [6, Thm. 10.7]. Corollary 2.4 shows that the rounding errors contribute an additional error with leading order term $mn^{1/2}ku\|A\|_F$ to the inherent approximation errors. If we are working in double precision arithmetic this extra error term should be negligible, but if single precision or half precision is being used it could be significant.

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]. Since the complex analogue to (2.1) is of the same form and Lemma 2.2 holds for complex data, our results for real arithmetic are valid for complex modulo an appropriate increase in the constants [8, sect. 3.6]. An SRFT matrix has the form

$$\Omega = \sqrt{\frac{n}{k}}DFR, \tag{2.20}$$

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.

Any given column ω of Ω is then just a randomly selected column of $\sqrt{n/k}DF$, and we will show that the columns of $\sqrt{n/k}DF$ are isotropic up to a constant scale factor.

We first show that $\mathbb{E}(z_i) = 0$ where z_i is a diagonal element of D. We have $z_i = \mathrm{e}^{\mathrm{i}\theta}$ with θ uniformly distributed on the interval $[0, 2\pi)$. Then $\mathbb{E}(z_i) = \int_0^{2\pi} \mathrm{e}^{\mathrm{i}\theta} f(\theta)\,\mathrm{d}\theta$, where $f(\theta)$ is the probability density function of z_i. As z_i is distributed uniformly on the unit circle, $f(\theta)$ must be constant, so $\mathbb{E}(z_i) = \int_0^{2\pi} \mathrm{e}^{\mathrm{i}\theta} f(\theta)\,\mathrm{d}\theta = 0$.

For $i \neq j$, we have $\mathbb{E}(z_i z_j^*) = \mathbb{E}(z_i)\mathbb{E}(z_j^*) = 0$ and so the off-diagonal entries of $\mathbb{E}(\omega\omega^*)$ are zero. We have $f_{jk}f_{jk}^* = 1/n$ and $\mathbb{E}(z_i z_i^*) = 1$, which means that the diagonal elements of $\mathbb{E}(\omega\omega^*)$ are $1/k$. Hence Ω is isotropic up to a constant scaling. This scaling has no effect on Algorithms 1.1 and 1.2, since it is absorbed in R. We conclude that Corollary 2.4 remains valid when Ω is the SRFT matrix, with the added caveat that the bound for $\|(I - QQ^T)A\|_F$, the exact error contribution, now comes from [6, Thm 11.2].

2.1. Extension to power iteration. If the singular values of A decay slowly then the accuracy of Algorithm 1.1 in exact arithmetic degrades. To address this issue a power scheme that generalizes Algorithm 1.1 was proposed in [6, sect. 4.5], which we reproduce in Algorithm 2.1.

Algorithm 2.1 Given $A \in \mathbb{R}^{m\times n}$ with $m \geq n$, 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 2.1 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 2.2 is usually considered.

Algorithm 2.2 Given $A \in \mathbb{R}^{m\times n}$ with $m \geq n$, 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 2.2, $q = 1$ 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 2 for Algorithm 2.2, to bound $\|A - \widehat{U}\widehat{\Sigma}\widehat{V}^T\|_F$. From (2.13),

$$\|A - \widehat{U}\widehat{\Sigma}\widehat{V}^T\|_F \leq \|(I - QQ^T)A\|_F + \|(QQ^T - \widetilde{Q}\widetilde{Q}^T)A\|_F + \|E\|_F, \tag{2.21}$$

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 1.2 once we have computed Q via Algorithm 2.2, is bounded by (2.12). The matrix $\widetilde{Q}$ is the exact Q that arises, as defined in (2.5), from the final QR factorization of Algorithm 2.2.

The key observation is that we can identify the factorization $Q_{2q+1}R_{2q+1} = Y_{2q+1} = AQ_{2q}$ in Algorithm 2.2 with the factorization $QR = Y = A\Omega$ in Algorithm 1.1. By a similar proof to that of Lemma 2.1 we have

$$Y_{2q+1} + \Delta Y_{2q+1} = \widetilde{Q}\widehat{R}_{2q+1}, \quad \|\Delta Y_{2q+1}\|_F \leq k^{1/2}(\gamma_n + \widetilde{\gamma}_{mk} + \gamma_n\widetilde{\gamma}_{mk})\|A\|_F \tag{2.22}$$

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 2.2 are full QR factorizations with $Q_j \in \mathbb{R}^{m\times m}$ and $R_j = \begin{bmatrix} R'_j \\ 0 \end{bmatrix}$ 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

$$\begin{aligned} QQ^T Y_{2q+1} &= Q_{2q+1} I_{m,k} I_{m,k}^T Q_{2q+1}^T Q_{2q+1} R_{2q+1} \\ &= Q_{2q+1} I_{m,k} I_{m,k}^T \begin{bmatrix} R'_{2q+1} \\ 0 \end{bmatrix} \\ &= Q_{2q+1} \begin{bmatrix} R'_{2q+1} \\ 0 \end{bmatrix} \\ &= Q_{2q+1} R_{2q+1} = Y_{2q+1}. \end{aligned} \tag{2.23}$$

Then, with $F = QQ^T - \widetilde{Q}\widetilde{Q}^T$, using (2.22) and (2.23) we have

$$\begin{aligned} FY_{2q+1} &= QQ^T Y_{2q+1} - \widetilde{Q}\widetilde{Q}^T Y_{2q+1} = (I - \widetilde{Q}\widetilde{Q}^T) Y_{2q+1} \\ &= -(I - \widetilde{Q}\widetilde{Q}^T)\Delta Y_{2q+1}. \end{aligned} \tag{2.24}$$

Combining (2.22) and (2.24) gives

$$\begin{aligned} \|(QQ^T - \widetilde{Q}\widetilde{Q}^T)A\|_F &= \|FA\|_F = \|FAQ_{2q}\|_F = \|FY_{2q+1}\|_F \\ &\le \|\Delta Y_{2q+1}\|_F \le k^{1/2}(\gamma_n + \widetilde{\gamma}_{mk} + \gamma_n\widetilde{\gamma}_{mk})\|A\|_F. \end{aligned} \tag{2.25}$$

Using this bound in (2.13) gives the next result.

THEOREM 2.5. *Let $A \in \mathbb{R}^{m\times n}$, where $m \ge n$, and assume that the SVD on line 2 of Algorithm 1.2 is computed exactly. Algorithm 2.2, with $q \ge 1$, and Algorithm 1.2 produce a computed SVD $A \approx \widehat{U}\widehat{\Sigma}\widehat{V}^T$ satisfying*

$$\begin{aligned} \|A - \widehat{U}\widehat{\Sigma}\widehat{V}^T\|_F &\le \|(I - QQ^T)A\|_F \\ &\quad + \big((1 + k^{1/2})\widetilde{\gamma}_{mk} + k^{1/2}\gamma_n(1 + \widetilde{\gamma}_{mk}) + k^{1/2}\widetilde{\gamma}_{mk}^2\big)\|A\|_F. \end{aligned} \tag{2.26}$$

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 $\|B\|_F \le \sqrt{\operatorname{rank}(B)}\|B\|_2$.

Note the leading order rounding error contribution is $mk^{3/2}u$ compared with $mn^{1/2}ku$ in (2.19). The smaller constant in (2.26) compared with (2.19) is attributable to the fact that the final QR factorization in Algorithm 2.2 is of AQ_{2q+1} rather than of $A\Omega$ as in Algorithm 1.1, and we have exploited the orthogonality of Q_{2q+1}.

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 1.1 and 1.2.

For matrix multiplication the probabilistic analogue of (2.4) is given by [4, Thm. 5.9], [10, Thm. 3.4]

$$\widehat{C} = C + \Delta C, \quad |\Delta C| \le \overline{\gamma}_n(\lambda)|A||B|, \tag{2.27}$$

where

$$\overline{\gamma}_n(\lambda) := \exp\left(\frac{\lambda\sqrt{n}u + nu^2}{1-u}\right) - 1 = \lambda\sqrt{n}u + O(u^2). \tag{2.28}$$

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 (2.5) is given by [3, Thm. 4.4]

$$A + \Delta A = \widetilde{Q}\widehat{R}, \quad \|\Delta a_j\|_2 \le c\lambda\sqrt{k}\overline{\gamma}_m(\lambda)\|a_j\|_2 + O(u^2), \quad j = 1\colon n, \tag{2.29}$$

where c is a modest integer constant. The same reduction in the error constant applies to (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 2.3. 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 2.6. *Let $A \in \mathbb{R}^{m\times n}$, where $m \ge n$, and assume that the SVD on line 2 of Algorithm 1.2 is computed exactly. In floating-point arithmetic and under the assumptions of probabilistic error analysis, Algorithm 1.2 produces a computed SVD $A \approx \widehat{U}\widehat{\Sigma}\widehat{V}^T$ satisfying*

$$\begin{aligned}\|A - \widehat{U}\widehat{\Sigma}\widehat{V}^T\|_F \le{}& \|(I - QQ^T)A\|_F \\ &+ \big[c\lambda k^{1/2}\overline{\gamma}_m(\lambda)(1 + k^{1/2}) \\ &\quad + k^{-1/2}\big(\overline{\gamma}_n(\lambda) + c\lambda k^{1/2}\overline{\gamma}_m(\lambda)\big)\mathbb{E}(\|\Omega\|_F^2)^{1/2}\big]\|A\|_F + O(u^2).\end{aligned} \tag{2.30}$$

For Gaussian Ω, for which $\mathbb{E}(\|\Omega\|_F^2) = nk$ as noted above, the third term in (2.30) is proportional to $(mnk)^{1/2}u$, compared with $mn^{1/2}ku$ for the worst case bound (2.18).

We can apply the same procedure to Theorem 2.5 and obtain a probabilistic analogue.

THEOREM 2.7. *Let $A \in \mathbb{R}^{m\times n}$, where $m \ge n$, and assume that the SVD on line 2 of Algorithm 1.2 is computed exactly. In floating-point arithmetic and under the assumptions of probabilistic error analysis, Algorithm 2.2 and Algorithm 1.2 produce a computed SVD $A \approx \widehat{U}\widehat{\Sigma}\widehat{V}^T$ satisfying*

$$\begin{aligned}\|A - \widehat{U}\widehat{\Sigma}\widehat{V}^T A\|_F \le{}& \|A - QQ^T A\|_F \\ &+ \big((k + k^{1/2})\overline{\gamma}_m(\lambda) + k^{1/2}\overline{\gamma}_n(\lambda)\big)\|A\|_F + O(u^2).\end{aligned}$$

In Table 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.

2.3. Numerical experiments. We present some numerical experiments to test the sharpness of the worst-case error bound in Theorem 2.3 and the probabilistic error

TABLE 2.1
Leading order rounding error terms in the bounds.

	Worst case	Probabilistic
Algorithms 1.2 and 1.1 (standard, Gaussian Ω)	$mn^{1/2}ku$	$(mnk)^{1/2}u$
Algorithms 1.2 and 2.2 (power iteration)	$mk^{3/2}u$	$m^{1/2}ku$

bound in Theorem 2.6. We first need the result [6, Thm. 10.7] that for Gaussian Ω the bound

$$\|A - QQ^TA\|_F \le \psi_1(k,p,t)\left(\sum_{j>k}\sigma_j^2\right)^{1/2} + \psi_2(k,p,t,s)\sigma_{k+1}, \tag{2.31}$$

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

$$\psi_1(k,p,t) = 1 + t\left(\frac{3k}{p+1}\right)^{1/2}, \quad \psi_2(k,p,t,s) = st\frac{\mathrm{e}\sqrt{k+p}}{p+1}. \tag{2.32}$$

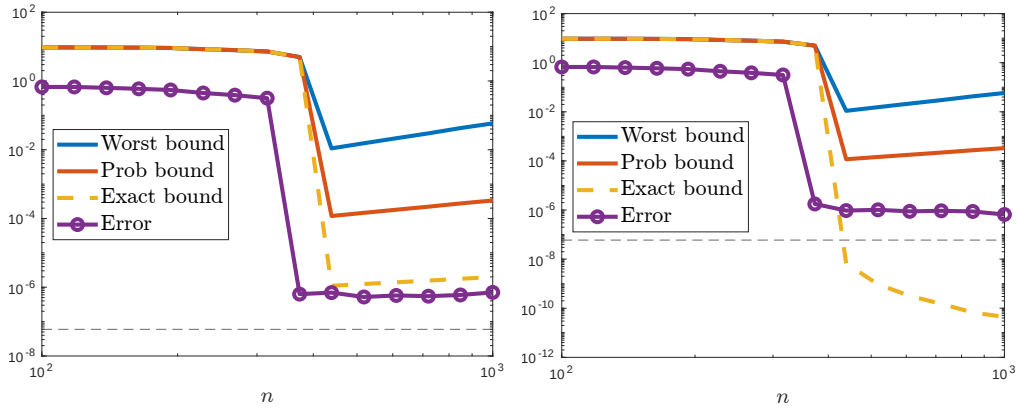
For even modest choices of these parameters the probability of failure is negligible. Choosing for example $p=t=s=5$ results in a probability of failure of 6×10^{-4}. We simply choose to set $p=t=s=1$, as we have observed that the probabilities associated with these parameters are pessimistic. For a similar reason, we set λ in Theorem 2.6 to 1. Additionally, all constants c contained in the $\widetilde{\gamma}$ terms in Theorem 2.3 are set to 1.

Note the parameter p above is an oversampling parameter, so in the bounds given in Theorem 2.3 we should strictly make the substitution $k \leftarrow k+p$. In Figure 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 [22]. In this section we just consider Algorithms 1.1 and 1.2. We do not consider the power iteration of section 2.1 as an error bound analogous to (2.31) 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
$$A = UDV^T, \qquad D = \mathrm{diag}(\phi I_r, 2^{-\alpha}, 3^{-\alpha}, \ldots, (n-r+1)^{-\alpha}). \tag{2.33}$$
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 precision. Test matrices are also rounded to single precision. The results are plotted in Figure 2.1, in which "Worst bound" denotes the worst-case rounding error bound (2.19) from Corollary 2.4, "Prob bound" denotes the probabilistic bound (2.30)



(a) Type 1 matrix with $k = \sqrt{n}$. (b) Type 2 matrix with $k = \sqrt{n}$.

FIGURE 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.*

from Theorem 2.6, "Exact bound" denotes the bound in (2.31), and "Error" is the approximation error $\|A - \widehat{U}\widehat{\Sigma}\widehat{V}^T\|_F / \|A\|_F$.

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 and is tighter than for the Type 1 matrices.

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 3.1 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.

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 [24, Alg. 2]. This is the algorithm implemented by the MATLAB `svdsketch` function[1]. We know from [24, Prop. 1] that when executed in exact arithmetic [24, Alg. 2] and Algorithm 3.1 are identical. In floating-point arithmetic, there are two main differences. The algorithm [24, Alg. 2] performs the same operations for computing Q_i as Algorithm 3.1, but in a different order. To analyse [24, Alg. 2] we would need to modify the analysis of Section 2 to account for this changed order. The error ρ_i is also calculated differently, which allows

[1] https://uk.mathworks.com/help/matlab/ref/svdsketch.html

Algorithm 3.1 Given $A \in \mathbb{R}^{m\times n}$ with $m \geq n$ and a tolerance $\epsilon > 0$, 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 $\|A - QB\|_F / \|A\|_F \leq \epsilon$. The parameter b is a block size and q determines the number of power iterations in the inner loop.

1: $Q = [\,]$, $B = [\,]$, $A_1 = A$, $\rho_1 = 1$
2: **for** $i = 1 : \mathtt{its}_{\mathtt{max}}$ **do**
3: Draw a Gaussian matrix $\Omega \in \mathbb{R}^{n\times b}$.
4: $Y = A_i\Omega$
5: $Q_i = \mathtt{qr}(Y, 0)$
6: **for** $j = 1 : q$ **do**
7: Compute $Y = A_i^T Q_i$ and $Q_i = \mathtt{qr}(Y, 0)$.
8: Compute $Y = A_i Q_i$ and $Q_i = \mathtt{qr}(Y, 0)$.
9: **end for**
10: $Q_i = \mathtt{qr}(Q_i - \sum_{j=1}^{i-1} Q_j Q_j^T Q_i, 0)$
11: $B_i = Q_i^T A_i$
12: $A_{i+1} = A_i - Q_i B_i$
13: $\rho_i = \|A_{i+1}\|_F / \|A\|_F$
14: If $\rho_i \leq \epsilon$ then quit.
15: **end for**
16: $Q = [Q_1 \ \cdots \ Q_i]$, $B = [B_1^T \ \cdots \ B_i^T]^T$

one to avoid retaining and computing the Frobenius norm of $\|A_{i+1}\|_F$ at each iteration, as is done in Algorithm 3.1. This new method of calculating the error introduces a limitation on the accuracy of the computation [24, sec. 3.3]. In the remainder of this work we focus on Algorithm 3.1, with the expectation that our analysis can be adapted to [24, Alg. 2].

3.1. Error analysis of fixed-precision algorithm. Here we present a framework for analyzing the effect of rounding errors on Algorithm 3.1. 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 $\|A - \widehat{Q}_i\widehat{B}_i\|_F$. From any of Theorems 2.3, 2.5, 2.6, and 2.7 we know that, to first order in u,

$$\|A - \widehat{Q}_i\widehat{Q}_i^T A\|_F \leq \|A - Q_iB_i\|_F + uf(m,n,b)\|A\|_F, \tag{3.1}$$

where Q_i and B_i are the exact matrices and the form of f depends on whether we incorporate power iterations in the rangefinder algorithm and 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 3.1) serves as our relative error. Define

$$\widehat{P}_i = I - \widehat{Q}_i\widehat{Q}_i^T, \qquad P_i = I - Q_iQ_i^T,$$

In exact arithmetic, the error after t iterations of the outer loop in Algorithm 3.1 is given by

$$\rho_t = \frac{\|P_t \ldots P_2P_1A\|_F}{\|A\|_F}. \tag{3.2}$$

In floating-point arithmetic it is given by

$$\widehat{\rho}_t = \frac{\|\widehat{P}_t \dots \widehat{P}_2 \widehat{P}_1 A\|_F}{\|A\|_F}. \tag{3.3}$$

Bounding the difference between (3.2) and (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 2. This term amounts to the $\|FA\|_F$ terms we bounded in (2.16), (2.17) and (2.25). In the case of both Theorems 2.3 and 2.5 (and their probabilistic extensions), the contribution of the $\|FA\|_F$ term is what gives the leading order rounding error term. For the sake of simplicity then, we write

$$\widehat{P}_i A = P_i A + \Delta_i, \quad \|\Delta_i\|_F \leq uf(m,n,b)\|A\|_F, \tag{3.4}$$

where, as described in (3.1), the precise form of f depends on the specific analysis and algorithm 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

$$\widehat{\rho}_t \leq \rho_t + \frac{\left\| \sum_{i=1}^{t-1} P_t \dots P_{i+1} \Delta_i + \Delta_t \right\|_F}{\|A\|_F} \leq \rho_t + \sum_{i=1}^{t} \|\Delta_i\|_F / \|A\|_F.$$

Finally, we have

$$\widehat{\rho}_t \leq \rho_t + tuf(m,n,b), \tag{3.5}$$

where the general form of (3.5) is the same as that of the bounds of section 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 3.1 is identical to that of Algorithm 1.1 [18, sect. 4]. Therefore, after t iterations of Algorithm 3.1 with block size b, we can identify ρ_t with the relative approximation error of Algorithm 1.1 with $\Omega \in \mathbb{R}^{n \times k}$ where $k = bt$.

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 2.2 to guide how to deploy low precisions. Algorithm 3.2 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.

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

Algorithm 3.2 Given $A \in \mathbb{R}^{m\times n}$ with $m \geq n$ and a relative tolerance $\epsilon > 0$, this algorithm computes a matrix $Q \in \mathbb{R}^{m\times b}$ with orthonormal columns such that $\|A - QQ^TA\|_F \leq \epsilon$. 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 $1 > \epsilon_1 > \cdots > \epsilon_p = \epsilon$ are given.

$Q = [\,]$, $B = [\,]$, $A_1 = A$, $\rho_1 = 1$
for $i = 1 : \texttt{its}_{\texttt{max}}$ **do**
  Draw a Gaussian matrix $\Omega \in \mathbb{R}^{n\times b}$.
  Find the largest j, $1 \leq j \leq p$ such that $\rho_i > \epsilon_j$.
  $Y = A_i\Omega$ at precision u_j.
  $Q_i = \texttt{qr}(Y, 0)$ at precision u_j
  for $j = 1 : q$ **do**
    Compute $Y = A_i^TQ_i$ and $Q_i = \texttt{qr}(Y, 0)$ at precision u_j.
    Compute $Y = A_iQ_i$ and $Q_i = \texttt{qr}(Y, 0)$ 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 $\rho_i \leq \epsilon$ then quit.
end for
$Q = [Q_1\ \cdots\ Q_i]$, $B = [B_1^T\ \cdots\ B_i^T]^T$

and a user-specified parameter to set these tolerances. In the problem setup we have a global tolerance ϵ, a sequence of available precisions $u_j, j = 1\colon p$, an $m \times n$ matrix A, and a block size b. For now, assume we do not perform any power iterations. The basic idea is simple: for t iterations at precision u_j, if $t\sqrt{mnb}u_j\rho_t$, from (3.5) with $f(m, n, b)$ chosen to be $\sqrt{mnb}$ from Table 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 $\theta\sqrt{mnb}u_j\rho_t$. 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

$$\epsilon_j = \begin{cases} \epsilon/(\theta\sqrt{mnb}u_{j+1}), & j = 1 : p-1, \\ \epsilon, & j = p. \end{cases} \tag{3.6}$$

This choice means that if $\rho_t < \epsilon_j$, when we switch to precision u_{j+1} we know that $\theta\sqrt{mnb}u_{j+1}\rho_t < \epsilon$. 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. If we include power iterations in the computation, the leading probabilistic rounding error contribution (see Table

2.1) is $\sqrt{m}bu$ and so we set

$$\epsilon_j = \begin{cases} \epsilon/(\theta\sqrt{m}bu_{j+1}), & j = 1:p-1, \\ \epsilon, & j = p. \end{cases} \tag{3.7}$$

3.3. Reorthonormalization. The reorthonormalization steps in Line 10 of Algorithm 3.1 and Line 11 of Algorithm 3.2 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 1.2, 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 3.2 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 (3.5). For this reason, we only allow precisions to be used in Algorithm 3.2 which have a unit roundoff less than the specified relative tolerance.

From the point of view of the error analysis of section 2, the orthonormalization step simply changes the constant slightly in (2.10) and subsequent bounds. It does not change the form of the final bounds, so Theorems 2.3, 2.5, 2.6, and 2.7 all remain valid with orthonormalization.

3.4. Numerical experiments. We now test the performance of Algorithm 3.2. 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 1.2. 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 3.2 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 3.1 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 (2.33) with $n = 500, r = 100$, $p = 2$, and $\phi = 1$.
- Type 3. Exponential decay[21, Sec. 7.3.1]. We generate random orthogonal matrices $U, V \in \mathbb{R}^{n\times n}$ from the Haar distribution, as in (2.33), and define

  $$A = UDV^T, \quad D = \operatorname{diag}(I_r, 10^{-p}, 10^{-2p}, \ldots, 10^{-(n-r)p}). \tag{3.8}$$

  We take $n = 500$, $r = 100$ and $p = 0.1$.

[2] https://github.com/higham/chop

TABLE 3.1
Iteration counts for experiments with Type 1 matrices with various relative tolerances ϵ and choices of θ.

	$\epsilon = 10^{-1}$			$\epsilon = 10^{-3}$			$\epsilon = 10^{-5}$			$\epsilon = 10^{-7}$		
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

TABLE 3.2
Iteration counts for experiments with Type 2 matrices with various relative tolerances ϵ and choices of θ.

	$\epsilon = 10^{-1}$			$\epsilon = 10^{-2}$			$\epsilon = 10^{-3}$			$\epsilon = 10^{-4}$		
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

In Tables 3.1, 3.2, and 3.3, (t_h, t_s, t_d) denote the number of iterations in half, single, and double precision respectively. We use (3.7) 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 3.2 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 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 3.2, 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.

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

TABLE 3.3
Iteration counts for experiments with Type 3 matrices with various relative tolerances ϵ and choices of θ.

	$\epsilon = 10^{-1}$			$\epsilon = 10^{-3}$			$\epsilon = 10^{-5}$			$\epsilon = 10^{-7}$		
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

order rounding error terms in Table 2.1. For the power iteration (Algorithms 1.2 and 2.2), 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 3.2 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. All data and codes supporting this work are available at https://github.com/michaelc100/Mixed-Precision-RandNLA.

REFERENCES

[1] Haim Avron, Petar Maymounkov, and Sivan Toledo. Blendenpik: Supercharging LAPACK's least-squares solver. *SIAM J. Sci. Comput.*, 32(3):1217–1236, 2010.

[2] Erin Carson and Ieva Daužickaitė. Single-pass Nyström approximation in mixed precision. *arXiv e-prints*, 2022. arXiv:2205.13355.

[3] Michael P. Connolly and Nicholas J. Higham. Probabilistic rounding error analysis of Householder QR factorization. MIMS EPrint 2022.5, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, February 2022. 16 pp.

[4] Michael P. Connolly, Nicholas J. Higham, and Theo Mary. Stochastic rounding and its probabilistic backward error analysis. *SIAM J. Sci. Comput.*, 43(1):A566–A585, 2021.

[5] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Fourth edition, Johns Hopkins University Press, Baltimore, MD, USA, 2013. xxi+756 pp. ISBN 978-1-4214-0794-4.

[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):217–288, 2011.

[7] Nathan Halko, Per-Gunnar Martinsson, Yoel Shkolnisky, and Mark Tygert. An algorithm for the principal component analysis of large data sets. *SIAM J. Sci. Comput.*, 33(5): 2580–2594, 2011.

[8] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Second edition, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002. xxx+680 pp. ISBN 0-89871-521-0.

[9] Nicholas J. Higham. *Functions of Matrices: Theory and Computation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008. xx+425 pp. ISBN 978-0-898716-46-7.

[10] Nicholas J. Higham and Theo Mary. A new approach to probabilistic rounding error analysis. *SIAM J. Sci. Comput.*, 41(5):A2815–A2835, 2019.

[11] Nicholas J. Higham and Theo Mary. Sharper probabilistic backward error analysis for basic linear algebra kernels with random data. *SIAM J. Sci. Comput.*, 42(5):A3427–A3446, 2020.

[12] Nicholas J. Higham and Srikara Pranesh. Simulating low precision floating-point arithmetic. *SIAM J. Sci. Comput.*, 41(5):C585–C602, 2019.

[13] Roger A. Horn and Charles R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, Cambridge, UK, 1991. viii+607 pp. ISBN 0-521-30587-X.

[14] Ilse C. F. Ipsen and Hua Zhou. Probabilistic error analysis for inner products. *SIAM J. Matrix Anal. Appl.*, 41(4):1726–1741, 2020.

[15] N. Kishore Kumar and J. Schneider. Literature survey on low rank approximation of matrices. *Linear Multilinear Algebra*, 65(11):2212–2244, 2017.

[16] Edo Liberty, Franco Woolfe, Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. Randomized algorithms for the low-rank approximation of matrices. *Proceedings of the National Academy of Sciences*, 104(51):20167–20172, 2007.

[17] Per-Gunnar Martinsson and Joel Tropp. Randomized numerical linear algebra: Foundations & algorithms. *Acta Numerica*, 29:403–572, 2020.

[18] Per-Gunnar Martinsson and Sergey Voronin. A randomized blocked algorithm for efficiently computing rank-revealing factorizations of matrices. *SIAM J. Sci. Comput.*, 38(5):S485–S507, 2016.

[19] Vladimir Rokhlin and Mark Tygert. A fast randomized algorithm for overdetermined linear least-squares regression. *Proc. Nat. Acad. Sci.*, 105(36):13212–13217, 2008.

[20] G. W. Stewart. The efficient generation of random orthogonal matrices with an application to condition estimators. *SIAM J. Numer. Anal.*, 17(3):403–409, 1980.

[21] Joel A. Tropp, Alp Yurtsever, Madeleine Udell, and Volkan Cevher. Practical sketching algorithms for low-rank matrix approximation. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1454–1485, 2017.

[22] Joel A. Tropp, Alp Yurtsever, Madeleine Udell, and Volkan Cevher. Streaming low-rank matrix approximation with an application to scientific simulation. *SIAM J. Sci. Comput.*, 41(4): A2430–A2463, 2019.

[23] Franco Woolfe, Edo Liberty, Vladimir Rokhlin, and Mark Tygert. A fast randomized algorithm for the approximation of matrices. *Applied and Computational Harmonic Analysis*, 25(3): 335–366, 2008.

[24] Wenjian Yu, Yu Gu, and Yaohang Li. Efficient randomized algorithms for the fixed-precision low-rank matrix approximation. *SIAM J. Matrix Anal. Appl.*, 39(3):1339–1359, 2018.