École polytechnique fédérale de Lausanne

Semester project spring 2023

Master in Computational Science and Engineering

Mixed precision single-pass algorithms for randomized low-rank matrix approximations

Author:
Valentin Comment

Supervisor:
Daniel Kressner

Contents

1 Introduction

Low-rank matrix approximations play a central role in today's data analysis and scientific computing. It has recently been shown [5] that randomization offers a powerful tool for performing low-rank matrix approximations. These techniques are usually simple and effective. When compared to standard deterministic algorithms, the randomized methods are often faster and surprisingly more robust. On the other hand, while scientific computing has traditionally used single precision and double precision floating-point arithmetic, the use of lower precisions has become more and more of interest because of the speed increase, the reduced communication and the lower energy costs they bring. But the use of lower precisions produces results of correspondingly low accuracy. A variety of mixed precision algorithms and techniques have been developed [8] and they often aim to produce the same quality as algorithms running in fixed precision but at much lower cost.

In this semester project, we will study the use of mixed precisions for randomized low-rank approximations algorithms. The project is structured as follows. In section 2, we will present a randomized method for approximating a symmetric positive semidefinite matrix called the Nyström method. We will give two different algorithms that implement it and discuss the technical details. Then in Sections 3 and 4, we will compare the stability and the behavior of both algorithms using numerical experiments on synthetic matrices and real data. Finally, in Section 5, we will derive the rounding error analysis of the well-known randomized SVD method for approximating general matrices and extend it to an algorithm for the fixed-precision problem, from which we propose a mixed-precision version.

2 Presentation of the Nyström method

Let $A \in \mathbb{R}^{n\times n}$ be a symmetric positive semi-definite (SPSD) matrix. The goal we want to achieve is to construct a low rank k approximation $\hat{A}_k$ of A using general random projections. Since A is SPSD, we can take advantage of its structure and use a natural way to construct its low rank approximation using the Nyström method [5, Section 5.4], which gives a low rank k approximation of the form :

$$\hat{A}_k = (AX)(X^TAX)^{\dagger}(AX)^T \tag{1}$$

where $X \in \mathbb{R}^{n\times k}$ is a random matrix and $\dagger$ denotes the Moore-Penrose pseudoinverse.
In this semester project, we are going to study two different mixed precisions algorithms that implement the Nyström method. Those algorithms have the same structure but implement the pseudoinverse in different ways. As done in [2, Algorithm 2.1], the first algorithm uses a Cholesky factorisation on X^TAX, since it is an SPSD matrix, to compute the pseudoinverse of this matrix. As the Cholesky factorisation is inherently unstable [7, Chapter 10], a shift is added to make it stable. The precision u denotes the double precision and the precision u_p denotes a lower or equal precisions than u. The steps where no precision is indicated are considered to run in double precision otherwise mentioned. We will use the same notation for the next algorithm.

Algorithm 2.1 Stabilised Nyström approximation for SPSD matrix A in mixed precisions u and u_p using the Cholesky factorisation. Otherwise mentioned, the working precision is in double precision u.

Input: SPSD matrix $A \in \mathbb{R}^{n\times n}$ stored in precision u_p, target rank k, oversampling parameter l
Output: Approximate eigendecompostion $A \approx U\Lambda U^T$

1: Sample $\Omega \in \mathbb{R}^{n\times k+l}$ with i.i.d. $\mathcal{N}(0,1)$ entries
2: Form the QR decomposition $[Q, \sim] = \text{qr}(\Omega, 0)$
3: **for** $i = 1 : n$ **do**
4: $\quad T = A(:, i) \times Q(i, :)$ in precision u_p
5: $\quad Y = Y + T$ in precision u_p
6: **end for**
7: Compute the shift $\mu = \epsilon_p * \|Y\|_F$
8: Compute $Y_\mu = Y + \mu Q$
9: Compute $B = Q^T Y_\mu$
10: Form the Cholesky factorisation $C = \text{chol}((B + B')/2)$, C is upper triangular
11: Solve $F = Y_\mu / C$
12: Form the economy size SVD $[U, \Sigma, \sim] = \text{svd}(F, 0)$
13: Truncate U and Σ to the target rank k if necessary: $U = U(:, 1 : k)$, $\Sigma = \Sigma(1 : k, 1 : k)$
14: Remove the shift: $\Lambda = max(0, \Sigma^2 - \mu I)$
15: **return** U, Λ

The second algorithm uses the ϵ-pseudoinverse to compute the pseudoinverse of X^TAX, as in [10, Algorithm 2.1], to make it more stable. Taking advantage of the fact that X^TAX is symmetric, the eigendecomposition was chosen for this particular step.

Algorithm 2.2 Stabilised Nyström approximation for SPSD matrix A in mixed precisions u and u_p using the ϵ-pseudoinverse. Otherwise mentioned, the working precision is in double precision u.

Input: SPSD matrix $A \in \mathbb{R}^{n\times n}$ stored in precision u_p, target rank k, oversampling parameter l
Output: Approximate eigendecompostion $A \approx U\Lambda U^T$

1: Sample $\Omega \in \mathbb{R}^{n\times k+l}$ with i.i.d. $\mathcal{N}(0,1)$ entries
2: Form the QR decomposition $[Q, \sim] = \text{qr}(\Omega, 0)$
3: **for** $i = 1 : n$ **do**
4: $\quad T = A(:, i) \times Q(i, :)$ in precision u_p
5: $\quad Y = Y + T$ in precision u_p
6: **end for**
7: Compute $B = Q^TY$
8: Compute the shift $\mu = \epsilon_p * \|Y\|_F$
9: Form the eigendecomposition $[\hat{U}, \hat{\Lambda}] = \text{eig}((B + B')/2)$
10: Find the indexes i s.t. $|\hat{\Lambda}(i, i)| \geq \mu$, store them in idx.
11: Truncate the eigenvalues smaller in absolute value than μ : $\tilde{U} = \hat{U}(:, idx)$, $\tilde{\Lambda} = \hat{\Lambda}(idx, idx)$
12: Solve $F = Y\tilde{U}/\tilde{\Lambda}^{\frac{1}{2}}$
13: Form the economy size SVD $[U, \Sigma, \sim] = \text{svd}(F, 0)$
14: Set $\Lambda = \Sigma^2$
15: Truncate U and Λ to the target rank k if necessary: $U = U(:, 1 : k)$, $\Lambda = \Lambda(1 : k, 1 : k)$
16: **return** U, Λ

A few comments need to be made about both algorithms:

- In both methods, we compute either the cholesky factorization or the epsilon-pseudoinverse of $(B+B^T)/2$ instead of B. This is because the matrix B is not symmetric if we perform the right multiplication of A with Q in another precision than the left multiplication of A with Q^T. Since we want to preserve the symmetry in the final approximation, it is necessary to take the symmetric part of B.

- For the steps 4 and 5 in both algorithms, the computation in lower precision in Matlab is done with the *chop* function of [6] for half precision, with the *single* function of Matlab for single precision and directly for double precision.

- One may wonder if computing the epsilon-pseudo inverse via the eigenvalue decomposition is equivalent to computing it via the SVD as described in [10, Section 5], in step 11 of Algorithm 2.2. It is in fact equivalent in our case since $(B + B^T)/2$ is symmetric. To see that we remind that for a symmetric matrix, the relationship between its eigendecomposition and its singular value decomposition is the following :

  Let $(B + B^T)/2 = U\Lambda U^T$ be its eigendecomposition, where we assume that the eigenvalues in the diagonal of the diagonal matrix Λ are ordered in decreasing absolute value. Let $D = diag(sign(\Lambda_{11}), sign(\Lambda_{22}), ..., sign(\Lambda_{nn}))$. Then we have the following relationship between the eigendecomposition and the SVD of $(B + B^T)/2$:

  $$(B + B^T)/2 = U\Lambda U^T = U\Lambda DDU^T = U\Sigma DU^T = U\Sigma\tilde{V}^T,$$

  where $\Sigma := \Lambda D$ is the diagonal matrix containing the singular values of $(B + B^T)/2$ in decreasing order, $\tilde{V} := UD$ is orthonormal.
  From this relation, since Σ is unique for a given matrix, it is clear that truncating the singular values smaller than epsilon for the epsilon-pseudo inverse is equivalent to truncating the eigenvalues smaller in absolute value, than epsilon. Note that when Matlab performs the eigendecomposition, the order of the eigenvalues is decreasing instead of absolutely decreasing. This does not change the result since we can use permutation matrices, which are orthonormal, to get to our result.

- Now the last point is on step 12 of Algorithm 2.2 where we use the square root of $\tilde{\Lambda}$. To do that, we need to be sure that $\tilde{\Lambda}$ is positive semi-definite after the truncation of $\hat{\Lambda}$ in step 11. From [2, proof of Theorem 2.2], we know that in Algorithm 2.1, $(B + B')/2 = (Q^TY + Y^TQ)/2 + \mu I$ is positive semi-definite. Since the computed quantities Y and Q are the same for both Algorithm 2.1 and 2.2, taking $\epsilon = \mu$ to perform the epsilon-pseudoinverse in Algorithm 2.2 ensures that the resulting $\tilde{\Lambda}$ is positive semi-definite.

To see that, consider any eigenvalue of $(Q^TY + Y^TQ)/2$. From the development above, we know that $\forall i \in [1, n], \lambda_i((Q^TY + Y^TQ)/2) + \epsilon \geq 0$. This means that the negative eigenvalues of $(Q^TY + Y^TQ)/2$ are smaller, in absolute value, than epsilon, meaning that all the negative eigenvalues will be removed after the truncation in steps 10-11 of Algorithm 2.2, ensuring the positive semi-definitiveness of $\tilde{\Lambda}$.

3 Numerical results : Question 1

In this section, a comparison of Algortihm 2.1 and Algorithm 2.2 from a point of view of stability is done. We use the following test matrices $A \in \mathbb{R}^{n \times n}$:

- *Polynomial decay:*

$$A = diag(1^{-p}, 2^{-p}, ..., n^{-p})$$

  Where p takes the values 0.5, 1 and 2.

- *Exponential decay:*

$$A = diag(1, 10^{-q}, 10^{-2q}, ..., (10)^{-(n-1)q})$$

  Where q takes the values 0.1, 0.25 and 1.

- *Stair decay:*

$$A = diag(1, 0.99, 0.98, \frac{1}{10}, \frac{0.99}{10}, \frac{0.98}{10}, ...)$$

3.1 Comparison of different precisions

For this experiment, we plot the absolute error in the 2-norm of Algorithms 2.1 and 2.2 versus the rank of the low rank approximation k using the double, single and half precisions for the matrix-matrix product. The span of k depends on the singular decay of the matrix considered. We set $n = 10^2$ and the oversampling parameter to be $l = 0$. Moreover, every experiment is performed 10 times and we take the mean of the absolute error. Finally, for every experiment, we set the random seed to be equal to the number of the experiment (i.e. if we are doing the ith repetition, the random seed will be i) for reproducibility. We obtained the following results:

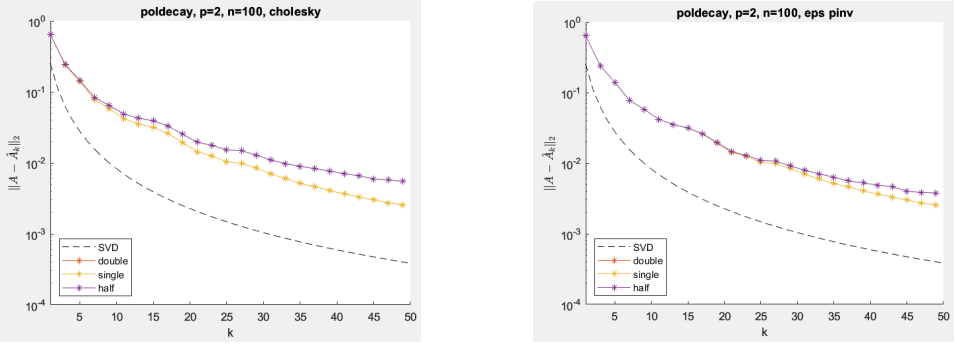

Figure 1: Polynomial decay case with $p = 2$. SVD denotes the optimal truncated SVD.
$A = diag(1^{-p}, 2^{-p}, ..., n^{-p})$

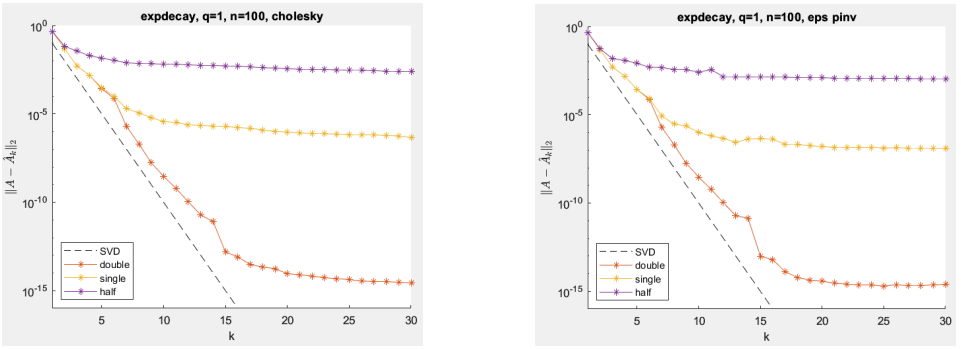

Figure 2: Exponential decay case with $q = 1$
$A = diag(1, 10^{-q}, 10^{-2q}, ..., (10)^{-(n-1)q})$

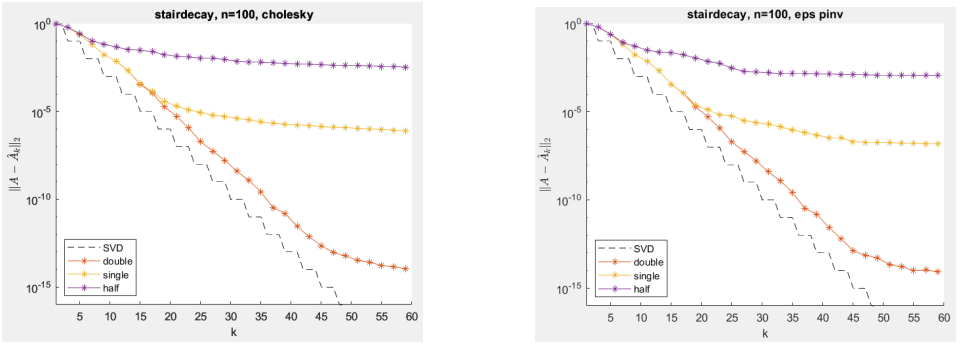

Figure 3: Stair decay case
$A = diag(1, 0.99, 0.98, \frac{1}{10}, \frac{0.99}{10}, \frac{0.98}{10}, ...)$

In light of Figures 1, 2 and 3 we make three main observations :

- In every example, Algorithm 2.2 seems slightly better than Algorithm 2.1.
- When the singular decay is slow, for both Algorithms, the rounding error does not seem to impact much.
- When the singular decay is fast, for both Algorithms, the error seem to converge quickly to a value corresponding to the unit roundoff of the lower precision used.

3.2 Comparison when all the steps are computed in single precision

Now, we modified the working precision used in both Algorithm 2.1 and 2.2 to see if experimentally there are a lot of changes with Figures 1, 2 and 3. To do this, we implemented both Algorithm 2.1 and Algorithm 2.2 with all their steps executed in single precision and compared their absolute error with the absolute error of Figure 1, 2 and 3 in the single precision case. We only display the cases of the exponential decay with $q = 1$ and stair decay as they give the most relevant informations:

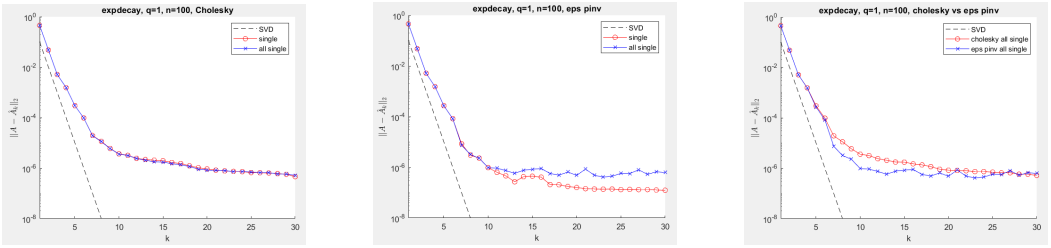

Figure 4: Exponential decay case with $q = 1$. The third graph compares the results of both Algorithms 2.1 and 2.2 when all the steps are computed in single precision.

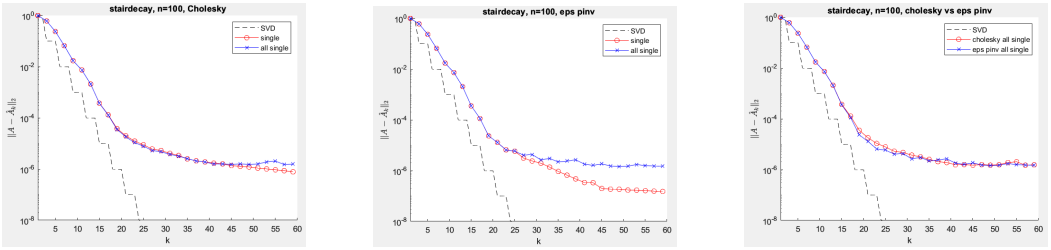

Figure 5: Stair decay case.

In light of Figures 4 and 5, a few observations can be made:

- For Algorithm 2.2, in both examples, the absolute error when everything is computed in single precision seems to be worse than when only the matrix-matrix product AQ is computed in single precision.
- For Algorithm 2.1, in both examples, the absolute error in the two cases considered seems to be much closer than for Algorithm 2.2.
- As displayed in the third graph in both cases, the absolute error when everything is computed in single precision for Algorithm 2.2 seems to be slightly better but not by much.

4 Numerical results : Question 2

In this section, we test both Algorithm 2.1 and Algorithm 2.2 on a class of matrix commonly encountered in Machine Learning and Data Analysis : the Gaussian Random Basis Function Kernel (RBFK) matrices. In this example, we are only going to show the results on dense RBKF matrices. From a matrix $A \in \mathbb{R}^{n \times d}$, we construct its dense RBKF matrix A^{σ} in the following way :

$$A^{\sigma}_{ij} = exp(-\frac{\|x_i - x_j\|_2^2}{\sigma^2}).$$

Where x_i and x_j denote respectively the i'th and j'th row of A and σ is a parameter.
Table 1 resumes the information about the real data used from [1] to construct the dense RKBF matrices for the experiments where σ was set to be equal to one.

Name	Description	n	d
Abalone	Physical measurements of Abalone	4177	8

Table 1: Data used in the experiment from [1]. Here n denotes the number of rows, d the number of columns.

For the experiments, we measured the absolute error $\|A - \hat{A}_k\|_2$, where $\hat{A}_k$ is the low rank k approximation using one of the algorithms for $k \in \{1, 2, ..., 150\}$ and with no oversampling. We obtained the results displayed in Figure 6:

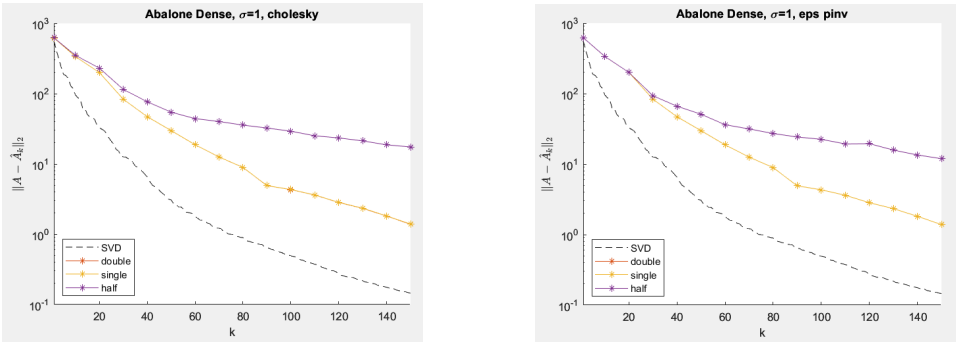

Figure 6: Absolute error of the low rank approximation of both algorithms for Abalone dense, when $\sigma = 1$.

As we can see in Figure 6, the behavior of the absolute error for both Algorithm is quite the same and follow the remark in Section 2 about the slow singular decay case. Moreover, we notice that Algorithm 2.2, seems to be slightly better again than Algorithm 2.1 for the half precision.

Since the singular values of the matrix above do not decay very quickly, we decided to construct a matrix that exhibits a much faster singular decay. The reason is when the singular decay is slow, the best low rank approximation absolute error will be high which may hide the effects of the roundoff error. To do that, we decided to construct a dense RBKF matrix as defined before but for a uniformly sample vector between 0 and 1 of size 500. We obtained the following results for $\sigma = 1$:

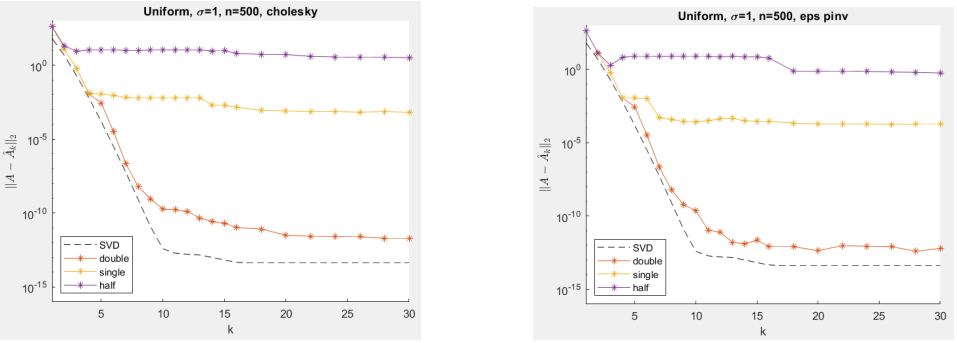

Figure 7: Absolute error of the low rank approximation of both algorithms for the RBKF matrix constructed from a uniform vector of size 500, when $\sigma = 1$.

From Figure 7, we make the following observations:

- For both methods, the roundoff errors dominate the approximation error when the singular values are really small. Furthermore, depending on the precision used, we can expect the contribution of the roundoff errors to be proportional to the unit roundoff of the lower precision used since there is an increment of the order of 10^8 between the simple and double precision iteration (corresponding approximately to $\frac{u_{simple}}{u_{double}}$) and an increment of order 10^4 between the half and simple precision iteration (corresponding approximately to $\frac{u_{half}}{u_{simple}}$).
- Algorithm 2.2 which uses the epsilon-pseudoinverse seems to be a bit more precise than Algorithm 2.1, which uses the cholesky factorisation.

5 Presentation of the mixed precision randomized SVD

It is well known that the randomized SVD alogrithm (RSVD) is an efficient random projection based method to give a low rank approximation SVD of a matrix. In this section, we will first give a roundoff error analysis of the RSVD algorithm and extend it to the power iteration variant. Then, those results will be used to construct a mixed precision adaptive rank RSVD algorithm, which, on a given precision tolerance, should be faster and use less space for approximately the same rank of approximation than the classical adaptive rank RSVD algorithm.

Throughout this section, we will use the standard model of floating point arithmetic [7, Section 2.2]. Following [7], we define :

$$\gamma_n = \frac{nu}{1-nu}, \quad \tilde{\gamma}_n = \frac{cnu}{1-cnu},$$

where c is a small constant independent of n and u is the unit roundoff of the working precision.

5.1 Roundoff error analysis

The main idea of the RVSD algorithm can be split naturally into two computational stages. The first is to construct a low dimensional subspace that captures the action of the matrix. The second is to restrict the matrix to the subspace and then compute a SVD of the reduced matrix. Algorithm 5.1 formalizes the idea described :

Algorithm 5.1 Randomized SVD

Input: Matrix $A \in \mathbb{R}^{m\times n}$ with $m \geq n$, target rank k
Output: 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: Sample $\Omega \in \mathbb{R}^{n\times k}$ with i.i.d. $\mathcal{N}(0,1)$ entries
2: Compute $Y = A\Omega$
3: Form the QR decomposition $[Q, \sim] = \text{qr}(Y, 0)$
4: Compute $B = Q^T A$
5: Form the economy size SVD $[\tilde{U}, \Sigma, V] = \text{svd}(B, 0)$
6: Set $U = Q\tilde{U}$
7: **return** U, Σ, V

Theorem 5.1. *Let $A \in \mathbb{R}^{m\times n}$ with $m \geq n$ and assume that the SVD on line 5 of Algorithm 5.1 is computed exactly. Let $\hat{U}$ be the orthogonal matrix computed in line 6 of Algorithm 5.1. Then Algorithm 5.1 produces a computed SVD $A \approx \hat{U}\Sigma V^T$ satisfying :*

$$\begin{aligned}\|A - \hat{U}\Sigma V^T\|_F \leq ((\gamma_n + \gamma_n\tilde{\gamma}_{mk} + \tilde{\gamma}_{mk})\|\Omega\|_F\|(\tilde{V}^T\Omega)^\dagger\|_F + (\tilde{\gamma}_{mk} + k^{\frac{1}{2}}\tilde{\gamma}_{mk}(1+\tilde{\gamma}_{mk})))\|A\|_F \\ + \sqrt{\|\Sigma_2\|_F^2 + \|\Sigma_2\tilde{V}_\perp^T\Omega(\tilde{V}^T\Omega)^\dagger\|_F^2},\end{aligned} \tag{2}$$

where $\tilde{V}$ contains the first r right singular vector of A, Σ_2 is the diagonal matrix containing the singular values $\sigma_{r+1}...\sigma_n$ and $\tilde{V}_\perp$ contains the corresponding right singular vectors, with r such that $r + 4 \leq k$.

Proof. The first part of the proof consists in analysing the rounding error induced from steps 2-6 of Algorithm 5.1.
Using [7, Section 3.5], we know that the rounding error induced in line 2 is such that:

$$\hat{Y} = Y + \Delta Y_1, \|\Delta Y_1\|_F \leq \gamma_n\|A\|_F\|\Omega\|_F$$

Now from [7, Theorem 19.4], we know that a Householder QR factorisation of $\hat{Y}$ yields to $\hat{Y} + \Delta Y_2 = \tilde{Q}\hat{R}$, where

$$\|\Delta Y_2\|_F \leq \tilde{\gamma}_{mk}\|\hat{Y}\|_F \leq \tilde{\gamma}_{mk}(\|Y\|_F + \|\Delta Y_1\|_F) \leq \tilde{\gamma}_{mk}(1+\gamma_n)\|A\|_F\|\Omega\|_F$$

.

Thus after line 3, we have $Y + \Delta Y = \tilde{Q}\hat{R}$, with

$$\|\Delta Y\|_F = \|\Delta Y_1 + \Delta Y_2\|_F \leq (\gamma_n + \gamma_n\tilde{\gamma}_{mk} + \tilde{\gamma}_{mk})\|A\|_F\|\Omega\|_F$$

.

For lines 4 and 6, we consider that Q^T and Q are applied in factored form to A and $\tilde{U}$ respectively. From [7, Lemma 19.3], we obtain:

$$\hat{B} = \tilde{Q}^T A + \Delta A, \|\Delta A\|_F \leq \tilde{\gamma}_{mk}\|A\|_F$$

$$\hat{U} = \tilde{Q}\tilde{U} + \Delta U, \|\Delta U\|_F \leq \tilde{\gamma}_{mk}\|U\|_F = k^{\frac{1}{2}}\tilde{\gamma}_{mk}$$

Now that we have all the elements of the rounding analysis, we can move onto the bound of the error of the approximation returned. First, we have :

$$\begin{aligned}A - \hat{U}\Sigma V^T &= A - \tilde{Q}\tilde{U}\Sigma V^T - \Delta U\Sigma V^T \\ &= A - \tilde{Q}\hat{B} - \Delta U\Sigma V^T \\ &= A - \tilde{Q}\tilde{Q}^T A - \tilde{Q}\Delta A - \Delta U\Sigma V^T \\ &= (I - \tilde{Q}\tilde{Q}^T)A + E.\end{aligned}$$

where

$$\begin{aligned}\|E\|_F &\leq \|\tilde{Q}\Delta A\|_F + \|\Delta U\Sigma V^T\|_F \\ &\leq \tilde{\gamma}_{mk}\|A\|_F + k^{\frac{1}{2}}\tilde{\gamma}_{mk}\|\Sigma V^T\|_F \\ &= \tilde{\gamma}_{mk}\|A\|_F + k^{\frac{1}{2}}\tilde{\gamma}_{mk}\|\hat{B}\|_F \\ &\leq (\tilde{\gamma}_{mk} + k^{\frac{1}{2}}\tilde{\gamma}_{mk}(1+\tilde{\gamma}_{mk}))\|A\|_F.\end{aligned}$$

Now it remains to bound $(I - \tilde{Q}\tilde{Q}^T)A$. Notice that for any X of suitable size, we have:

$$\|(I - \tilde{Q}\tilde{Q}^T)A\|_F \leq \|(I - \tilde{Q}\tilde{Q}^T)A\Omega X\|_F + \|(I - \tilde{Q}\tilde{Q}^T)A(I - \Omega X)\|_F$$

For the first term, we have:

$$(I - \tilde{Q}\tilde{Q}^T)A\Omega = A\Omega - \tilde{Q}(\hat{R} - \tilde{Q}^T\Delta Y) = A\Omega + \tilde{Q}\tilde{Q}^T\Delta Y - A\Omega - \Delta Y = -(I - \tilde{Q}\tilde{Q}^T)\Delta Y.$$

Using the fact that $\|I - \tilde{Q}\tilde{Q}^T\|_2 \leq 1$, we obtain the following bound for the first term :

$$\|(I - \tilde{Q}\tilde{Q}^T)A\Omega X\|_F \leq \|I - \tilde{Q}\tilde{Q}^T\|_2\|\Delta Y X\|_F \leq \|\Delta Y\|_F\|X\|_F$$

For the second term, we take $X = (\tilde{V}^T\Omega)^\dagger\tilde{V}^T$, where $\tilde{V}$ contains the first r right singular vector of A with r such that $r + 4 \leq k$. Now assuming $\tilde{V}^T\Omega$ has full rank, we obtain :

$$\begin{aligned}
\|(I - \tilde{Q}\tilde{Q}^T)A(I - \Omega X)\|_F^2 &\leq \|A(I - \Omega(\tilde{V}^T\Omega)^\dagger\tilde{V}^T)\|_F^2 \\
&= \|A(I - \tilde{V}\tilde{V}^T)(I - \Omega(\tilde{V}^T\Omega)^\dagger\tilde{V}^T)\|_F^2 \\
&\leq \|A(I - \tilde{V}\tilde{V}^T)\|_F^2 + \|A(I - \tilde{V}\tilde{V}^T)\Omega(\tilde{V}^T\Omega)^\dagger\tilde{V}^T\|_F^2 \\
&= \|\Sigma_2\|_F^2 + \|\Sigma_2\tilde{V}_\perp^T\Omega(\tilde{V}^T\Omega)^\dagger\|_F^2,
\end{aligned}$$

where Σ_2 is a the diagonal matrix containing the singular values $\sigma_{r+1}...\sigma_n$ and $\tilde{V}_\perp$ contains the corresponding right singular vectors.

Regrouping everything ends the proof :

$$\begin{aligned}
\|A - \hat{U}\Sigma V^T\|_F &\leq \|\Delta Y\|_F\|(\tilde{V}^T\Omega)^\dagger\|_F + \sqrt{\|\Sigma_2\|_F^2 + \|\Sigma_2\tilde{V}_\perp^T\Omega(\tilde{V}^T\Omega)^\dagger\|_F^2} + \|E\|_F \\
&\leq ((\gamma_n + \gamma_n\tilde{\gamma}_{mk} + \tilde{\gamma}_{mk})\|\Omega\|_F\|(\tilde{V}^T\Omega)^\dagger\|_F + (\tilde{\gamma}_{mk} + k^{\frac{1}{2}}\tilde{\gamma}_{mk}(1 + \tilde{\gamma}_{mk})))\|A\|_F \\
&+ \sqrt{\|\Sigma_2\|_F^2 + \|\Sigma_2\tilde{V}_\perp^T\Omega(\tilde{V}^T\Omega)^\dagger\|_F^2}
\end{aligned}$$

□

Algorithm 5.1 tends to work well on matrices that exhibits some singular decay. However, when the singular spectrum is flat, it may not perform well. To improve the accuracy of the low rank approximation in this case, a common idea is to improve the singular decay while keeping the same singular vectors so that the relative weight of the singular vectors associated to the small singular values decreases. One way to do it is to apply the scheme of Algorithm 5.1 on $(AA^T)^qA$, where $q > 0$ is an integer, instead of A. This leads to the following algorithm :

Algorithm 5.2 Randomized SVD extended to power iteration

Input: Matrix $A \in \mathbb{R}^{m\times n}$ with $m \geq n$, target rank k, power q

Output: Approximate SVD $A \approx U\Sigma V^T$ where $U \in \mathbb{R}^{m\times k}$, $\Sigma \in \mathbb{R}^{k\times k}$, $V \in \mathbb{R}^{n\times k}$. All QR factorizations used are thin QR factorizations.

1: Sample $\Omega \in \mathbb{R}^{n\times k}$ with i.i.d. $\mathcal{N}(0, 1)$ entries
2: Compute $Y_1 = A\Omega$
3: Form the QR decomposition $[Q_1, \sim] = \text{qr}(Y_1, 0)$
4: **for** $i = 1 : q$ **do**
5: Compute $Y_{2i} = A^TQ_{2i-1}$ and form $[Q_{2i}, \sim] = \text{qr}(Y_{2i}, 0)$
6: Compute $Y_{2i+1} = AQ_{2i}$ and form $[Q_{2i+1}, \sim] = \text{qr}(Y_{2i+1}, 0)$
7: **end for**
8: Set $Q = Q_{2q+1}$
9: Compute $B = Q^TA$
10: Form the economy size SVD $[\tilde{U}, \Sigma, V] = \text{svd}(B, 0)$
11: Set $U = Q\tilde{U}$
12: **return** U, Σ, V

Note that the scheme of Algorithm 5.2 has a slight change as it iterates on the power q to form $(AA^T)^qA$, using QR factorizations, instead of directly computing $(AA^T)^qA$. This is because when the Algorithm 5.1 is

directly applied to $(AA^T)^q A$, it will fail to captures the singular vectors associated to singular values that are small relative to $\|A\|_2$.
Now we extend the analysis done in Theorem 5.1 to Algorithm 5.2:

Theorem 5.2. *Let $A \in \mathbb{R}^{m\times n}$ with $m \geq n$ and assume that the SVD on line 10 of Algorithm 5.2 is computed exactly. Let $\hat{U}$ be the orthogonal matrix computed in line 9 of Algorithm 5.2. Then Algorithm 5.2 with $q \geq 1$ produces a computed SVD $A \approx \hat{U}\Sigma V^T$ satisfying :*

$$\begin{aligned}\|A - \hat{U}\Sigma V^T\|_F \leq \|(I - QQ^T)A\|_F \\ + ((1 + 2k^{1/2})\tilde{\gamma}_{mk} + k^{1/2}\gamma_n(1 + \tilde{\gamma}_{mk}) + k^{1/2}\tilde{\gamma}_{mk}^2)\|A\|_F\end{aligned} \tag{3}$$

Proof. First, notice that part of the proof of 5.1 still applies, notably the expression of:

$$A - \hat{U}\Sigma V^T = (I - \tilde{Q}\tilde{Q}^T)A + E,$$

where,

$$\|E\|_F \leq (\tilde{\gamma}_{mk} + k^{\frac{1}{2}}\tilde{\gamma}_{mk}(1 + \tilde{\gamma}_{mk}))\|A\|_F,$$

still holds. Now introducing the exact Q, we have :

$$\|A - \hat{U}\Sigma V^T\|_F \leq \|(I - QQ^T)A\|_F + \|(QQ^T - \tilde{Q}\tilde{Q}^T)A\|_F + \|E\|_F,$$

Then we analyze the rounding error term of the last step of the power iteration $QR_{2q+1} = Y_{2q+1} = AQ_{2i}$ as in the beggining of the proof of Theorem 5.1. We have :

$$Y_{2q+1} + \Delta Y_{2q+1} = \tilde{Q}\hat{R}_{2q+1},$$

where,

$$\|\Delta Y_{2q+1}\|_F \leq (\gamma_n + \gamma_n\tilde{\gamma}_{mk} + \tilde{\gamma}_{mk})\|A\|_F\|\tilde{Q}\|_F = k^{\frac{1}{2}}(\gamma_n + \gamma_n\tilde{\gamma}_{mk} + \tilde{\gamma}_{mk})\|A\|_F$$

Purely for the purpose of the demonstration, we consider that all the QR factorizations performed in 5.2 are full QR factorizations with $Q_i \in \mathbb{R}^{m\times m}$ and $R_i = \begin{bmatrix} R_i' \\ 0 \end{bmatrix}$, $R_i' \in \mathbb{R}^{k\times k}$. We then set $Q = Q_{2q+1}I_{m,k} \in \mathbb{R}^{m\times k}$ with $I_{m,k} = I(:, 1:k)$. We have :

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

Thus we get :

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

Combining the results, we get :

$$\begin{aligned}\|(QQ^T - \tilde{Q}\tilde{Q}^T)A\|_F &= \|(QQ^T - \tilde{Q}\tilde{Q}^T)AQ_{2q}\|_F \\ &= \|(I - \tilde{Q}\tilde{Q}^T)\Delta Y_{2q+1}\|_F \\ &\leq k^{\frac{1}{2}}(\gamma_n + \gamma_n\tilde{\gamma}_{mk} + \tilde{\gamma}_{mk})\|A\|_F.\end{aligned}$$

Hence,

$$\begin{aligned}\|A - \hat{U}\Sigma V^T\|_F &\leq \|(I - QQ^T)A\|_F + k^{\frac{1}{2}}(\gamma_n + \gamma_n\tilde{\gamma}_{mk} + \tilde{\gamma}_{mk})\|A\|_F + (\tilde{\gamma}_{mk} + k^{\frac{1}{2}}\tilde{\gamma}_{mk}(1 + \tilde{\gamma}_{mk}))\|A\|_F \\ &= \|(I - QQ^T)A\|_F + ((1 + 2k^{1/2})\tilde{\gamma}_{mk} + k^{1/2}\gamma_n(1 + \tilde{\gamma}_{mk}) + k^{1/2}\tilde{\gamma}_{mk}^2)\|A\|_F\end{aligned}$$

□

We can now improve the worst case bounds derived in Theorem 5.1 and Theorem 5.2 using probabilistic error analysis. We give the two following results :

For a matrix-matrix multiplication $C = AB$, from [4, Theorem 4.9] the probabilistic roundoff analysis gives:

$$\hat{C} = C + \Delta C, |\Delta C| \leq \bar{\gamma}_n(\lambda) = |A||B|,$$

where $\bar{\gamma}_n(\lambda) = exp(\frac{\lambda\sqrt{n}u+nu^2}{1-u}) = \lambda\sqrt{n}u + O(u^2)$

For the Householder QR factorization, from [3, Theorem 4.4] the probabilistic roundoff analysis gives :

$$A + \Delta A = \tilde{Q}\hat{R}, \|\Delta a_j\|_2 \leq c\lambda\sqrt{k}\bar{\gamma}_m(\lambda)\|a_j\|_2 + O(u^2).$$

Using these results, probabilistic bounds of Theorem 5.1 and Theorem 5.2 can be derived straightforwardly by applying the same proofs but replacing the deterministic roundoff analysis by the probabilistic one. This leads to the next theorem :

Theorem 5.3. *Let $A \in \mathbb{R}^{m\times n}$ with $m \geq n$ and assume that the SVD on line 5 of Algorithm 5.1 and line 10 of Algorithm 5.2 is computed exactly. Let $\hat{U}$ be the orthogonal matrix computed in line 6 of Algorithm 5.1. Then Algorithm 5.1 produces a computed SVD $A \approx \hat{U}\Sigma V^T$ satisfying :*

$$\begin{aligned}\|A - \hat{U}\Sigma V^T\|_F \leq ((\bar{\gamma}_n(\lambda) + c\lambda k^{\frac{1}{2}}\bar{\gamma}_m(\lambda))\|\Omega\|_F\|(\tilde{V}^T\Omega)^\dagger\|_F + c\lambda k^{\frac{1}{2}}\bar{\gamma}_m(\lambda)(1+k^{\frac{1}{2}}))\|A\|_F \\ + \sqrt{\|\Sigma_2\|_F^2 + \|\Sigma_2\tilde{V}_\perp^T\Omega(\tilde{V}^T\Omega)^\dagger\|_F^2} + O(u^2),\end{aligned} \tag{4}$$

where $\tilde{V}$ contains the first r right singular vector of A, Σ_2 is the diagonal matrix containing the singular values $\sigma_{r+1}...\sigma_n$ and $\tilde{V}_\perp$ contains the corresponding right singular vectors, with r such that $r + 4 \leq k$.

Let $\hat{U}$ be the orthogonal matrix computed in line 9 of Algorithm 5.2. Then Algorithm 5.2 with $q \geq 1$ produces a computed SVD $A \approx \hat{U}\Sigma V^T$ satisfying :

$$\begin{aligned}\|A - \hat{U}\Sigma V^T\|_F \leq \|(I - QQ^T)A\|_F \\ + (c\lambda(2k + k^{1/2})\bar{\gamma}_m(\lambda) + k^{1/2}\bar{\gamma}_n(\lambda))\|A\|_F + O(u^2)\end{aligned} \tag{5}$$

5.2 Leading order rounding error terms

From, Theorem 5.1, Theorem 5.2, and Theorem 5.3, we notice that the error bounds are always of the form: exact error approximation + roundoff error. In this subesction, we will derive the leading order rounding error terms for Algorithm 5.1 and Algorithm 5.2 in the worst case bound and the probabilistic bounds. Those terms are going to be used later in the adaptive precision iterative rank Randomized SVD algorithm presented in the next section. To do that, we will consider that the sketch matrix Ω used in both algorithms is a Gaussian random matrix.

We first focus on the bound of Algorithm 5.1. To proceed, one key observation is needed. Since the Gaussian distribution is rotationally invariant, the matrix $\tilde{V}^T\Omega \in \mathbb{R}^{r\times k}$ is also Gaussian. Since $r + 4 \leq k$, from [5, Proposition 10.4] we have $\|(\tilde{V}^T\Omega)^\dagger\|_F \leq t\sqrt{k}$ where the bound holds with at least some probability that depends on t and r. Then, from [11, Theorem 4.1.1] , we can bound $\|\Omega\|_F \leq t\sqrt{kn}$ where again the bound holds with at least some probability that depends on k, n and t. Thus we have $\|(\tilde{V}^T\Omega)^\dagger\|_F = O(\sqrt{k})$ and $\|\Omega\|_F = O(\sqrt{kn})$. Finally, using the fact that $\gamma_n = O(nu)$ and $\tilde{\gamma}_{mk} = O(mku)$ we obtain the leading order rounding error term in the deterministic case for Algorithm 5.1:

$$\begin{aligned}((O(nu) + O(nu)O(mku) + O(mku))O(\sqrt{kn})O(\sqrt{k}) + O(mku)(1 + k^{\frac{1}{2}}(1 + O(mku))))\|A\|_F \\ = O(m\sqrt{n}k^2u)\|A\|_F\end{aligned}$$

For the probabilistic case, we remark that from the definition of $\bar{\gamma}_n(\lambda)$, we have $\bar{\gamma}_n(\lambda) = O(\sqrt{nu})$ and $\bar{\gamma}_m(\lambda) = O(\sqrt{mu})$. Thus this time we obtain for the probabilistic case of Algorithm 5.1 :

$$O(\sqrt{mn}k^{\frac{3}{2}}u)\|A\|_F$$

Using the same approximations for Algorithm 5.2, we obtain in the deterministic case :

$$((1 + k^{1/2})O(mku) + k^{1/2}O(nu)(1 + O(mku)) + k^{1/2}O(m^2k^2u^2))\|A\|_F$$
$$= O(mk^{\frac{3}{2}}u)\|A\|_F$$

For the probabilistic case, we obtain :

$$O(\sqrt{mku})\|A\|_F$$

5.3 Adaptive precision of iterative rank Randomized SVD

Up to now, we have analyzed two algorithms which give low rank approximations for a fixed-rank problem. Perhaps a more common situation computationally is the fixed-precision problem, where, given a tolerance, we want our computed low-rank approximation to be accurate at the precised tolerance. The fixed-precision rangefinder algorithm, proposed in [9, Section 5] follows naturally from Algorithms 5.1 and 5.2. Here, the approximation is build incrementally by increasing the rank multiple time by a block size b until the prescribed tolerance is reached. In the following, we study a modification of this algorithm where mixed precisions is introduced to speed up and lower the space of the algorithm while mitigating the damages to the approximation. We use the different results derived in the previous subsections to give a practical heuristic to know when to switch safely of precisions. This leads to the next algorithm :

Algorithm 5.3 Adaptive precision of iterative rank Randomized SVD

Input: Matrix $A \in \mathbb{R}^{m\times n}$ with $m \geq n$, tolerance ϵ, power q, block size parameter b and a number of maximum iterations its_{max}. Sequences of precisions $u_1 < u_2 < ... < u_p$ and tolerances $\epsilon_1 > ... > \epsilon_p = \epsilon$ are given.

Output: Orthonormal matrix Q and a matrix B such that $\|A - QB\|_F/\|A\|_F \leq \epsilon$. All QR factorizations used are thin QR factorizations.

$Q = [\,], B = [\,], A_1 = A, \rho_1 = 1$
for $i = 1 : its_{max}$ **do**
  Sample $\Omega \in \mathbb{R}^{n\times b}$ with i.i.d. $\mathcal{N}(0, 1)$ entries
  Find smallest j, $1 \leq j \leq p$ such that $\rho_i > \epsilon_j$
  Compute $Y = A_i\Omega$ at precision u_j
  Compute $Q_i = qr(Y, 0)$ at precision u_j
  for $i = 1 : q$ **do**
    Compute $Y = A_i^T Q_i$ and $Q_i = qr(Y, 0)$ at precision u_j
    Compute $Y = A_i Q_i$ and $Q_i = qr(Y, 0)$ at precision u_j
  end for
  Reorthonormalize $Q_i = \mathrm{qr}(Q_i - \sum_{i=1}^{i-1} Q_j Q_j^T Q_i)$ at precision u_1
  Set $Q = [Q, Q_i]$
  Compute $B_i = Q_i^T A_i$ at precision u_j
  Set $B = [B, B_i^T]$
  Compute $A_{i+1} = A_i - Q_i B_i$ at precision u_j
  Compute $\rho_{i+1} = \|A_{i+1}\|_F/\|A\|_F$
  If $\rho_{i+1} \leq \epsilon$ then quit
end for
return Q, B^T

Now, we analyze the effect of rounding error on Algorithm 5.3. On a given iteration i, we compute $\hat{Q}_i$ and $\hat{B}_i = \hat{Q}_i^T A_i$ and we are interested in the error $\|A_i - \hat{Q}_i\hat{B}_i\|_F$. Here, for simplicity, we assume $q > 0$ meaning that we incorporate the power iteration. From either Theorem 5.2 or Theorem 5.3, we know that in the first order of u:

$$\|A_i - \hat{Q}_i\hat{Q}_i^T A_i\|_F \leq \|A_i - Q_i Q_i^T A_i\|_F + uf(m, n, b)\|A_i\|_F,$$

where Q_i is the exact matrix and f depends on whether the bound considered is the worst-case bound or the probabilistic one. As the computation proceeds, A_i is updated and the quantity ρ_i serves as our relative error. For the following, we define:

$$\hat{P}_i = I - \hat{Q}_i\hat{Q}_i^T, P_i = I - Q_i Q_i^T.$$

In exact arithmetic, the relative error after t iterations of the outer loop of Algorithm 5.3 is given by :

$$\rho_{t+1} = \frac{\|P_t...P_1A\|_F}{\|A\|_F}.$$

In floating-point arithmetic, the relative error is :

$$\hat{\rho}_{t+1} = \frac{\|\hat{P}_t...\hat{P}_1A\|_F}{\|A\|_F}.$$

The following theorem summarizes the impact of rounding errors on Algorithm 5.3:

Theorem 5.4. *Let $A \in \mathbb{R}^{m\times n}$ with $m \geq n$ and assume that on line 11 of Algorithm 5.3 is computed exactly. Then, after t steps of the same working precision u, Algorithm 5.3 with $q \geq 1$ produces a floating-point arithmetic relative error such that:*

$$\hat{\rho}_{t+1} \leq \rho_{t+1} + uf(m,n,b)\sum_{i=1}^{t}\hat{\rho}_i \tag{6}$$

Proof. First, we have $\hat{P}_iA_i - P_iA_i = \hat{Q}_i\hat{Q}_i^TA_i - Q_iQ_i^TA_i$ which can be bounded from the proof of Theorem 5.2 . For the sake of simplicity, we bound it with respect to the leading order probabilistic rounding error term, which gives:

$$\hat{P}_iA_i = P_iA_i + \Delta_i, \|\Delta_i\|_F \leq uf(m,n,b)\|A_i\|_F,$$

where, as described before, the form of $f(m,n,b)$ depends on whether we consider the worst case bound or the probabilistic bound. We then have :

$$\begin{aligned}\hat{P}_t...\hat{P}_1A &= \hat{P}_t...\hat{P}_2(P_1A + \Delta_1)\\ &= \hat{P}_t...\hat{P}_3(P_2(P_1A + \Delta_1) + \Delta_2)\\ &= ... = P_t...P_1A + \sum_{i=1}^{t-1}P_t...P_{i+1}\Delta_i + \Delta_t\end{aligned}$$

Taking the frobenius norm of this quantity and dividing by $\|A\|_F$ gives:

$$\begin{aligned}\hat{\rho}_{t+1} &\leq \rho_{t+1} + \frac{\|\sum_{i=1}^{t-1}P_t...P_{i+1}\Delta_i + \Delta_t\|_F}{\|A\|_F} \leq \rho_{t+1} + \sum_{i=1}^{t}\frac{\|\Delta_i\|_F}{\|A\|_F}\\ &\leq \rho_{t+1} + uf(m,n,b)\sum_{i=1}^{t}\frac{\|A_i\|_F}{\|A\|_F} = \rho_{t+1} + uf(m,n,b)\sum_{i=1}^{t}\hat{\rho}_i\end{aligned}$$

□

Now, one of the primary key of this algorithm is to know when to switch to lower precision. The problem setup is the following : we are given a global tolerance ϵ, a sequence of available precisions u_j, an $m\times n$ matrix A and a block size b.
Let us assume that we are at the beggining of the i-th iteration. If we perform t iterations of the power iterations ($q > 0$) at the same precision u_j, then if $u_jf(m,n,b)\sum_{k=i}^{i+t+1}\hat{\rho}_k = \sqrt{mb}u_j\sum_{k=i}^{i+t+1}\hat{\rho}_k$, with $f(m,n,b)$ chosen to be $\sqrt{mb}$ from Subsection 5.2, is significantly less than the tolerance, we know that the contribution of rounding error will be neglictible compared to the algorithmic error. Note that since we do not have access to the $\hat{\rho}_k$'s where $k \geq i+1$, we can bound our quantity of interest by $t\sqrt{mb}u_j\hat{\rho}_i$ because the sequence of $\hat{\rho}_k$ is decreasing. Now the problem is that we don't know t, so we set a user-specified parameter θ which enables the user to insert a degree of optimism or pessimism in the algorithm. The quantity of interest is now $\theta\sqrt{mb}u_j\hat{\rho}_i$. This gives the following choice :

$$\epsilon_j = \begin{cases}\epsilon/(\theta\sqrt{mb}u_{j+1}), & j = 1 : p-1\\ \epsilon, & j = p\end{cases} \tag{7}$$

This implies that if j is the smallest j such that, $\hat{\rho}_i > \epsilon_j$, then $\theta\sqrt{mb}u_j\hat{\rho}_i < \epsilon$, which means that we can safely use u_j depending on the degree of pessimism of θ.

References

[1] A. Asunción and D.J. Newman. UCI Machine Learning Repository. `http://www.ics.uci.edu/~mlearn/MLRepository.html`, 2007.

[2] Erin Carson and Ieva Daužickaitė. Single-pass nyström approximation in mixed precision. 05 2022.

[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. Revised April 2023.

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

[5] Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, 2010.

[6] Nicholas Higham and Srikara Pranesh. Simulating low precision floating-point arithmetic. *SIAM Journal on Scientific Computing*, 41(5), 2019.

[7] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, second edition, 2002.

[8] Nicholas J. Higham and Theo Mary. Mixed precision algorithms in numerical linear algebra. *Acta Numerica*, 31:347–414, 2022.

[9] Per-Gunnar Martinsson and Sergey Voronin. A randomized blocked algorithm for efficiently computing rank-revealing factorizations of matrices, 2015.

[10] Yuji Nakatsukasa. Fast and stable randomized low-rank matrix approximation, 2020.

[11] Joel A. Tropp. An introduction to matrix concentration inequalities, 2015.