Randomized Householder QR

Laura Grigori, Edouard Timsit

Randomized Householder QR

Laura Grigori[*], Edouard Timsit[†]

July 7th, 2023

Abstract: This paper introduces a randomized Householder QR factorization (RHQR). This factorization can be used to obtain a well conditioned basis of a set of vectors and thus can be employed in a variety of applications. The RHQR factorization of the input matrix W is equivalent to the standard Householder QR factorization of matrix ΨW, where Ψ is a sketching matrix that can be obtained from any subspace embedding technique. For this reason, the RHQR algorithm can also be reconstructed from the Householder QR factorization of the sketched problem. In most contexts, left-looking RHQR requires a single synchronization per iteration, with half the computational cost of Householder QR, and a similar cost to Randomized Gram-Schmidt (RGS) overall. We discuss the usage of RHQR factorization in the Arnoldi process and then in GMRES, showing thus how it can be used in Krylov subspace methods to solve systems of linear equations. Based on Charles Sheffield's connection between Householder QR and Modified Gram-Schmidt (MGS), a randomized Modified Gram Schmidt (RMGS) process is also derived.

Numerical experiments show that RHQR produces a well conditioned basis whose sketch is numerically orthogonal, and an accurate factorization. For some very difficult cases, it can be more stable than RGS in both unique precision (half, single, or double) and mixed precision.

1 Introduction

Computing the QR factorization of a matrix $W \in \mathbb{R}^{n\times m}$, $m \ll n$, is a process that lies at the heart of many linear algebra algorithms, for instance for solving least-squares problems or computing bases in Krylov subspace methods. Such process outputs the following factorization:

$$W = QR, \quad Q \in \mathbb{R}^{n\times m}, \quad Q^tQ = I_m, \quad R \in \mathbb{R}^{m\times m}, \quad R \text{ upper triangular.}$$

The QR factorization of $W \in \mathbb{R}^{n\times m}$ is usually obtained using the Gram-Schmidt process or the Householder process. The Householder process relies on *Householder vectors* $u_1, \cdots, u_m \in \mathbb{R}^n$ and orthogonal matrices called *Householder reflectors* $P(u_1), \cdots, P(u_m) \in \mathbb{R}^{n\times n}$ such that:

$$W = P(u_1)\cdots P(u_m)\begin{bmatrix} R \\ 0_{(n-m)\times m}\end{bmatrix} = QR, \quad Q = P(u_1)\cdots P(u_m)\cdot\begin{bmatrix} I_m \\ 0_{(n-m)\times m}\end{bmatrix}, \quad Q^tQ = I_m,$$

where $R \in \mathbb{R}^{m\times m}$ is an upper-triangular factor. It was shown in [15] that the composition of Householder reflectors admits the following factorization:

$$P(u_1)\cdots P(u_m) = I_n - UTU^t, \quad P(u_m)\cdots P(u_1) = I_n - UT^tU^t$$

[*]Laboratory for Simulation and Modelling, Paul Scherrer Institute, Switzerland; Institute of Mathematics, EPFL, Switzerland; part of the work was performed while the author was at Sorbonne Université, Inria, CNRS, Université de Paris, Laboratoire Jacques-Louis Lions, Paris, France.

[†]Sorbonne Université, Inria, CNRS, Université de Paris, Laboratoire Jacques-Louis Lions, Paris, France

where $U = [u_1 \mid \cdots \mid u_m] \in \mathbb{R}^{n\times m}$ is formed by the Householder vectors (and is lower-triangular by design, see Section 2), and $T \in \mathbb{R}^{m\times m}$ is an upper-triangular factor. Applying the Woodburry-Morrison formula to $I_n - UTU^t$, it was outlined in [12] that

$$U^tU = T^{-t} + T^{-1}.$$

J.H. Wilkinson showed in [16] that the Householder QR factorization process is normwise backward stable in finite precision. Later on, it has been shown in [10, Chapter 19.3, Thm 19.4 p360] that the Householder procedure is column-wise backward stable in finite precision. Overall, computations with Householder reflectors are well-known for their excellent numerical stability. An idea from Charles Sheffield and praised by Gene Golub led to the analysis in [11] revealing that the T factor produced by the Householder QR factorization of $[0_{m\times m}; W] \in \mathbb{R}^{(n+m)\times m}$ captures the loss of orthogonality of the Modified Gram-Schmidt procedure on W in finite precision arithmetics. Accordingly, new stable implementations of MGS were derived in [5, 4] (see a comparison of different implementations of Gram-Schmidt procedures in the recent review [6]).

The present work focuses on the use of the ϵ-embedding technique, which has proved to be an effective solution for reducing the communication and computational cost of several classic operations, while providing quasi-optimal results. When solving large scale linear algebra problems on a parallel computer, the communication is the limiting factor preventing scalability to large number of processors. For an m dimensional vector subspace of interest $\mathcal{W} \in \mathbb{R}^n$, some positive real number $\epsilon < 1$, and integer $m < \ell \ll n$, a *sketching matrix* $\Omega \in \mathbb{R}^{\ell\times n}$ is said to be an ϵ-embedding of $\mathcal{W}$ if

$$\forall w \in \mathcal{W}, \quad |\|\Omega w\| - \|w\|| \leq \epsilon\|w\|. \tag{1}$$

If ℓ is a modest multiple of m, there exist simple distributions on $\mathbb{R}^{\ell\times m}$ from which one can draw $\Omega \in \mathbb{R}^{\ell\times m}$ independently of $\mathcal{W}$ and verifying (1) with high probability. Furthermore, some distributions allow the sketching operation $x \mapsto \Omega x$ to be done for a quasi linear cost $O(n\log(n))$ and negligible storage cost of Ω (see [1, 7]). Several methods have been proposed in order to approximate a least squares problem, i.e finding the minimizer of $x \mapsto \|Wx-b\|_2$ using the solution of the cheaper *sketched least squares problem*, i.e finding the minimizer of $x \mapsto \|\Omega(Wx-b)\|_2$. In the pioneering work [13], the authors propose to first compute ΩW, then compute its rank revealing QR factorization to solve the sketched least squares problem. The minimizer and the computed R factor are then used respectively as a starting point and a preconditioner for the conjugate gradient method approximating the initial least squares problem. Later, an alternative randomized Gram-Schmidt (RGS) process was introduced in [3], where each vector of the sketched basis is obtained by sketching the corresponding basis vector right after its orthogonalization step. For a given basis W of a vector subspace of interest and an ϵ-embedding $\Omega \in \mathbb{R}^{\ell\times m}$ of Range(W), the RGS algorithm outputs the following factorization

$$W = QR, \quad Q \in \mathbb{R}^{n\times m}, \quad (\Omega Q)^t\Omega Q = I_m, \quad \mathrm{Cond}(Q) \leq \frac{1+\epsilon}{1-\epsilon}, \quad R \in \mathbb{R}^{m\times m}, \quad R \text{ upper-triangular}$$

for the same communication cost as Classical Gram-Schmidt (CGS), half the asymptotic computational cost, and the stability of MGS (sometimes even better). This *sketch orthogonal basis* Q can then be used as a well conditioned basis of Range(W), also allowing to approximate the least squares problem. It was shown in [3], among other things, that this basis could be used in the Arnoldi iteration and GMRES allowing to obtain a quasi-optimal solution. The authors however identified very difficult cases in which both RGS and MGS experience instabilities in finite precision, with the sketch basis ΩQ losing orthogonality and $\mathrm{Cond}(Q) \approx 10^2$.

In this work we introduce a randomized version of the Householder QR factorization (RHQR). By using a modified sketching matrix Ψ, which can be obtained from any ϵ-embedding matrix Ω, this RHQR factorization relies on *randomized Householder vectors* $u_1, \ldots u_m \in \mathbb{R}^n$ and upper-triangular T factor, and verifies

$$\Psi\Big(I_n - UT(\Psi U)^t\Psi\Big)\cdot\begin{bmatrix}R\\0_{(n-m)\times m}\end{bmatrix} = \Big(I_{\ell+m} - (\Psi U)T(\Psi U)^t\Big)\cdot\begin{bmatrix}R\\0_{\ell\times m}\end{bmatrix}, \quad (\Psi U)^t\Psi U = T^{-t} + T^{-1}, \tag{2}$$

where the left-hand side is the sketch of the RHQR factorization of W, and the right-hand side is the standard Householder QR factorization of the sketch ΨW. This equation shows that the sketch of the basis output by RHQR is orthogonal, i.e

$$Q := \left(I_n - UT(\Psi U)^t \Psi\right) \begin{bmatrix} I_n \\ 0_{(n-m)\times m} \end{bmatrix}, \quad W = QR, \quad (\Psi Q)^t \Psi Q = I_{\ell+m}, \quad \mathrm{Cond}(Q) \leq \frac{1+\epsilon}{1-\epsilon}.$$

We also show that the randomized Householder vectors (and thus the whole RHQR factorization of W) can be reconstructed from the Householder QR factorization of ΨW, which outputs the first m rows of U and matrices $T, \Psi U$ verifying

$$W_{m+1:m,1:m} = U_{m+1:m,1:m} \cdot \mathrm{ut}\left(T^t (\Psi U)^t (\Psi W)\right), \tag{3}$$

where ut denotes the upper-triangular part of the input matrix. Equations (2) and (3) yield three main procedures outlined in this work : *right-looking* and *left-looking* RHQR, and *reconstruct* RHQR. The *right-looking* and *left-looking* RHQR presented in Algorithms 2 and 3 are based on (2). We show that the core iteration of left-looking RHQR makes a single synchronization as long as the next vector of the input basis is available. We also show that the computational cost of its core iteration is dominated by that of two sketches and a matrix vector product, resulting in twice less flops than Householder QR. We observe in our experiments that RHQR is as stable as Householder QR. Furthermore, the core iteration of RHQR relies only on basic linear algebra operations and doesn't need the solving of a sketched least-squares problem as in RGS, making it potentially even cheaper than RGS. On the previously mentioned difficult cases from [3], the left-looking RHQR outputs a basis ΨQ that is numerically orthogonal, with $\mathrm{Cond}(Q) < 2$, while maintaining an accurate factorization. The third process, *reconstruct* RHQR shown in Algorithm 5, is based on (3) and does a single synchronization, like CholeskyQR and its variants. On the previously mentioned difficult cases, reconstructRHQR is more stable than Randomized CholeskyQR and outputs a basis whose condition number is less than 5, while maintaining an accurate factorization. We note that the RHQR factorization can be used in place of the Householder QR factorization in TSQR [9], resulting thus in a communication avoiding algorithm for half the flops of TSQR. Following the methods in [14, Chapter 6.3.2], this RHQR factorization is then embedded in the Arnoldi iteration, allowing to solve systems of linear equations or eigenvalue problems. In this case, RHQR-Arnoldi requires an additional sketch and an additional generalized matrix vector product when compared to RGS-Arnoldi (we recall that Householder-Arnoldi is also more expensive than MGS-Arnoldi). Finally, based on Charles Sheffield's connection between MGS and Householder QR and as it was used in [4] to stabilize MGS, we derive a Randomized Modified Gram-Schmidt (RMGS) in Algorithm 8 verifying the following equations:

$$\begin{bmatrix} 0_{m\times m} \\ W \end{bmatrix} = \overline{Q}R, \quad \overline{Q} = \begin{bmatrix} I_m - T \\ QT \end{bmatrix}, \quad (\Psi\overline{Q})^t \Psi \overline{Q} = I_m, \quad I_m + (\Omega Q)^t \Omega Q = T^{-1} + T^{-t}.$$

On the previously mentioned difficult cases, RMGS outputs a basis whose condition number is less than 20, while maintaining an accurate factorization.

This paper is organized as follows. Section 2 discusses the classical Householder QR factorization and the subspace embedding technique. Section 3 introduces the randomized Householder reflector, and proves that the sketched RHQR factorization of W coincides with the Householder QR factorization of the sketch ΨW (i.e (2)). We then introduce compact formulas that allow to handle this procedure almost in-place and in a left-looking context, as described in our main contribution Algorithm 3 (left-looking RHQR). We also describe a block version in Algorithm 4 (blockRHQR). In Section 4, we adapt the randomized Arnoldi iteration to RHQR in Algorithm 6 (RHQR-Arnoldi), as well as GMRES in Algorithm 7 (RHQR-GMRES), using the same principles as those found in [14]. In Section 5, based on Charles Sheffield's connection between Householder QR and Modified Gram-Schmidt, we introduce a Randomized Modified Gram-Schmidt process (RMGS) in Algorithm 8. In Section 6, we describe

an alternative randomized Householder reflector and the corresponding RHQR factorizations in Algorithms 10 and 11 (left-looking and right-looking trimRHQR). Section 7 describes a set of numerical experiments showing the performance of the algorithms introduced in this paper compared to state of the art algorithms.

2 Preliminaries

In this section, we first introduce our notations. We then outline the original Householder procedure. Finally, we introduce the subspace embedding technique.

2.1 Notations

We denote, for all $n \in \mathbb{N}^*$, 0_n the nul vector of $\mathbb{R}^n$. We denote, for all $n, m \in \mathbb{N}$, $0_{n\times m}$ the nul matrix of $\mathbb{R}^{n\times m}$. The matrix for which we seek to produce a factorization is formed by the vectors $w_1 \dots w_m \in \mathbb{R}^n$, is denoted $W \in \mathbb{R}^{n\times m}$, and is viewed as two blocks:

$$[w_1 \mid \cdots \mid w_m] = W = \begin{bmatrix} W_1 \\ W_2 \end{bmatrix}, \quad W_1 \in \mathbb{R}^{m\times m}, \quad W_2 \in \mathbb{R}^{(n-m)\times m}.$$

The symbol $\oslash$ used between two vectors denotes the concatenation of vectors. If $x \in \mathbb{R}^{n_1}$ and $y \in \mathbb{R}^{n_2}$, then the vector $x \oslash y \in \mathbb{R}^{n_1+n_2}$ is defined as:

$$x \oslash y = \begin{bmatrix} x \\ y \end{bmatrix}.$$

The vertical concatenation of matrices with same column dimension is denoted $C = [A,\ B]$, $A \in \mathbb{R}^{a\times b}$, $B \in \mathbb{R}^{c\times b}$, $C \in \mathbb{R}^{(a+c)\times b}$. The null space of a matrix $A \in \mathbb{R}^{n\times m}$ is denoted

$$\mathrm{Ker}\,(A) = \{x \in \mathbb{R}^m, \quad Ax = 0_n\}.$$

The symbol u_j for $j \in \{1, \dots, m\}$ is dedicated to Householder vectors. The symbol T_j for all $j \in \{1, \dots, m\}$ denotes $j \times j$ matrices that are related to randomized Householder vectors $u_1 \dots u_j$. The symbol U denotes the matrix formed by the vectors $u_1, \dots, u_m$. The symbols Ω and Ψ denote sketching matrices. If the matrix to be factored is $W \in \mathbb{R}^{n\times m}$, then Ω and Ψ are respectively in $\mathbb{R}^{\ell\times(n-m)}$ and $\mathbb{R}^{(\ell+m)\times n}$.

The symbol P denotes several versions of the Householder reflector. When P has one non-zero argument $v \in \mathbb{R}^n$, then it denotes the standard Householder reflector of $\mathbb{R}^n$,

$$P(v) = I_n - \frac{2}{\|v\|^2} vv^t.$$

When P has two arguments $v \in \mathbb{R}^n$ and $\Theta \in \mathbb{R}^{\ell\times n}$, $v \notin \mathrm{Ker}\,(\Theta)$, then it denotes the randomized Householder reflector,

$$P(v, \Theta) = I_n - \frac{2}{\|\Theta v\|^2} \cdot v(\Theta v)^t \Theta,$$

which is described in Section 3.

Given any matrix $A \in \mathbb{R}^{n\times m}$, we denote respectively $\mathrm{ut}(A)$, $\mathrm{sut}(A)$, $\mathrm{lt}(A)$ and $\mathrm{slt}(A)$ the upper triangular, strictly upper triangular, lower triangular, strictly lower triangular parts of A, respectively. We denote $A_{i,j}$ the entry of A located on the i-th row and the j-th column. We denote $A_{j_1:j_2}$ the matrix formed by the $j_1, j_1+1, \dots j_2$-th columns of A. We denote $a_1, \dots a_m$ the column vectors of A. In the algorithms, we denote the entries i to j of a vector w with the symbol $(w)_{i:j}$.

The canonical vectors of $\mathbb{R}^n$ are denoted $e_1, \dots, e_m$. For some $k \in \{1, \dots, n\}$ and $j \in \{1, \dots, k\}$, we denote e_j^k the j-th canonical vector of $\mathbb{R}^k$.

The letter ϵ denotes a real number of $]0, 1[$, which is a parameter that determines the dimensions of Ω and thus Ψ.

2.2 Householder orthonormalization

The Householder QR factorization is an orthogonalization procedure, alternative to the Gram Schmidt process. While the Gram-Schmidt process focuses on building explicitly the orthogonal factor Q, Householder's procedure focuses instead on building the factor R using orthogonal transformations. Q and Q^t are stored in a factored form. The central operator of this procedure is the Householder reflector:

$$\forall z \in \mathbb{R}^n \setminus \{0\}, \quad P(z) = I_n - \frac{2}{\|z\|^2} z\, z^t, \tag{4}$$

or equivalently

$$\forall z \in \mathbb{R}^n \setminus \{0\},\ \forall x \in \mathbb{R}^n, \quad P(z) \cdot x = x - \frac{2}{\|z\|^2} \langle z, x \rangle\, z.$$

$P(z)^2 = I_n$, hence its name, and also $P(z)^t = P(z)$. These two properties combined show that $P(z)$ is an orthogonal matrix. The vector z is an eigenvector associated to eigenvalue -1, and the orthogonal of its span is an $n-1$ dimensional eigenspace associated to eigenvalue 1, i.e the Householder reflector is an orthogonal reflector with respect to the latter hyperplane. Given a vector $w \in \mathbb{R}^n$ that is not a multiple of e_1, and setting $z = w - \|w\| e_1$, one can straightforwardly derive,

$$P(z) \cdot w = \|w\| e_1,$$

hence the Householder reflector can be used to annihilate all the coordinates of a vector, except its first entry (it has been shown that the formula for z can be refined into stabler formulas in finite precision, see [10, Chapter 19.1]). It can be inferred by induction that, given a vector $w = c \oslash d \in \mathbb{R}^n$, $c \in \mathbb{R}^{j-1}$, $d \in \mathbb{R}^{n-j+1}$, there exists $v_j \in R^{n-j+1}$, $u_j = 0_{j-1} \oslash v_j \in \mathbb{R}^n$ such that

$$P(u_j) \cdot w = \begin{bmatrix} I_{j-1} & \\ & P(v_j) \end{bmatrix} \cdot w = \begin{bmatrix} c \\ \|d\| e_1^{n-j+1} \end{bmatrix} = \begin{bmatrix} c \\ \|d\| \\ 0_{n-j} \end{bmatrix},$$

that is the reflector $P(u_j)$ now annihilates the coordinates $j+1$ to n of the vector w, without modifying its first $j-1$ entries. By induction (this classic argument is detailed below in the proof of Theorem 3.5), there exist *Householder vectors* $u_1, \cdots u_m$ such that

$$P(u_m) \cdots P(u_1) W = \begin{bmatrix} R \\ 0_{(n-m)\times m} \end{bmatrix} \iff W = P(u_1) \cdots P(u_m) \begin{bmatrix} R \\ 0_{(n-m)\times m} \end{bmatrix}. \tag{5}$$

It was outlined in [15] that the composition $P(u_1) \cdots P(u_m)$ can be factored as:

$$\begin{cases} P(u_1) \cdots P(u_m) = I_n - UTU^t \\ P(u_m) \cdots P(u_1) = I_n - UT^tU^t \end{cases} \tag{6}$$

where $U \in \mathbb{R}^{n\times m}$ is formed by $u_1, \ldots, u_m$ and $T \in \mathbb{R}^{m\times m}$ is an upper-triangular matrix. If we denote $\beta_j = 2/\|u_j\|^2$ for all $j \in \{1, \ldots, m\}$, then T defined by induction as:

$$T_1 = [\beta_1], \quad \forall j \in \{1, \ldots, m-1\}, \quad T_{j+1} = \begin{bmatrix} T_j & -\beta_{j+1} T_j U_j^t u_{j+1} \\ 0_{1\times j} & \beta_{j+1} \end{bmatrix}, \quad T := T_m$$

As a triangular matrix with a diagonal of non-zeros, T is non-singular. As outlined in [12], we can then apply the Woodburry Morrison formula to $I_n - UTU^t$ and derive

$$U^tU = T^{-1} + T^{-t}, \quad T = \left(\mathrm{sut}(U^tU) + \frac{1}{2}\mathrm{Diag}(U^tU)\right)^{-1}$$

where sut denotes the strictly upper-triangular part and Diag denotes the matrix formed by the diagonal entries. We insist that both the explicit compositions of Householder reflectors and formulas (6) are operators of $\mathbb{R}^n$, hence implicitly represented by $\mathbb{R}^{n\times n}$ matrices. The right-hand sides of (6) are referred to as *compact forms* of the compositions $P_1 \cdot P_2 \cdots P_m \in \mathbb{R}^{n\times n}$ and $P_m \cdot P_{m-1} \cdots P_1 \in \mathbb{R}^{n\times n}$.

Whereas the Gram-Schmidt type algorithms produces the factors $Q \in \mathbb{R}^{n\times m}$ and an upper-triangular $R \in \mathbb{R}^{m\times m}$ such that $W = QR$, the Householder process outputs instead an isometric operator $P(u_1)\cdots P(u_m) \in \mathbb{R}^{n\times n}$, and an upper-triangular factor $R \in \mathbb{R}^{m\times m}$ such that $P(u_1)\cdots P(u_m) \cdot \left[R;\ 0_{(n-m)\times m}\right]$. A matrix $Q \in \mathbb{R}^{n\times m}$ can still be output by Householder QR using the compact formulas (6),

$$W = P(u_1)\cdots P(u_m)\begin{bmatrix} R \\ 0_{(n-m)\times m}\end{bmatrix} \iff W = \left(\begin{bmatrix} I_m \\ 0_{(n-m)\times m}\end{bmatrix} - UTU^t\begin{bmatrix} I_m \\ 0_{(n-m)\times m}\end{bmatrix}\right)\cdot R =: QR,$$

We refer to $Q \in \mathbb{R}^{n\times m}$ as the *thin Q factor* associated to U, T. We stress that U, T and (6) are sufficient for the computation of $Q^t x, x \in \mathbb{R}^n$ for solving the least-squares problem. If W admits the factorization in (5), then

$$\arg\min_{x\in\mathbb{R}^m} \|Wx - b\| = R^{-1}\cdot\left[I_m\ \ 0_{m\times(n-m)}\right]\cdot\Big(P(u_m)\cdots P(u_1)\Big)\cdot b = R^{-1}\cdot\left[I_m\ 0_{m\times(n-m)}\right]\cdot(b - UT^tU^tb)$$

that is the application of $(P(u_1)\cdots P(u_m))^{-1} = P(u_m)\cdots P(u_1)$, followed by the sampling of the first m coordinates, followed by the backward solve of R. In this work, we denote the pair (U, T) in (6) as the *implicit* Q factor.

As described in [11, 4], Charles Sheffield pointed out to Gene Golub that given a matrix $W \in \mathbb{R}^{n\times m}$, its modified Gram-Schmidt (MGS) QR factorization was equivalent to the Householder QR factorization of $[0_{m\times m};\ \ W]$ in both exact and finite precision arithmetics. The analysis derived in [11] shows that, in this context of finite precision, the factor T computed by Householder QR captures the loss of orthogonality in Modified Gram-Schmidt. Accordingly, a stabler version of MGS was derived in [5, 4].

All these principles and formulas find a straightforward randomized equivalent in the randomized Householder QR factorization (RHQR) that we derive in Section 3.

2.3 Subspace embeddings

Definition 2.1. [17, Definition 1] We say that a matrix $\Omega \in \mathbb{R}^{\ell\times n}$ is an ϵ-embedding of a vector-subspace $\mathcal{W} \subset \mathbb{R}^n$ if and only if

$$\forall x \in \mathcal{W},\ \ (1-\epsilon)\|x\| \le \|\Omega x\| \le (1+\epsilon)\|x\|. \tag{7}$$

The ϵ-embedding property is sometimes written as in (7), only with squared norms. The use of ϵ-embedding matrices is obviously justified by one or other of these properties. For any m-dimensional vector subspace $\mathcal{W} \subset \mathbb{R}^n$, there exist simple distributions over $\mathbb{R}^{\ell\times n}$ whose realizations $\Omega \in \mathbb{R}^{\ell\times n}$ drawn independently of $\mathcal{W}$ are an ϵ-embedding of $\mathcal{W}$ with high probability. These distributions are called oblivious subspace embeddings (OSE).

Definition 2.2. [17, Definition 2] Let $\epsilon, \delta \in]0, 1[$. Let $m \le \ell \ll n \in \mathbb{N}^*$. We say that a distribution $\mathcal{D}$ over $\mathbb{R}^{\ell\times n}$ is an oblivious subspace embedding with parameters (ϵ, δ, m), denoted $\mathrm{OSE}(\epsilon, \delta, m)$, if and only if for any m-dimensional vector subspace $\mathcal{W}_m \subset \mathbb{R}^n$, Ω drawn from $\mathcal{D}$ independently of $\mathcal{W}_m$ is an ϵ-embedding of $\mathcal{W}_m$ with probability at least $1 - \delta$.

As an example, we mention three such distributions. The Gaussian OSE is the simplest, and consists in drawing a matrix $G \in \mathbb{R}^{\ell\times n}$ where the coefficients are i.i.d standard Gaussian variables, and set

$$\Omega = \frac{1}{\sqrt{\ell}}\,G.$$

As proven in [17], this distribution is an (ϵ, δ, m) OSE if $\ell = \mathcal{O}\left(\epsilon^{-2}(m + \log(1/\delta))\right)$. If this technique has the advantage of giving a simple way of building ϵ-embedding matrices, it has the disadvantage of relying on a dense matrix Ω, whose storage and application to a vector $x \in \mathbb{R}^n$ are expensive (it still has the advantage of scaling well on a parallel computer). A lighter and faster technique is given by the subsampled randomized Hadamard transform (SRHT) OSE, which writes

$$\Omega = \sqrt{\frac{n}{\ell}} PHD,$$

where P is made of ℓ rows of the identity matrix I_n drawn uniformly at random (which corresponds to a uniform sampling step), H is the normalized Walsh-Hadamard transform, and D is a diagonal of random signs (see [1]). In most applications, this matrix is not explicitly stored, and is only given as a matrix-vector routine. In this case, its storage cost is negligible. Its application to a vector costs $\mathcal{O}(n \log n)$ flops when implemented with standard Walsh-Hadamard Transform. This distribution is OSE(ϵ, δ, m) if $\ell \in \mathcal{O}(\epsilon^{-2}(m + \log(n/\delta)) \log(m/\delta))$. Finally, we mention distributions based on hashing techniques. In this case, the obtained sketching matrix Ω is a sparse matrix with a specified number of expected non-zeros per columns, see [7].

Let us consider a set of linearly independent vectors $w_1, \dots w_m$ forming a full-rank matrix $W \in \mathbb{R}^{n \times m}$ and spanning $\mathcal{W} = \text{Range}(W)$. The largest and smallest *singular values* of W are defined as

$$\sigma_{\max}(W) = \|W\|_2 = \max_{\|x\|_2=1} \|Wx\|_2 = \|Wv_{\max}\|_2, \quad \sigma_{\min}(W) = \min_{\|x\|_2=1} \|Wx\|_2 = \|Wv_{\min}\|_2, \quad v_{\max}, v_{\min} \in \mathbb{R}^m$$

i.e the greatest dilatation and the greatest contraction of an input vector $x \in \mathbb{R}^m$. Their ratio also defines the condition number of W, as shown by the min-max theorem.

$$\text{Cond}(W) = \sqrt{\frac{\lambda_{\max}(W^t W)}{\lambda_{\min}(W^t W)}} = \frac{\sigma_{\max}(W)}{\sigma_{\min}(W)} = \frac{\|Wv_{\max}\|}{\|Wv_{\min}\|}, \quad v_{\max}, v_{\min} \in \mathbb{R}^m.$$

If $\Omega \in \mathbb{R}^{\ell \times n}$ is an ϵ-embedding for Range(W), then (see [17, 3])

$$\text{Cond}(W) \leq \frac{1+\epsilon}{1-\epsilon} \text{Cond}(\Omega W).$$

In particular, if $Q \in \mathbb{R}^{n \times m}$ is a basis of $\mathcal{W}$ such that $\Omega Q \in \mathbb{R}^{\ell \times m}$ has orthonormal columns, the condition number of Q is simply bounded by the constant $(1+\epsilon)/(1-\epsilon)$. With $\epsilon = 1/2$, we get $\text{Cond}(Q) \leq 3$. Hence sketch-orthonormal bases are well-conditioned bases.

3 Randomized Householder process

In this section we introduce a randomized version of the Householder reflector and show its main properties. We then show how several randomized Householder reflectors can be used to factor $W = QR$ such that $\Psi W = (\Psi Q) \cdot R$ is the Householder factorization of ΨW (Theorem 3.5). These computations yield the right-looking version of the RHQR factorization in Algorithm 2. We then derive a compact representation of the composition of multiple randomized Householder reflectors, yielding our main contribution, the left-looking version of the RHQR factorization in Algorithm 3. Exploiting the equivalence between the Householder factorizations, we derive an algorithm reconstructing the RHQR factorization of W from the Householder QR factorization of ΨW in Algorithm 5. Finally, we show a block version of RHQR in Algorithm 4.

3.1 Algebra of the randomized Householder QR

Let us set a matrix $\Theta \in \mathbb{R}^{\ell \times n}$, a vector $z \in \mathbb{R}^n \setminus \mathrm{Ker}\,(\Theta)$ and define the *randomized Householder reflector associated to* Θ *and* z:

$$\forall \Theta \in \mathbb{R}^{\ell \times n}, \quad \forall z \in \mathbb{R}^n \setminus \mathrm{Ker}\,(\Theta)\,, \quad P(z,\Theta) = I_n - \frac{2}{\|\Theta z\|^2} \cdot z(\Theta z)^t \Theta \tag{8}$$

Proposition 3.1. *Let* $P(z,\Theta)$ *be defined as in* (8)*. The following properties hold:*

1. $\forall \lambda \in \mathbb{R}^*, \quad P(\lambda z, \Theta) = P(z,\Theta)$
2. $P(z,\Theta)^2 = I_n$*, i.e* $P(z,\Theta)$ *is a (non orthogonal) reflector*
3. $\forall x \in \mathbb{R}^n, \quad \Theta \cdot P(z,\Theta) \cdot x = P(\Theta z) \cdot \Theta x$ *where* $P(\Theta z)$ *denotes the standard Householder reflector of* $\mathbb{R}^\ell$ *associated to* $\Theta z \in \mathbb{R}^\ell$*.*
4. $\forall x \in \mathbb{R}^n, \quad \|\Theta \cdot P(z,\Theta) \cdot x\|_2 = \|\Theta x\|_2$*, (consequence of 3) We say that* $P(z,\Theta)$ *is* ***sketch-isometric*** *with respect to* Θ*.*

Proof. Let us denote $P := P(z,\Theta)$. First, for $\lambda \in \mathbb{R}^*$, we obtain:

$$I_n - \frac{2}{\lambda^2 \|\Theta z\|^2} \cdot (\lambda z) \cdot (\lambda \Theta z)^t \Theta = I_n - \frac{2}{\|\Theta z\|^2} \cdot \frac{\lambda^2}{\lambda^2} \cdot z \cdot (\Theta z)^t \Theta = P.$$

For the next two points, we suppose that $\|\Theta z\|^2 = 2$. Now, for $x \in \mathbb{R}^n$,

$$Px = x - \langle \Theta x, \Theta z \rangle z.$$

Remark also that $Pz = -z$. Then,

$$PPx = Px + \langle \Theta x, \Theta z \rangle z = x - \langle \Theta x, \Theta z \rangle z + \langle \Theta x, \Theta z \rangle z = x.$$

Lastly, developping the squared norm,

$$\|\Theta Px\|^2 = \|\Theta x\|^2 - 2\langle \Theta x, \Theta z \rangle^2 + \langle \Theta x, \Theta z \rangle^2 \|\Theta z\|^2 = \|\Theta x\|^2 - 2\langle \Theta x, \Theta z \rangle^2 + 2\langle \Theta x, \Theta z \rangle^2 = \|\Theta x\|^2.$$

For the third property, by definition of (8), for all $x \in \mathbb{R}^n$, we get

$$\Theta \cdot P \cdot x = \Theta x - \frac{2}{\|\Theta z\|^2} \Theta z (\Theta z)^t \Theta x = \left(I_\ell - \frac{2}{\|\Theta z\|^2} \Theta z (\Theta z)^t \right) \cdot \Theta x = P(\Theta z) \cdot \Theta x$$

where $P(\Theta z)$ is the Householder reflector of $\mathbb{R}^\ell$ associated to $\Theta z \in \mathbb{R}^\ell$ defined in (4). □

Remark 3.2. The sketch-isometric property of $P = P(z,\Theta)$ is equivalent to

$$P^t \Theta^t \Theta P = \Theta^t \Theta,$$

while $P^t \Theta^t \Theta P$ is at most of rank ℓ.

Compositions of multiple randomized Householder reflectors can be represented with compact formulas.

Proposition 3.3. *Let $u_1, \cdots u_m \in \mathbb{R}^n$. Define for all $j \in \{1, \dots, m\}$ the coefficients $\beta_j = 2/\|\Theta u_j\|^2$. Define by induction the matrix T_j for all $1 \leq j \leq m$ as*

$$T_1 \in \mathbb{R}^{1\times 1}, \;\; T_1 = [\beta_1]$$
$$\forall 1 \leq j \leq m-1, \;\; T_{j+1} = \begin{bmatrix} T_j & -\beta_j \cdot T_j(\Theta U_j)^t \Theta u_{j+1} \\ 0_{1\times j} & \beta_j \end{bmatrix}$$

Then, denoting $U \in \mathbb{R}^{n\times m}$ the matrix formed by $u_1, \dots, u_m$ and $T = T_m$, we get the following compact representation:

$$\begin{aligned} P(u_1, \Theta) \cdots P(u_m, \Theta) &= I_n - UT(\Theta U)^t \Theta \\ P(u_m, \Theta) \cdots P(u_1, \Theta) &= I_n - UT^t(\Theta U)^t \Theta \end{aligned} \tag{9}$$

Furthermore, the factor T verifies

$$(\Theta U)^t \Theta U = T^{-1} + T^{-t} \tag{10}$$

Proof. The compact formulas can be derived by straightforward induction as in the deterministic case shown in [15]. Then, apply the Woodburry-Morrison formula to the left-wise and right-wise compositions as in [12]. □

Let us set a matrix $\Omega \in \mathbb{R}^{\ell\times(n-m)}$, and define the matrix

$$\Psi = \left[\begin{array}{c|c} I_m & \\ \hline & \Omega \end{array}\right] \in \mathbb{R}^{(\ell+m)\times n}. \tag{11}$$

Then the randomized Householder reflector associated to z and Ψ is

$$\forall z \in \mathbb{R}^n \setminus \mathrm{Ker}\,(\Psi), \quad P(z, \Psi) = I_n - \frac{2}{\|\Psi z\|^2} \cdot z \cdot (\Psi z)^t \Psi \in \mathbb{R}^{n\times n} \tag{12}$$

All properties from Propositions 3.1 and 3.3 apply to $P(z, \Psi)$ and to multiple compositions of those reflectors, simply replacing Θ by Ψ. We now show that the design of Ψ allows to annihilate entries of a vector below a given index.

Proposition 3.4. *Let $j \in \{1, \dots, m\}$. Denote $c \in \mathbb{R}^{j-1}$ and $d \in \mathbb{R}^{n-j+1}$ such that $w = c \oslash d$. Define $w' = 0_{j-1} \oslash d$, and suppose that w' is neither in the kernel of Ψ nor a multiple of e_j^n. Define the randomized Householder vector as:*

$$u_j = w' - \|\Psi w'\| e_j \in \mathbb{R}^n. \tag{13}$$

Then

$$P(u_j, \Psi) \cdot w = c \oslash (\|\Psi w'\| e_1^{n-j+1}) = \begin{bmatrix} c \\ \|\Psi w'\| \\ 0_{n-j} \end{bmatrix}.$$

Proof. First, remark that $\langle \Psi u_j, \Psi w \rangle = \langle \Psi u_j, \Psi w' \rangle$ because of u_j's first $j-1$ zero entries coupled with the design of Ψ. Also by design of Ψ, remark that $\langle \Psi w, \Psi e_j \rangle = \langle \Psi w', \Psi e_j \rangle$. Developing the expression of u_j, we get:

$$\langle \Psi w, \Psi u_j \rangle = \|\Psi w'\|^2 - \|\Psi w'\| \langle \Psi w', \Psi e_j \rangle.$$

On the other hand, recalling that $\|\Psi e_j\| = 1$ (see (11)),

$$\|\Psi u_j\|^2 = 2\|\Psi w'\|^2 - 2\|\Psi w'\| \langle \Psi w', \Psi e_j \rangle,$$

hence

$$\frac{2}{\|\Psi u_j\|^2} \cdot \langle \Psi w, \Psi u_j \rangle = 1,$$

which finally yields

$$P(u_j, \Psi) \cdot w = w - u_j = c \oslash \left(d - d + \|\Psi w'\| e_1 \right) = c \oslash \left(\|\Psi w'\| e_1 \right).$$

□

Let us quickly point out that the formula in Proposition 3.4 suffers the same cancellation problems as in the deterministic case in finite precision arithmetics. Indeed, let us suppose that w is almost a positive multiple of e_1, that is $w_1 = \lambda e_1 + g \in \mathbb{R}^n$, where g is a vector of small norm, and $\lambda \in \mathbb{R}$. The vector u_1 as defined in Proposition 3.4 might suffer cancellations. It is then useful to flip the sign of u_1, and choose in general

$$u_j := w' + \operatorname{sign}\langle w_j, e_j \rangle \|\Psi w'\| e_j^n,$$

with w' from Proposition 3.4. To sum up, denoting $c \in \mathbb{R}^{j-1}$ and $d \in \mathbb{R}^{n-j+1}$ such that $w = c \oslash d$,

$$\begin{cases} \sigma_j = \operatorname{sign}\langle w, e_j \rangle \\ \rho_j = \|\Psi(0_{j-1} \oslash d)\| \\ u_j = (0_{j-1} \oslash d) + \sigma_j \rho_j e_j^n \end{cases} \implies P(u_j, \Psi) \cdot w = \begin{bmatrix} c \\ -\sigma_j \rho_j \\ 0_{n-j} \end{bmatrix} \in \mathbb{R}^n \tag{14}$$

We can now state our main result.

Theorem 3.5. *(simultaneous factorizations) Let $W \in \mathbb{R}^{n \times m}$, $W = [W_1; W_2]$, $W_1 \in \mathbb{R}^{m \times m}$, and assume that W_2 is full-rank. Let $\Omega \in \mathbb{R}^{\ell \times (n-m)}$ such that ΩW_2 is also full-rank. Define Ψ as in* (11)*. Then there exist randomized Householder vectors $u_1, \ldots, u_m \in \mathbb{R}^n$ such that*

$$\begin{cases} W = P(u_1, \Psi) \cdots P(u_m, \Psi) \cdot \begin{bmatrix} R \\ 0_{(n-m) \times m} \end{bmatrix} = \left(I_n - UT(\Psi U)^t \Psi \right) \begin{bmatrix} R \\ 0_{(n-m) \times m} \end{bmatrix} = QR \ \textit{(RHQR)} \\ \Psi W = P(\Psi u_1) \cdots P(\Psi u_m) \cdot \begin{bmatrix} R \\ 0_{\ell \times m} \end{bmatrix} = \left(I_{\ell+m} - \Psi UT(\Psi U)^t \right) \begin{bmatrix} R \\ 0_{\ell \times m} \end{bmatrix} = (\Psi Q) R \ \textit{(Householder QR)} \end{cases} \tag{15}$$

i.e the sketch of the RHQR factorization of W is the Householder QR factorization of the sketch ΨW. In particular, $(\Psi Q)^t \Psi Q = I_m$. If Ω is an ϵ-embedding of $\operatorname{Range}(W_2)$*, then Ψ is an ϵ-embedding of* $\operatorname{Range}(W)$*, and we get*

$$\operatorname{Cond}(Q) \leq \frac{1+\epsilon}{1-\epsilon}$$

Proof. Let us first compute u_1 as in Proposition 3.4 such that the first column of W gets all its entries below the first one cancelled. Since W_2 and ΩW_2 are full-rank, u_1 is well-defined and is not in the kernel of Ψ. We get

$$P(u_1, \Psi) \cdot W = \left[\begin{array}{c|ccc} r_{1,1} & r_{1,2} & \cdots & r_{1,m} \\ \hline 0 & & & \\ \vdots & & & \\ \vdots & & * & \\ \vdots & & & \\ 0 & & & \end{array}\right] \in \mathbb{R}^{n \times m}$$

Let us now denote w the second column of $P(u_1, \Psi) \cdot W$, and compute u_2 such that all the entries of w below the second one are cancelled. The span of the last $n - m$ rows of $P(u_1, \Psi)$ is still that of W_2.

Since W_2 and ΩW_2 are full rank, u_2 is well-defined and is not in the kernel of Ψ. Furthermore, as shown in Proposition 3.4, the reflector $P(u_2, \Psi)$ does not modify the first row of the input matrix. We get

$$P(u_2,\Psi)P(u_1,\Psi)\cdot W = \left[\begin{array}{cc|ccc} r_{1,1} & r_{1,2} & r_{1,3} & \cdots & r_{1,m} \\ 0 & r_{2,2} & r_{2,3} & \cdots & r_{2,m} \\ \hline 0 & 0 & & & \\ \vdots & \vdots & & & \\ \vdots & \vdots & & * & \\ \vdots & \vdots & & & \\ 0 & 0 & & & \end{array}\right] \in \mathbb{R}^{n\times m}$$

Iterating the argument and denoting $P_j = P(u_j, \Psi)$ for all $j \in \{1, \ldots, m\}$, we get

$$P_m P_{m-1} \cdots P_1 W = \begin{bmatrix} R \\ 0_{(n-m)\times m} \end{bmatrix} \tag{16}$$

where $R \in \mathbb{R}^{m\times m}$ is upper-triangular by construction. Recalling that $P_j^2 = I_n$ for all $j \in \{1, \ldots, m\}$, we deduce

$$W = P_1 P_2 \cdots P_m \begin{bmatrix} R \\ 0_{(n-m)\times m} \end{bmatrix} \tag{17}$$

If we sketch the above equation, we get

$$\Psi W = \Psi P_1 P_2 \cdots P_m \begin{bmatrix} R \\ 0_{(n-m)\times m} \end{bmatrix}$$

Using the third point of Proposition 3.1, we get

$$\Psi W = P(\Psi u_1)\cdots P(\Psi u_m)\cdot \Psi \begin{bmatrix} R \\ 0_{(n-m)\times m} \end{bmatrix} = P(\Psi u_1)\cdots P(\Psi u_m)\cdot \begin{bmatrix} R \\ 0_{\ell\times m} \end{bmatrix}$$

where $P(\Psi u_1), \ldots P(\Psi u_m)$ are the standard Householder reflectors of $\mathbb{R}^{ell+m}$ associated with vectors $\Psi u_1 \ldots \Psi u_m \in \mathbb{R}^{\ell+m}$, which shows the equivalence of the two factorizations. The thin Q factor of the RHQR factorization is

$$Q = P_1 \cdots P_m \begin{bmatrix} I_m \\ 0_{(n-m)\times m} \end{bmatrix}$$

hence

$$\Psi Q = P(\Psi u_1)\cdots P(\Psi u_m)\cdot \begin{bmatrix} I_m \\ 0_{\ell\times m} \end{bmatrix}$$

which is the thin Q factor of the Householder QR factorization, hence it is orthogonal, hence Q is sketch-orthogonal.

Finally, the ϵ-embedding property of Ψ follows from the Pythagorea's theorem and the ϵ-embedding property of Ω. □

3.2 Algorithms

We detail the randomized Householder vector computation in Algorithm 1 (RHVector). We stress that in Algorithm 1 and in a distributed environment, supposing that sketches are available to all processors after synchronization, then by design of Ψ the scalings in lines 11 and 12 do not require further synchronization. The scaling in line 12 is more precise than sketching again.

Algorithm 1: Computation of randomized Householder vector

Input: $d \in \mathbb{R}^{n-j+1}$ where $j \in \{1, \dots, m\}$, $\Omega \in \mathbb{R}^{(\ell-m)\times n}$

Output: $v, v', \sigma, \rho, \beta$ such that $0_{j-1} \oslash v' = \Psi(0_{j-1} \oslash v)$ and $P(0_{j-1} \oslash v, \Psi) \cdot (c \oslash d) = c \oslash (-\sigma\rho e_j^n)$ for all $c \in \mathbb{R}^{j-1}$, with Ψ defined in (11)

1 **function** `RHVector(`y, Ω`):`
2 $v \leftarrow d$
3 $a \leftarrow$ first $m - j + 1$ coordinates of v
4 $b \leftarrow$ last $n - m$ coordinates of v
5 Sketch Ωb
6 $v' \leftarrow a \oslash \Omega b$
7 $\sigma \leftarrow$ sign of first entry of v'
8 $\rho \leftarrow \|v'\|$
9 Add $\sigma\rho$ to the first entry of v and to the first entry of v'
10 Choose $\alpha =$ first entry of v' or $\alpha = \|v'\|/\sqrt{2}$
11 $v \leftarrow v/\alpha$
12 $v' \leftarrow v'/\alpha$
13 $\beta \leftarrow 2/\|v'\|^2$ # for second choice of α, just set $\beta = 1$
14 **return** $v, v', \sigma, \rho, \beta$

We detail the process from the proof of Theorem 3.5 in its simplest form in Algorithm 2 (right-looking RHQR). One sketching is performed in the RHVector routine called in line 5, which induces one synchronization. During this synchronization, it is possible to perform the sketching in line 10. This algorithm can be performed almost in place, since U is lower-triangular by construction, and R is upper-triangular by construction. Both can almost fit in the original W, while the diagonal of either R or U can be stored in some additional space. Then, a small additional space is required for ΨU (S in the algorithm). We stress that, in a distributed environment where the sketches are available to all processors after synchronization, then all processors have access to the $m \times m$ upper-block of W, R, U by design of Ψ. This version of RHQR may be the simplest conceptually, it is clearly not optimal in terms of flops, as it requires to sketch $m - j + 1$ vectors at step j. This issue is addressed by the left-looking RHQR factorization.

In the light of the compact formulas (9) for the composition of multiple randomized Householder reflectors, we introduce the main algorithm for computing the RHQR factorization in Algorithm 3. Only the j-th column of W is modified at iteration j. At the beginning of this iteration, the compact form of the previously computed reflectors is used to apply them to the j-th column of W. Then the randomized Householder reflector of the resulting column is computed, and the compact factorization is updated. After refreshing the vector w in line 7, one synchronization is made at line 8. If the next vector w_{j+1} is already available (e.g QR factorization, s-step and block Krylov methods), its sketch can be computed in the same synchronization, hence avoiding a further synchronization in line 5. This algorithm can be executed almost in place, with the same recommendations as for the right-looking version, and a minor additional storage space for the matrix T. The sole high-dimensional operations in Algorithm 3 are the two sketches in Lines 5 and 8, and the update of the j-th column in Line 7. The cost of the update is dominated by that of $U_{j-1} \cdot (T_{j-1}^t S_{j-1}^t z)$, which is $2nj$ flops, for a final cost of $2nm^2$ flops. Unlike the Householder QR factorization, updating the T factor has a negligible cost. The two sketches add $2n\log(n)$ flops. The cost of the whole factorization is then essentially $2nm^2 + 2n\log(n)$. This shows that the left-looking RHQR requires twice less flops than Householder QR, essentially the cost of an LU factorization of W or that of RGS. We note that, similarly to RGS, mixed precision can be used.

Algorithm 2: Randomized-Householder QR factorization (right-looking)

Input: Matrix $W \in \mathbb{R}^{n\times m}$, matrix $\Omega \in \mathbb{R}^{\ell\times(n-m)}$, $m < l \ll n-m$
Output: R, U, S, such that $S = \Psi U$ and (15) holds

1 **function** `RHQR_right(`W, Ω`):`
2 **for** $j = 1 : m$ **do**
3 $r_j \leftarrow (w_j)_{1:j-1}$
4 $w \leftarrow w_j$
5 $v, v', \rho, \sigma, \beta =$ `RHVector(`$(w)_{j:n}$, Ω`)`
6 $u_j \leftarrow 0_{j-1} \oslash v$
7 $s_j \leftarrow 0_{j-1} \oslash v'$
8 $r_j \leftarrow r_j \oslash (-\sigma\rho) \oslash 0_{m-j}$
9 **if** $j < m$ **then**
10 Sketch $\Omega W_{m+1:n,j+1:m}$
11 $X \leftarrow \left[W_{1:m,j+1:m};\ \Omega W_{m+1:n,j+1:m}\right]$ # i.e $X \leftarrow \Psi W$
12 $W_{j+1:m} \leftarrow W_{j+1:m} - \beta \cdot u_j \cdot s_j^t X$
13 **return** $R = (r_j)_{j\le m}$, $U = (u_j)_{j\le m}$, $S = (s_j)_{j\le m}$

For example, one could use half or single precision for storing and computing with high-dimensional matrices and for the sketching operation, while performing the low-dimensional operations in double precision, see experiments in Section 7.

If the thin Q factor is needed, one needs to perform the following operation:

$$Q = \begin{bmatrix} I_m \\ 0_{(n-m)\times m} \end{bmatrix} - UTU_{1:m,1:m}^t.$$

The cost is dominated by that of the matrix-matrix multiply $U_m \cdot (T_m U_{1:m,1:m}^t)$, which is $2nm^2$.

To conclude this section, we detail in Algorithm 4 a block version of Algorithm 3. The matrix $W \in \mathbb{R}^{n\times(mb)}$ is now considered as formed by vertical slices $W^{(1)}, \cdots W^{(b)} \in \mathbb{R}^{n\times b}$. Algorithm 3 is then applied consecutively to each block. For each $W^{(j)}$, we denote $U_m^{(j)}, \Psi U_m^{(j)}, T_m^{(j)}$ the output of Algorithm 3 applied to $W^{(j)}$. As to the storage space and the communications, we refer to the comments made for Algorithm 3. The update of columns $(j-1)b+1, \ldots, jb$ of W is presented as such in line 7 for simplicity, however it should be done in place.

3.3 RHQR reconstructed from HQR

If W is entirely available at the begining of the procedure, we remark that in exact arithmetics, all elements of RHQR can be deduced from the output of the Householder QR factorization of ΨW. Indeed, in the light of Theorem 3.5, the latter writes

$$\begin{bmatrix} W_1 \\ \Omega W_2 \end{bmatrix} = \left(I_{\ell+m} - \begin{bmatrix} U_1 \\ \Omega U_2 \end{bmatrix} T \begin{bmatrix} U_1^t & (\Omega U_2)^t \end{bmatrix} \right) \begin{bmatrix} R \\ 0_{\ell\times m} \end{bmatrix} \quad \text{(HQR factorization)},$$

where U_1, U_2 denote respectively the first m rows and last $n-m$ rows of U_m output by RHQR of W, T is their shared T factor, and R is their shared R factor. To retrieve U_2, we may follow the construction of the randomized Householder vectors $u_1, \ldots, u_m$, focusing on their last $n-m$ entries $u_1^{(2)}, \ldots, u_m^{(2)}$. With the same notations as in Algorithms 1 and 3, it is straightforward to see that $u_1^{(2)}$ is simply $w_1^{(2)}$ divided by α_1 from line 10 of Algorithm 1. Then for $u_j^{(2)}$, $j \ge 2$,

Algorithm 3: Randomized-Householder QR (RHQR) (left-looking)

Input: Matrix $W \in \mathbb{R}^{n\times m}$, matrix $\Omega \in \mathbb{R}^{\ell\times n}$, $m < l \ll n$

Output: R, U, S, T such that $S = \Psi U$, with Ψ as in (11), and such that (15) holds. Q factor implicit : $Q = (I_n - UTS^t)\,[I_m;\ 0_{\ell\times m}]$, $W = QR$.

function `RHQR_left(`W, Ω`):`

- **for** $j = 1 : m$ **do**
  - $w \leftarrow w_j$
  - **if** $j \geq 2$ **then**
    - Sketch $\Omega(w)_{m+1:n}$
    - $z \leftarrow (w)_{1:m} \oslash \Omega(w)_{m+1:n}$ # i.e $z \leftarrow \Psi w$
    - $w \leftarrow w_j - U_{j-1}T^t_{j-1}S^t_{j-1}z$ $\#U_{j-1} = (u_i)_{i\leq j-1}$, idem S_{j-1}, T_{j-1} = first $j-1$ rows of $(t_i)_{i\leq j-1}$
  - $v, v', \rho, \sigma, \beta =$ `RHVector(`$(w)_{j:n}$, Ω`)`,
  - $u_j \leftarrow 0_{j-1} \oslash v$
  - $s_j \leftarrow 0_{j-1} \oslash v'$
  - $r_j \leftarrow (w)_{1:j-1} \oslash (-\sigma\rho) \oslash 0_{m-j}$
  - $t_j \leftarrow 0_{j-1} \oslash \beta \oslash 0_{m-j}$
  - **if** $j \geq 2$ **then**
    - $(t_j)_{1:j-1} = -\beta \cdot T_{1:j-1,1:j-1}S^t_{j-1}s_j$
- **return** $R = (r_j)_{j\leq m},\ U = (u_j)_{j\leq m},\ S = (s_j)_{j\leq m},\ T = (t_j)_{j\leq m}$

Algorithm 4: Block Randomized-Householder QR (block RHQR)

Input: $W = \left[W^{(1)} \cdots W^{(b)}\right] \in \mathbb{R}^{n\times mb}$, matrix $\Omega \in \mathbb{R}^{(\ell+mb)\times(n-mb)}$, $mb < l \ll n$

Output: $R, (U^{(j)}, S^{(j)}\,T^{(j)})_{1\leq j\leq b}$ such that $W = \left(\prod_{j=1}^{b}(I_n - U^{(j)}T^{(j)}(S^{(j)})^t)\right) \cdot [R;\ 0_{\ell\times mb}]$

function `blockRHQR(`W, Ω`):`

- **for** $j = 1 : b$ **do**
  - $M \leftarrow W^{(j)}$
  - **for** $k = 1 : j-1$ **do**
    - Sketch $\Omega M_{mb+1:n,1:m}$
    - $Z \leftarrow \left[M_{1:mb,1:m};\ \Omega M_{mb+1:n,1:m}\right]$ # i.e $Z \leftarrow \Psi M = \Psi W^{(j)}$
    - $M \leftarrow M - U^{(j)}(T^{(j)})^t(S^{(j)})^t Z$
    - Sketch $\Omega M_{mb+1:n,1:m}$
    - $Z \leftarrow \left[M_{1:mb,1:m}, \Omega M_{mb+1:n,1:m}\right]$
  - $R_{1:(j-1)b,\,(j-1)b+1:jb} \leftarrow M_{1:(j-1)b,\,(j-1)b+1:jb}$
  - $R^{(j)}, U^{(j)}, S^{(j)}, T^{(j)} =$ `RHQR_left(`$W^{(j)}, \Omega$`)`
  - $R_{(j-1)b+1:jb,\,(j-1)b+1,jb} \leftarrow R^{(j)}$
- Return R, $\left\{U^{(j)}, S^{(j)}, T^{(j)}\right\}_{j\leq b}$

1. $w^{(2)} \leftarrow w_j^{(2)} - U_2 T^t (\Psi U)^t \Psi w_j x$

2. $u_j^{(2)} \leftarrow \alpha^{-1} \cdot w^{(2)}$

which corresponds to the formula $W_2 = U_2 T^t (\Psi U)^t \Psi W$. This formula should be consistent with that obtained by writing the last $n - m$ lines of the RHQR factorization of W, namely $W_2 = -U_2 T U_1^t R$. Indeed,

$$\Psi W = \begin{bmatrix} R \\ 0_{\ell \times m} \end{bmatrix} - \Psi U_m \cdot T_m \cdot U_1^t R$$

Then

$$(\Psi U)^t \Psi W = U_1^t R - (\Psi U)^t \Psi U \cdot T U_1^t R = U_1^t R - (T^{-1} + T^{-t}) \cdot T U_1^t R = -T^{-t} T U_1^t$$

Finally multiplying on the left by T^t, we get $T^t (\Psi U)^t \Psi W = -T U_1^t R$ (which shows that it is upper-triangular), hence $U_2 T^t (\Psi U)^t \Psi W = U_2 T U_1^t R$. Also, according to Theorem 3.5, the coefficient α_j scaling the j-th randomized Householder is equal to that scaling the j-th Householder vector in Householder QR of ΨW. These computations yield Algorithm 5 (reconstructRHQR). This algorithm makes a single synchronization in line 2. There are multiple ways of carrying out the computations retrieving U_2 in line 5. We choose the explicit recursion using the coefficients $(\alpha_j)_{1 \leq j \leq m}$ retrieved from the Householder factorization of ΨW. Experiments show that on some very difficult examples, this algorithm is more stable than Randomized Cholesky QR [2], see experiments in Section 7.

Algorithm 5: RHQR reconstructing randomized Householder vectors (reconstructRHQR)

Input: Matrix $W \in \mathbb{R}^{n \times m}$, matrix $\Omega \in \mathbb{R}^{\ell \times n}$, $m < l \ll n$
Output: R, U, S, T such that $S = \Psi U$, with Ψ defined as in (11)

```
1 function reconstructRHQR(W, Ω):
2     Sketch ΩW_{m+1:n,1:m}
3     Z ← [W_{1:m,1:m}; ΩW_{m+1:n,1:m}]    # i.e Z ← ΨW
4     S, T, R ← HouseholderQR(Z) #Householder vectors S = ΨU ∈ R^{(ℓ+m)×m}, T factor, R factor
5     Backward solve for X in W_{m+1:n,1:m} = X · ut(T^t Y^t Z)
6     return R,  U = [S_{1:m,1:m}; X],  S,  T
```

Remark 3.6. The unique LU factorization of $\begin{bmatrix} I_m ; \ 0_{(n-m)\times m} \end{bmatrix} - Q$ yielded by Householder QR translates in the unique LU factorization of both $\begin{bmatrix} I_m ; \ 0_{(n-m)\times m} \end{bmatrix} - Q$ and $[I_m ; \ 0_{\ell \times m}] - \Psi Q$ yielded by the RHQR factorization. Indeed, writing the thin Q factor yielded by RHQR,

$$Q = \begin{bmatrix} I_m \\ 0_{(n-m)\times m} \end{bmatrix} - U_m T_m U_1^t \iff \begin{bmatrix} I_m \\ 0_{(n-m)\times m} \end{bmatrix} - Q = \underbrace{U}_{\substack{\text{lower-triangular,} \\ \text{diagonal of ones}}} \cdot \underbrace{T U_1^t}_{\text{upper-triangular}} \tag{18}$$

which is the unique LU factorization of $\begin{bmatrix} I_m ; \ 0_{(n-m)\times m} \end{bmatrix} - Q$. Now sketching this equation, and by design of Ψ which preserves the upper block of the sketched matrix,

$$\begin{bmatrix} I_m \\ 0_{\ell \times m} \end{bmatrix} - \Psi Q = \underbrace{\begin{bmatrix} U_1 \\ \Omega U_2 \end{bmatrix}}_{\substack{\text{lower-triangular,} \\ \text{diagonal of ones}}} \cdot \underbrace{T U_1^t}_{\text{upper-triangular}} . \tag{19}$$

which is the unique LU factorization of $[I_m;\ 0_{\ell\times m}] - \Psi Q$. Hence both the randomized Householder vectors and their sketches can be obtained from Q and ΨQ. In turn, if another orthonormalization process of W yields a factor Q such that $\mathrm{Range}(Q) = \mathrm{Range}(W)$, $(\Psi Q)^t \Psi Q = I_m$, the randomized Householder vectors can be obtained from the LU factorization of $[I_m;\ 0_{\ell\times m}] - \Psi Q$. The L factor of the latter is ΨU, from which we can deduce U_1. T can be deduced from ΨU_m thanks to (10), but can also be deduced from the upper-triangular factor of the LU factorization, which is $U_1^t T$. If R is not given, it can be retrieved as $(\Psi Q)^t \Psi W$. The factor U_2 can be retrieved in many ways, for example as in reconstructRHQR.

4 Application to Arnoldi and GMRES

As in [14, Algorithm 6.3, p163], the left-looking RHQR factorization is straightforwardly embedded into the Arnoldi iteration, yielding Algorithm 6. Before detailing the algorithm, let us justify right away that it computes a basis of the Krylov subspace:

Proposition 4.1. *Algorithm 6 applied to $A \in \mathbb{R}^{n\times n}$, $b, x_0 \in \mathbb{R}^n$, and $\Omega \in \mathbb{R}^{\ell\times(n-m)}$ produces $Q_{m+1} = [Q_m \mid q_{m+1}] \in \mathbb{R}^{n\times(m+1)}$ and $H_{m+1,m} \in \mathbb{R}^{(m+1)\times m}$ such that*

$$AQ_m = Q_{m+1} H_{m+1,m}. \tag{20}$$

and q_1 is a multiple of $b - Ax_0$.

Proof. The proof is almost identical to that found in [14]. Note that $\mathcal{P}_m^{-1} \neq \mathcal{P}_m^t$ for randomized Householder reflectors, which has no consequence in the proof. By the definition of the vector z at line 13, once the randomized Householder reflector is applied at line 8, we obtain:

$$h_j = P_{j+1} P_j \cdots P_1 \cdot A q_j. \tag{21}$$

Since the coordinates $j+2, j+3 \cdots n$ of h_j are zero, it is invariant under the remaining reflectors:

$$h_j = P_m \cdots P_{j+2} h_j = P_m \cdots P_1 A q_j. \tag{22}$$

This relation being true for all j, we obtain the factorization:

$$P_m \cdots P_1 \left[r_0 \ \ A q_1 \ \ \ldots \ \ A q_m \right] = \left[h_0 \ \ h_1 \ \ \cdots \ \ h_m \right].$$

Multiplying on the left by the reflectors in reverse order $P_1 \cdots P_m$, we obtain:

$$\left[r_0 \ \ A q_1 \ \ \ldots \ \ A q_m \right] = P_1 \cdots P_m \left[h_0 \ \ h_1 \ \ \cdots \ \ h_m \right] = \mathcal{P}_m^{-1} \left[h_0 \ \ h_1 \ \ \cdots \ \ h_m \right]$$

which concludes the proof. □

As Householder-Arnoldi costs more than MGS-Arnoldi, RHQR Arnoldi costs more than RGS Arnoldi. The reason is similar : the vector z used as input for the $j+1$-th RHQR iteration (z in Algorithm 6) is not Aq_j, but $P_j \cdots P_1 A q_j$. This induces an additional generalized matrix vector product in line 13, and in our context, an additional sketch and thus an additional synchronization in line 15. This additional cost (one generalized matrix vector product and one sketch) is put into perspective by the greater stability of RHQR-Arnoldi when compared to RGS-Arnoldi on difficult examples in simple precision, see Section 7. Since this work concerns large-scale contexts, we choose the current presentation in which the Arnoldi basis is its implicit form. This form is sufficient to compute the coordinates of an element from the Krylov subspace in the computed basis. However, one can also store the vectors r from Algorithm 6 along the iteration if the storage space is available.

We also adapt the GMRES process to the RHQR-Arnoldi from Algorithm 6 as in [14][Algorithm 6.10, p174].

Algorithm 6: Randomized Householder Arnoldi

Input: Matrix $A \in \mathbb{R}^{n\times n}$, $x_0, b \in \mathbb{R}^n$, $m \in \mathbb{N}^*$, matrix $\Omega \in \mathbb{R}^{\ell\times n}$, $m < l \ll n$,
Output: Matrices $Q_m, H_{m+1,m}$ such that (20) holds (Q_m in compact form)

function `RHQR_Arnoldi`(A, b, x_0, Ω)**:**
- $w \leftarrow b - Ax_0$
- $z \leftarrow \Psi w$
- **for** $j = 1 : m+1$ **do**
  - $v, v', \rho, \sigma, \beta \leftarrow$ `RHVector`$((w)_{j:n}, \Omega)$
  - $u_j \leftarrow 0_{j-1} \oslash v$
  - $s_j \leftarrow 0_{j-1} \oslash v'$
  - $h_{j-1} \leftarrow (w)_{1:j-1} \oslash (-\sigma\rho) \oslash 0_{m+1-j}$
  - Update T_j as in Algorithm 3
  - **if** $j \le m$ **then**
    - $q_j \leftarrow e_j^{\ell+m} - U_j T_j S_j^t e_j^{\ell+m}$ # Optional : store q_j
    - Compute $y = Aq_j$, sketch $\Omega(y)_{m+1:n}$
    - $w \leftarrow y - U_j T_j^t S_j^t \left[(y)_{1:m} \oslash \Omega(y)_{m+1:n}\right]$ # i.e $w \leftarrow Aq_j - U_j T_j^t S_j^t \Psi A q_j$
    - Sketch $\Omega(w)_{m+1:n}$
    - $z \leftarrow w_{1:m} \oslash \Omega(w)_{m+1:n}$ # i.e $z \leftarrow \Psi w$
- Set H as the first $m+1$ rows of $[h_1\ h_2 \dots h_m]$, discard h_0.
- **return** $(u_j)_{j\le m+1}, (s_j)_{j\le m+1}, (t_j)_{j\le m+1}, H$ $\#S = \Psi U$. Optional : return $(q_j)_{1\le j\le m}$

Algorithm 7: Randomized Householder GMRES

Input: Matrix $A \in \mathbb{R}^{n\times n}$, $x_0, b \in \mathbb{R}^n$, $m \in \mathbb{N}^*$, matrix $\Psi \in \mathbb{R}^{\ell\times n}$, $m < l \ll n$,
Output: $x_m \in \mathcal{K}_j(A, r_0)$ such that (23) holds

function `RHQR_GMRES`(A, b, x_0, Ω)**:**
- Sketch $\Omega(b)_{m+1:n}$
- $\beta = \|(b)_{1:m} \oslash \Omega(b)_{m+1:n}\|$ # i.e $\beta = \|\Psi b\|$
- $[U \mid u], [S \mid s], [T \mid t], H \leftarrow$`RHQR_Arnoldi`$(A, b, x_0, \Omega)$
- Solve Hessenberg system $Hy = \beta e_1^{m+1}$
- $x \leftarrow (I_n - UTS^t)(y \oslash 0_\ell)$
- **return** x

Proposition 4.2. *Let $H_{m+1,m}, Q_m$ and Q_{m+1} be the Arnoldi relation output by RHQR-Arnoldi in Algorithm 6. Let us pick*

$$x_m = Q_m y, \quad y := \arg\min_{y \in \mathbb{R}^m} \|\beta e_1 - H_{m+1,m} y\|, \quad \beta = \|\Omega b\|.$$

Then

$$x_m = \arg\min_{x \in \mathcal{K}_m(A,x_0)} \|\Psi(b - Ax)\|. \tag{23}$$

In particular, if W denotes any basis of $\mathcal{K}_m(A, x_0)$ and if Ω is an ϵ-embedding for Range(W_2)*, then*

$$\|b - Ax_m\| \leq \frac{1+\epsilon}{1-\epsilon} \arg\min_{x \in \mathbb{R}^n} \|b - Ax\|$$

Proof. Just as in RGS, it follows from the orthogonality of the sketch ΨQ_{m+1} of the Arnoldi basis built by Algorithm 6:

$$\arg\min_{\rho \in \mathbb{R}^m} \|\beta e_1 - H_{m+1,m}\rho\| = \arg\min_{\rho \in \mathbb{R}^m} \|\Psi Q_{j+1} \left[\beta e_1 - H_{m+1,m}\rho\right]\| = \arg\min_{\rho \in \mathbb{R}^m} \|\Psi \left[b - AQ_j\rho\right]\|$$

□

5 Randomized Modified Gram-Schmidt

In this section, we derive a randomized Modified Gram-Schmidt (RMGS) process based on the RHQR algorithm applied to matrix $W \in \mathbb{R}^{n \times m}$ topped by a $m \times m$ bloc of zeros.

Let us take $\Omega \in \mathbb{R}^{\ell \times n}$, and suppose that W and ΩW are full-rank (hence the RHQR factorization does not crash). Let us form $\Psi \in \mathbb{R}^{(\ell+m)\times(n+m)}$, and perform the RHQR factorization of the also full rank matrix

$$\overline{W} = \begin{bmatrix} 0_{m \times m} \\ W \end{bmatrix} \in \mathbb{R}^{(n+m)\times m}. \tag{24}$$

We stress that in such context, the sign in the construction of the randomized Householder vectors is never flipped. Let us denote $\bar{u}_1 \dots \bar{u}_m \in \mathbb{R}^{n+m}$ the randomized Householder vectors, forming the matrix $\overline{U}$. Let us also scale them such that the diagonal of $\overline{U} \in \mathbb{R}^{n \times m}$ is the identity matrix I_m. Let us denote T the associated T factor. By construction, $\overline{U}$ and its sketch $\Psi\overline{U}$ are of the form

$$\overline{U} = \begin{bmatrix} I_m \\ U \end{bmatrix} \in \mathbb{R}^{n \times m}, \quad \Psi\overline{U} = \begin{bmatrix} I_m \\ \Omega U \end{bmatrix} \in \mathbb{R}^{(\ell+m)\times m}, \quad U \in \mathbb{R}^{n \times m}.$$

Supposing that $[0_{m\times m}; W]$ is full rank and seeing as the composition of the randomized Householder reflectors is non-singular, R is also non-singular. Hence, writing the factorization in its compact form, we get (in exact arithmetics)

$$\begin{bmatrix} R - TR \\ -UTR \end{bmatrix} = \begin{bmatrix} 0_{m\times m} \\ W \end{bmatrix} \implies \begin{cases} T = I_m \\ -UR = W \end{cases}$$

Let us write $\overline{Q}$ the thin Q factor output by RHQR. In this context,

$$\overline{Q} = \begin{bmatrix} I_m \\ 0_{(n+m)\times m} \end{bmatrix} - \begin{bmatrix} T \\ U \end{bmatrix} T \begin{bmatrix} I_m & (\Omega U)^t \end{bmatrix} \cdot \begin{bmatrix} I_m \\ 0_{\ell \times m} \end{bmatrix} = \begin{bmatrix} I_m \\ 0_{(n+m)\times m} \end{bmatrix} - \begin{bmatrix} T \\ U \end{bmatrix} = - \begin{bmatrix} 0_{m\times m} \\ U \end{bmatrix}.$$

$\Psi\overline{Q}$ is orthogonal by construction, hence, denoting $Q = -U$, we see that ΩQ is orthogonal. Hence $W = QR$ is a randomized QR factorization.

Let us then study in detail the arithmetics of one iteration. Remembering that we choose to scale u_1 such that its first coordinate is +1, it is clear that it is given by

$$z_1 \leftarrow \begin{bmatrix} 0_m \\ w_1 \end{bmatrix} - \|\Omega w_1\| e_1, \quad u_1 \leftarrow \frac{1}{-\|\Omega w_1\|} \cdot z_1 = \begin{bmatrix} e_1 \\ -w_1/\|\Omega w_1\| \end{bmatrix} := \begin{bmatrix} e_1 \\ -q_1 \end{bmatrix}, \quad \|\Psi u_1\|^2 = 2, \quad \|\Omega q_1\| = 1$$

(remark that both scaling of u_1 are equivalent in this context). Hence the first vector q_1 is the same as that built by RGS on W with sketching matrix Ω. At iteration $j \geq 2$, denoting T_{j-1} the T factor at iteration $j-1$ denoting U_{j-1} the matrix formed by $u_1, \dots u_{j-1}$, denoting Q_{j-1} the matrix formed by $q_1, \dots q_{j-1}$, we first write

$$z_j \leftarrow \mathcal{P}_{j-1} \begin{bmatrix} 0_{j-1} \\ 0_{m-j+1} \\ w_j \end{bmatrix} = \begin{bmatrix} 0_{j-1} \\ 0_{m-j+1} \\ w_j \end{bmatrix} - \begin{bmatrix} I_{j-1} \\ 0_{1\times(m-j+1)} \\ -Q_{j-1} \end{bmatrix} T_{j-1}^t \begin{bmatrix} I_{j-1} & 0_{(m-j+1)\times 1} & (-\Omega Q_{j-1})^t \end{bmatrix} \begin{bmatrix} 0_{j-1} \\ 0_{m-j+1} \\ w_j \end{bmatrix}$$

yielding

$$z_j \leftarrow \begin{bmatrix} T_{j-1}^t (\Omega Q_{j-1})^t \Omega w_j \\ 0_{m-j+1} \\ w_j - Q_{j-1} T_{j-1}^t (\Omega Q_{j-1})^t \Omega w_j \end{bmatrix} =: \begin{bmatrix} x \\ 0_{m-j+1} \\ y \end{bmatrix}.$$

By design of the RHQR algorithm, x (i.e the first $j-1$ entries of z_j) gives the first $j-1$ entries of the j-th column of the R factor. Now computing the vector u_j,

$$\bar{u}_j \leftarrow \begin{bmatrix} 0_m \\ y \end{bmatrix} - \|\Omega y\| e_j, \quad u_j \leftarrow \frac{1}{-\|\Omega y\|} \cdot \bar{u}_j = \begin{bmatrix} 0_{j-1} \\ 1 \\ 0_{m-j} \\ -y/\|\Omega y\| \end{bmatrix} = \begin{bmatrix} e_j \\ -q_j \end{bmatrix}$$

By design of the RHQR algorithm, the j-th entry of the j-th column of R is given by $\|\Omega y\|$. Seeing as $T_m = I_m$ in exact arithmetics, we are supposed to get

$$\tilde{q}_j \leftarrow w_j - Q_{j-1}(\Omega Q_{j-1})^t \Omega w_j, \quad q_j \leftarrow \frac{1}{\|\Omega \tilde{q}_j\|} \cdot \tilde{q}_j$$

i.e the factor Q is precisely that output by RGS in exact arithmetics.

6 Randomized reflectors with partial sketching

We showcase here another randomization of the Householder reflector, allowing for another RHQR process (trimRHQR) which produces accurate factorizations and well conditioned bases in all cases of our test set. This modified algorithm does not compute a basis whose sketch is orthogonal, but experimentally we observed that the sampling size is smaller than the one required by RHQR.

Let us pick a matrix $\Omega \in \mathbb{R}^{\ell \times m}$ with unit columns. Let us rule out the possibility that the m first columns of Ω are orthogonal to the last $n-m$, nor that they are orthogonal to each other. One can verify as in Proposition 3.4 that with any such matrix Ω, it is possible to eliminate all coordinates of an input vector below its first entry, setting $u = w \pm \|\Omega u\| e_1$, and using the fact that $\|\Omega e_1\| = 1$:

$$P(u, \Omega) \cdot w = \pm \|\Omega w\| e_1$$

If we denote $w = c \oslash d$, $d \in \mathbb{R}^{n-1}$, if we denote $w' = 0 \oslash d$ and set $u = w' - \|\Omega w'\| e_2$, one will remark that the operator $P(u, \Omega)$ should modify the first entry of some input vector. This comes from the

Algorithm 8: Randomized Modified Gram-Schmidt (RMGS)

Input: Matrix $W \in \mathbb{R}^{n\times m}$, matrix $\Omega \in \mathbb{R}^{\ell\times n}$
Output: Matrices R, Q such that $W = QR$

for $j = 1 : m$ **do**
  $w \leftarrow w_j$
  if $j \geq 2$ **then**
    $z \leftarrow \Omega w$
    $r_j \leftarrow T_{j-1}^t S_{j-1}^t z$ $\#S_{j-1} = (s_i)_{1\leq i\leq j-1}$, T_{j-1} = first j rows and j columns of $(t_i)_{1\leq i\leq j}$
    $w \leftarrow w_j - Q_{j-1} r_j$ $\#U_{j-1} = (u_i)_{1\leq i\leq j-1}$
  $z \leftarrow \Omega w$
  $\rho = \|\Omega z\|$
  $q_j \leftarrow \rho^{-1} \cdot w, \quad s_j \leftarrow \rho^{-1} \cdot s$
  $r_j \leftarrow r_j \oslash \rho \oslash 0_{m-j}$
  $t_j \leftarrow 0_{j-1} \oslash 1 \oslash 0_{m-j}$
  $(t_j)_{1:j-1} \leftarrow -T_{1:j-1,1:j-1} \cdot S_{j-1}^t s_j$
Optional : $Q_j \leftarrow Q_j \cdot T_m$
return $(r_j)_{j\leq m}, (q_j)_{j\leq m}$

geometric properties that we ruled out for the first m columns of Ω (this observation motivated for the design of Ψ from Section 3 in the first place). We can still enforce the behavior we seek. Let us denote $\Omega' \in \mathbb{R}^{\ell\times(n-1)}$ the matrix formed by the last $n-1$ columns of Ω. For the same reasons as before, denoting $u' = d - \|\Omega' d\| e_1 \in \mathbb{R}^{n-1}$, we can build a reflector $P(\Omega', u') \in \mathbb{R}^{(n-1)\times(n-1)}$ that is sketch isometric with respect to Ω', and that verifies

$$P(\Omega', u') \cdot w' = \|\Omega' w'\| e_1 \in \mathbb{R}^{n-1}$$

Denoting now

$$H' = \left[\begin{array}{c|c} 1 & \\ \hline & P(\Omega', u') \end{array}\right] \in \mathbb{R}^{n\times n}$$

we get by design a reflector that was forced to leave the first line of any input untouched, while cancelling the entries of a given input below its second one. However, we have lost the sketch-isometric property of the final reflector of $\mathbb{R}^n$. It is by design sketch-isometric for the following matrix

$$\Psi' = \left[\begin{array}{c|c} 1 & \\ \hline & \Omega' \end{array}\right] \in \mathbb{R}^{(\ell+1)\times n}$$

Let us generalize this construction. First, we define the sequence of sampling matrices

$$\Psi_1 := \Omega; \qquad \forall j \in \{2 \dots m\}, \quad \Psi_j := \left[\begin{array}{c|c} I_{j-1} & \\ \hline & \Omega_{j:n} \end{array}\right] \in \mathbb{R}^{(\ell+j-1)\times n} \tag{25}$$

and the modified randomized Householder reflectors

$$\forall z \in \mathbb{R}^n \setminus \mathrm{Ker}\left(\Psi_j\right), \quad H(z, \Omega, j) := P(z, \Psi_j)) = I_n - \frac{2}{\|\Psi_j z\|^2} \cdot z \cdot (\Psi_j z)^t \Psi_j \quad \in \mathbb{R}^{n\times n}. \tag{26}$$

We remark that if u_j is a vector which first $j-1$ entries are nil, then (26) can be equivalently formulated

$$P(u_j, \Psi_j) = H(u_j, \Omega, j) = I_n - \frac{2}{\|\Omega u_j\|^2} \cdot u_j (\Omega u_j)^t \begin{bmatrix} 0_{\ell \times (j-1)} & \Omega_{j:n} \end{bmatrix} \tag{27}$$

We summarize their properties in the following proposition:

Proposition 6.1. *Let $j \in \{1, \dots, m\}$, let $w \in \mathbb{R}^n$, where $w = c \oslash d$, with $c \in \mathbb{R}^{j-1}$ and $d \in \mathbb{R}^{n-(j-1)}$, where $\Omega \in \mathbb{R}^{\ell \times n}$ has unit first m columns. Let us denote $w' = 0_{j-1} \oslash d$. Suppose that $\Omega d \neq 0$ and that w' is not a multiple of e_j. Define $u_j = w' - \|\Omega w'\| e_j \in \mathbb{R}^n$. Then $H(u_j, \Omega, j) \cdot w = c \oslash (\|\Omega w'\| e_1^{n-j+1})$.*

Proof. The proof is identical to that of Proposition 3.4, only with the important hypothesis that the vectors $\Omega e_1 \dots \Omega e_m$ are unit vectors. □

Using successive modified randomized Householder reflectors $H(u_1, \Omega, 1), \dots, H(u_m, \Omega, m)$, it is then possible to get the following factorization

$$H(u_m, \Omega, m) \cdots H(u_1, \Omega, 1) \cdot W = \begin{bmatrix} R \\ 0_{(n-m)\times m} \end{bmatrix} \iff W = H(u_1, \Omega, 1) \cdots H(u_m, \Omega, m) \cdot \begin{bmatrix} R \\ 0_{(n-m)\times m} \end{bmatrix} \tag{28}$$

where $R \in \mathbb{R}^{m \times m}$ is upper-triangular by construction. We show this procedure in Algorithms 9 and 10. In line 2 of Algorithm 9, one communication is made to sketch w'. If the sketches of the first m canonical vectors are available, then no further synchronization is required. In Algorithm 10, we make the same memory management indications as in Algorithm 2. We note that the sketching made in line 8 can be done in the communication made before to compute the modified randomized Householder vector.

Algorithm 9: Computation of modified randomized Householder vector (trimRHVector)

Input: vector $\tilde{w} \in \mathbb{R}^{n-j+1}$, $j \in \{1, \dots, m\}$, $\tilde{\Omega} \in \mathbb{R}^{\ell \times (n-j+1)}$ with unit first $m - j + 1$ columns
Output: $v, v', \rho, \sigma, \beta$ such that $\tilde{w} - \beta \langle v', \tilde{\Omega}\tilde{w} \rangle v = -\sigma \rho e_1^{n-j+1}$

1 **function** `trimRHVector(`$\tilde{w}$`,` $\tilde{\Omega}$`):`
2 Sketch $\tilde{\Omega}\tilde{w}$
3 $\rho \leftarrow \|\tilde{\Omega}\tilde{w}\|$
4 $\sigma = \operatorname{sign}\langle \tilde{w}, e_1 \rangle$
5 $v \leftarrow \tilde{w} + \sigma \rho e_1$
6 Sketch $\tilde{\Omega} v$
7 $v' \leftarrow \tilde{\Omega} v$
8 Choose α as the first entry of v' or as $\sqrt{2}/\|v'\|$
9 $v \leftarrow \alpha v$
10 $v' \leftarrow \alpha v'$
11 $\beta = 2/\|s\|^2$ # for second choice of α, just set $\beta = 1$
12 **return** $v, v', \rho, \sigma, \beta$

We focus now on the formulas that allow to handle multiple modified randomized Householder reflectors efficiently, allowing for the left-looking version.

Algorithm 10: trimRHQR (right-looking)

Input: Matrix $W \in \mathbb{R}^{n\times m}$, matrix $\Omega \in \mathbb{R}^{\ell\times n}$, $m < l \ll n$
Output: R, U, S such that $S = \Omega U$ and (28) holds

function `trimRHQR_right(`W, Ω`):`
  for $j = 1 : m$ **do**
    $r_j \leftarrow (w_j)_{1:j-1}$
    $w \leftarrow w_j$
    $u_j, \rho_j, \sigma, \beta =$ `trimRHVector(`w_j, Ω, j`)`
    $r_j \leftarrow r_j \oslash (-\sigma\rho) \oslash 0_{m-j}$
    if $j \geq 2$ **then**
      $X \leftarrow \left[0_{\ell\times(j-1)} \;\; \Omega_{j:n}\right] W_{j+1:n}$
      $W_{j+1:m} \leftarrow W_{j+1:m} - \beta u_j s_j^t X$
  return $(r_j)_{1\leq j\leq m}, (u_j)_{1\leq j\leq m}, (s_j)_{1\leq j\leq m}$

Proposition 6.2. *Let $u_1 \dots u_m$ be the Householder vectors generated by Algorithm 10. For all $j \in \{1, \dots, m\}$, define $\beta_j = 2/\|\Omega u_j\|^2$. Define by induction the matrix T_j for all $1 \leq j \leq m$ as*

$$\begin{aligned} &T_1 \in \mathbb{R}^{1\times 1}, \;\; T_1 = [\beta_1] \\ &\forall 1 \leq j \leq m-1, \;\; T_{j+1} = \begin{bmatrix} T_j & -\beta_j \cdot T_j (\Omega U_j)^t \Omega u_{j+1} \\ 0_{1\times j} & \beta_j \end{bmatrix} \end{aligned} \tag{29}$$

Then for all $1 \leq j \leq m$, we have the factored form:

$$\mathcal{H}_j^{-1} := H(u_1, \Omega, 1) \cdots H(u_j, \Omega, j) = P(u_1, \Psi_1) \cdots P(u_j, \Psi_j) = I - U_j T_j \mathrm{ut}\left((\Omega U_j)^t \Omega\right). \tag{30}$$

Proof. See appendix. □

Proposition 6.3. *With the previous notations, and regardless of the scaling of the modified randomized Householder vectors, we get*

$$\forall j \in \{1, \dots, m\}, \quad T_j^{-1} + T_j^{-t} = (\Omega U_j)^t \Omega U_j, \quad T_j = \left[\mathrm{ut}\left((\Omega U_j)^t \Omega U_j\right) - \frac{1}{2}\mathrm{Diag}\left((\Omega U_j)^t \Omega U_j\right)\right]^{-1} \tag{31}$$

In particular, when the randomized Householder vectors are scaled such that their sketched norm is $\sqrt{2}$, we get

$$\forall j \in \{1, \dots, m\}, \quad T_j = \left[\mathrm{ut}\left((\Omega U_j)^t \Omega U_j\right) - I_j\right]^{-1} \tag{32}$$

Proof. The T factor from Proposition 6.3 has the same formula as that of the composition of randomized Householder reflectors from Section 3. □

If the right-wise compositions admit the same T factor as for the randomized Householder reflectors from section 3, the left-wise compositions $P(u_m, \Psi_m) \cdots P(u_1, \Psi_1) = H(u_m, \Omega, m) \cdots H(u_1, \Omega, 1)$ feature a T factor that slightly differs from T^t. Straightforward computations yield the following result:

Proposition 6.4. *Define by induction the matrix $\tilde{T}_j$ for all $1 \leq j \leq m$ as*

$$\begin{aligned}
&\tilde{T}_1 \in \mathbb{R}^{1\times 1}, \;\; \tilde{T}_1 = [\beta_1] \\
&\forall 1 \leq j \leq m-1, \quad \tilde{t}_j = -\beta \cdot \tilde{T}_{1:j-1,1:j-1}\left(\left[0_{\ell\times(j-1)} \;\; \Omega_{j:n}\right] U_{j-1}\right)^t \Omega u_j \in \mathbb{R}^{(j-1)\times 1} \\
&\forall 1 \leq j \leq m-1, \;\; \tilde{T}_{j+1} = \begin{bmatrix} \tilde{T}_j & \tilde{t}_j \\ 0_{1\times(j-1)} & \beta_{j+1} \end{bmatrix}
\end{aligned} . \tag{33}$$

Then for all $1 \leq j \leq m$, we have the factored form

$$\mathcal{H}_j = H(u_j, \Omega, j) \cdots H(u_1, \Omega, 1) = P(u_j, \Psi_j) \cdots P(u_1, \Psi_1) = I - U_j \tilde{T}_j^t \mathrm{ut}\left((\Omega U_j)^t \Omega\right). \tag{34}$$

We also get an equivalent formulation for $\tilde{T}_j$ from the Woodburry-Morrison formula:

Proposition 6.5. *With the previous notations, and regardless of the scaling of the modified randomized Householder vectors,*

$$\tilde{T}_j^t = -\left[\mathrm{ut}\left((\Omega U_j)^t \Omega U_j\right) - \frac{1}{2}\mathrm{Diag}\left((\Omega U_j)^t \Omega U_j\right) - \mathrm{ut}\left((\Omega U_j)^t \Omega\right) U_j\right]^{-1} = -\left[T_j^{-1} - \mathrm{ut}\left((\Omega U_j)^t \Omega\right) U_j\right]^{-1} \tag{35}$$

Proof. See appendix. □

All these formulas allow to identify the compact factorizations

$$W = \left(I_n - U_m T_m \mathrm{ut}\left((\Omega U_m)^t \Omega\right)\right) \cdot \begin{bmatrix} R \\ 0_{(n-m)\times m} \end{bmatrix} \iff \left(I_n - U_m \tilde{T}_m^t \mathrm{ut}\left((\Omega U_m)^t \Omega\right)\right) \cdot W = \begin{bmatrix} R \\ 0_{(n-m)\times m} \end{bmatrix} \tag{36}$$

and to derive the left-looking version of trimRHQR, given in Algorithm 11. We make the same memory management indications as in Algorithm 3. If needed, the output can be used to produce the thin factor $Q_m = \mathcal{H}_m^{-1} I_{1:m}$. As to Line 6, the main challenge is the application of $\mathrm{ut}\left((\Omega U_j)^t \Omega\right)$, which can be done in several ways. For example, one can first compute $(\Omega U_j)^t \Omega x$, then substract the vector

$$\begin{bmatrix} 0 \\ \langle \Omega u_2, x_1 \Omega_{1:1} \rangle \\ \langle \Omega u_3, \Omega_{1:2} x_{1:2} \rangle \\ \vdots \\ \langle \Omega u_j, \Omega_{1:j} x_{1:j} \rangle \end{bmatrix}.$$

If Ω is not stored as a dense matrix but can only be applied to a vector (e.g SRHT OSE), one can also compute explicitly along the iterations

$$\mathrm{slt}\left((\Omega U_j)^t \Omega\right) = \begin{bmatrix} 0 & 0 & 0 & \dots & & \dots & 0 \\ \langle \Omega u_2, \Omega e_1 \rangle & 0 & 0 & \dots & & \dots & 0 \\ \langle \Omega u_3, \Omega e_1 \rangle & \langle \Omega u_3, \Omega e_2 \rangle & 0 & \dots & & & \\ \vdots & & \ddots & & & & \\ \langle \Omega u_j, \Omega e_1 \rangle & \langle \Omega u_j, \Omega e_2 \rangle & \dots & \langle \Omega u_j, \Omega e_{j-1} \rangle & 0 & \dots & 0 \end{bmatrix}. \tag{37}$$

Then, the computation of $\mathrm{ut}\left((\Omega U_j)^t \Omega\right) x$ is replaced by that of $(\Omega U_j)^t \Omega x$ followed by the substraction of $\mathrm{slt}\left((\Omega U_j)^t \Omega\right) x$. The same approach can be used to compute the update of $\tilde{T}_j$ when using (33).

We note that a block version of Algorithm 11 can be straightforwardly derived as for Algorithms 3 and 4.

Algorithm 11: trimRHQR (left-looking)

Input: Matrix $W \in \mathbb{R}^{n\times m}$, matrix $\Omega \in \mathbb{R}^{\ell\times n}$, $m < l \ll n$
Output: $R, U_m, \Omega U_m, T_m, \tilde{T}_m$ such that (36) holds

function `trimRHQR_left(`W, Ω`):`
  for $j = 1 : m$ **do**
    $w \leftarrow w_j$
    if $j \geq 2$ **then**
      Sketch Ωw, compute $\mathrm{ut}\left((\Omega U_{j-1})^t \Omega\right) \cdot w$ # e.g using (37)
      $w \leftarrow w - U_{j-1}\tilde{T}_{j-1}^t \mathrm{ut}\left((\Omega U_{j-1})^t \Omega\right) \cdot w$
    $v, s_j, \rho, \sigma, \beta =$ `trimRHVector(`$(w)_{j:n}$, Ω`)`
    $u_j \leftarrow 0_{j-1} \oslash v$
    $r_j \leftarrow (w)_{1:j-1} \oslash (-\sigma\rho) \oslash 0_{m-j}$
    $t_j \leftarrow 0_{j-1} \oslash \beta \oslash 0_{m-j}$
    $\tilde{t}_j \leftarrow 0_{j-1} \oslash \beta \oslash 0_{m-j}$
    if $j \geq 2$ **then**
      $(t_j)_{1:j-1} \leftarrow -\beta \cdot T_{1:j-1,1:j-1} S_j^t s_j$
      $(\tilde{t}_j)_{1:j-1} \leftarrow -\beta \cdot \tilde{T}_{1:j-1,1:j-1}\left(\left[0_{\ell\times(j-1)} \;\; \Omega_{j:n}\right] U_{j-1}\right)^t s_j$
  return $R = (r_j)_{1\leq j\leq m},\ U = (u_j)_{1\leq j\leq m},\ S = (s_j)_{1\leq j\leq m},\ T = (t_j)_{1\leq j\leq m}$

7 Numerical experiments

In this section, we test numerically the algorithms derived in this work. We first detail the matrices of our test set. We then compare left-looking RHQR to RGS on a difficult example where RHQR is the most stable, both in simple and double precision. We then compare reconstructRHQR and Randomized Cholesky QR (both algorithms perform a single synchronization) on the same difficult example, where reconstrcutRHQR is the most stable. Next, we compare RGS-GMRES and RHQR-GMRES on medium and high difficulty problems, and the results are consistent with those of the QR factorization. We then compare RMGS with RGS on the same difficult example, and observe a slight advantage for RMGS. We finally showcase the performance of trimRHQR on the same difficult examples, and observe that trimRHQR is as stable as RHQR. Furthermore, with sampling size inferior to the column dimension of W, it is stabler than RGS.

The first example is formed by uniformly discretized parametric functions, also used in [3], that we denote $C_m \in \mathbb{R}^{50000\times m}$. For all floating (and thus rational) numbers $0 \leq x, \mu \leq 1$, the function is defined as

$$f(x,\mu) = \frac{\sin\left(10(\mu + x)\right)}{\cos\left(100(\mu - x)\right) + 1.1}$$

and the associated matrix is

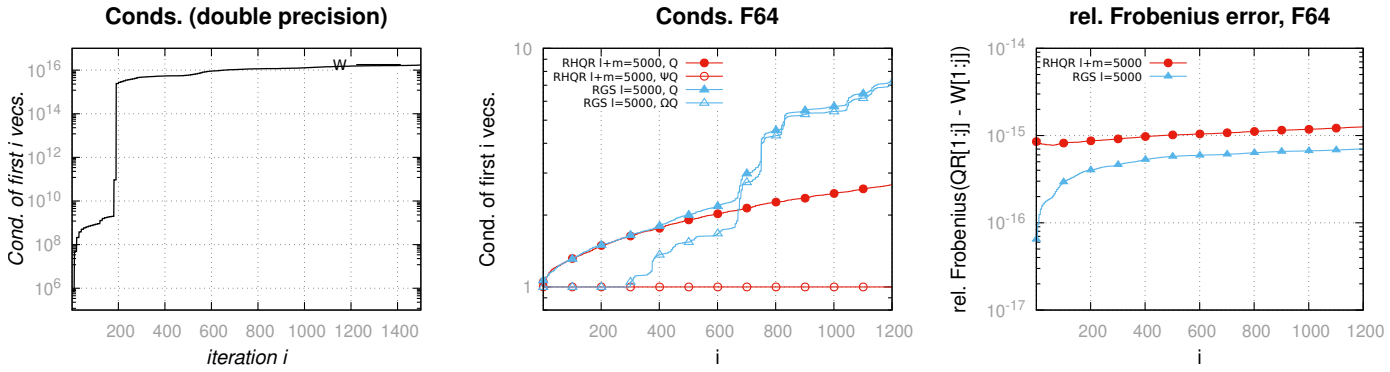
$$C_m \in \mathbb{R}^{n\times m},\ \ C_{i,j} = f\left(\frac{i-1}{n-1}, \frac{j-1}{m-1}\right), \quad i \in \{1\ldots n\}, \quad j \in \{1,\ldots,m\}. \tag{38}$$

The condition number of C_{1500} in double precision is displayed in Figure 1a, and that of C_{600} in single precision in Figure 2a.

For all experiments, the ϵ-embedding property of drawn matrix Ω is tested by first computing a deterministic QR factorization of W. If the factorization is accurate to machine precision, we check that

Name	Size	Cond.	Origin
C_m	$50000 \times m$	∞	synthetic functions
SiO2	155331	≈ 2300	quantum chemistry problem
El3D	32663	$\approx 10^{28}$	Near incompressible regime elasticity problem

Table 1: Set of matrices used in experiments



(a) Condition number of test matrix W

(b) Condition number of bases output by RHQR and RGS

(c) Factorization relative Frobenius error for RHQR and RGS

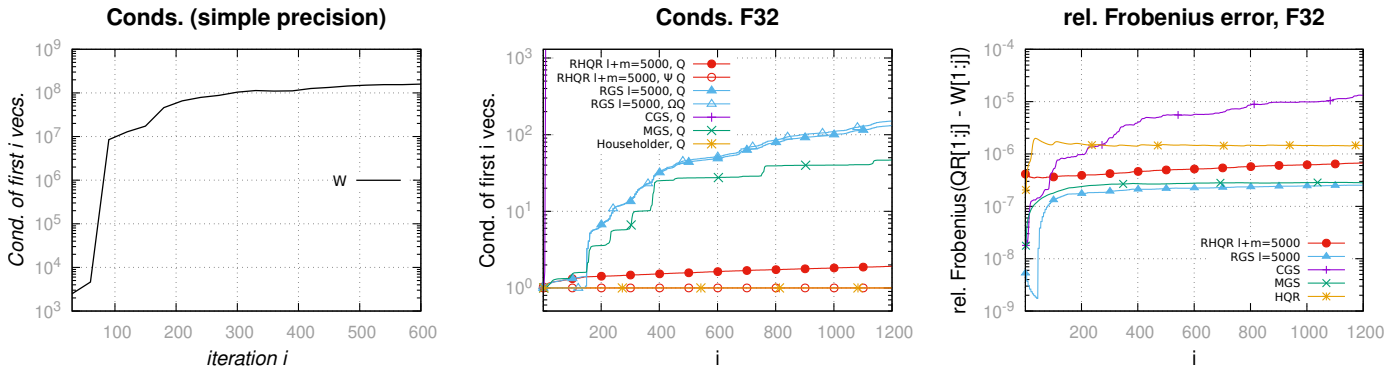
Figure 1: Randomized Householder QR in double precision (RHVector in the graphs) and comparison with RGS.

the condition number of the sketch of the Q factor is $1 + \mathcal{O}(\epsilon)$. The least squares problem that needs to be solved in RGS is computed using LAPACK's pivoted QR at each iteration.

Figure 1 displays the accuracy obtained in double precision for both RHQR (Algorithms 2 and 3) and RGS ([3, Algorithm 2]) , in terms of condition number of the computed basis and the accuracy of the obtained factorization. Figure 1a shows the condition number of W as the number of its vectors increases from 1 to 1500. The condition number grows exponentially and the matrix becomes numerically singular when approximately the 500-th vector is reached. Figure 1b displays in red (and circle points) the condition number of the basis and the sketched basis obtained through RHQR from Algorithm 3, and in blue (and triangle points) those obtained by RGS for reference. We observe that for similar sampling size, RGS and RHQR have similar performance. However, around the 500-th vector (which coincides with the moment the matrix becomes numerically singular), we see that RGS experiences some instabilities, while the behavior of RHQR does not change. We observe that the sketched basis of RHQR (a posteriori sketch of the output thin Q factor) remains numerically orthogonal. Figure 1c shows the relative Frobenius error of the QR factorization obtained by RHQR (in red) and RGS (in blue). Both algorithms produce accurate factorizations, with a slight advantage for RGS, but that becomes negligible when m grows.

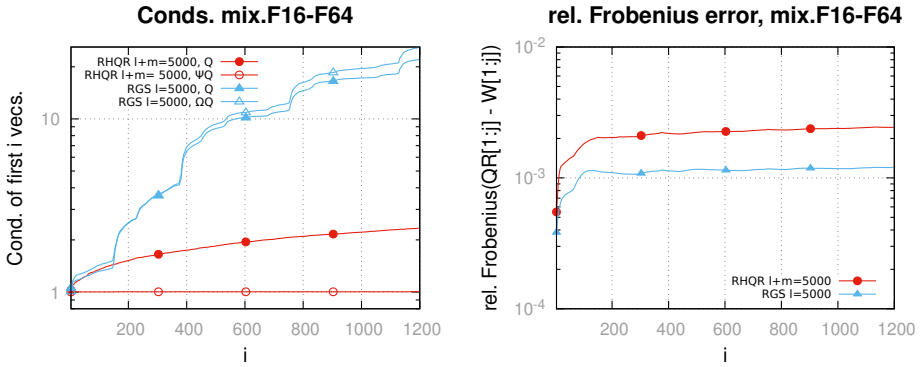
Figure 2 showcases the same experiments but in single precision. As to the comparison between RHQR and RGS, we make the same observations as in Figure 1, and we stress that the sketch of the RHQR basis remains numerically orthogonal. We also display the results obtained by Classical Gram-Schmidt (CGS), Modified Gram-Schmidt (MGS) and Householder QR (HQR). We can see that the basis computed by CGS is ill-conditioned very early in the iterations, yet the factorization remains accurate. The performance of RGS is similar to that of MGS, as pointed out in [3]. Finally, the basis produced by Householder QR is numerically orthogonal, just as the sketched basis output by RHQR, with an accurate factorization.

Figure 3 showcases the same experiment but in mixed precision. The input matrix $W = C_{1200}$ is given in half precision. All sketching operations are computed in half precision, and the sketches are



(a) Condition number of test matrix W (b) Condition number of bases output by RHQR and RGS (c) Factorization relative Frobenius error for RHQR and RGS

Figure 2: Randomized Householder QR in single precision (RHouse in the graphs) and comparison with RGS.

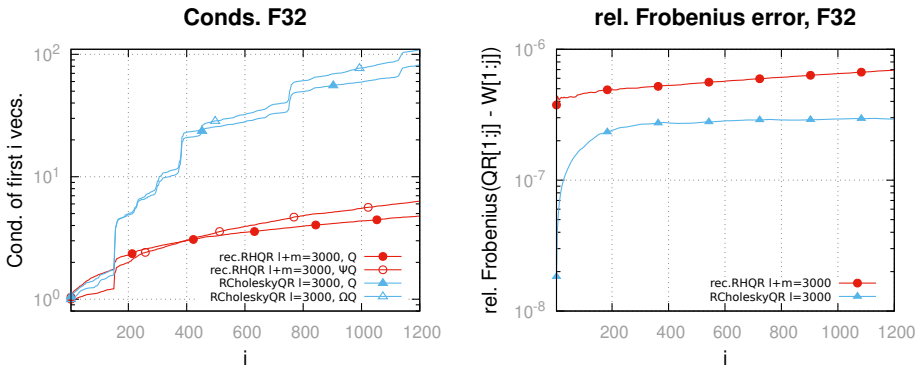


(a) Condition number of bases output by RHQR and RGS (b) Factorization relative Frobenius error for RHQR and RGS

Figure 3: Randomized Householder QR compared to RGS in mixed half/double precision.

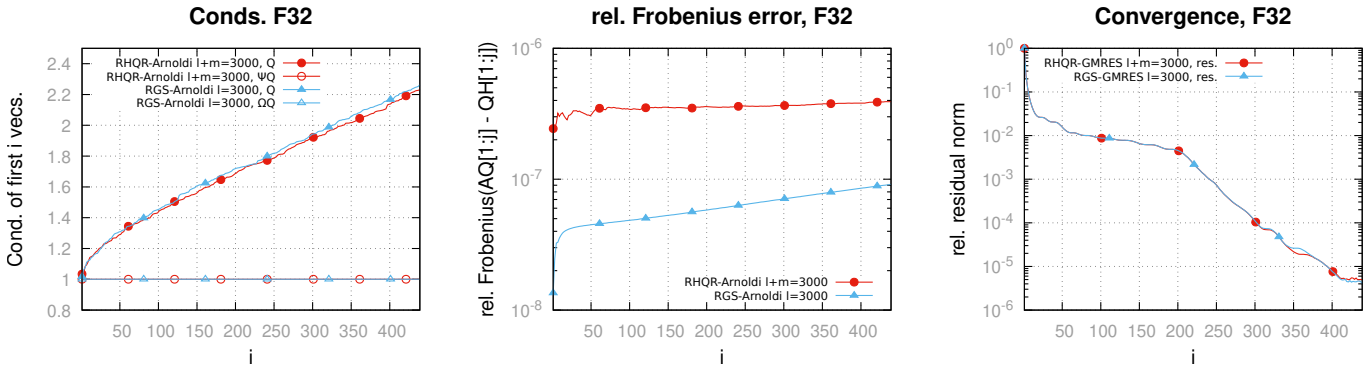
then converted to double precision. The computation of ρ_j in Algorithm 1 (RHVector) is then done in double precision. It is integrated to the sketch s_j in double precision, and to the randomized Householder vector u_j in half precision. Then in Algorithm 3, the matrix T_j is computed in double precision. In turn, the factor $T_{j-1}^t S_{j-1}^t z$ in Line 7 is computed in double precision, then converted in half precision to perform the update. We apply the same method to RGS: the input matrix is given in half precision, the sketching is made in half precision, the sketches are then converted in double precision. The least squares problem is computed in double precision. Its output is then converted back to half precision to perform the vector update. The result of the update is sketched again and converted to double precision. The norm of the sketch is computed in double precision, and the scaling of the obtained basis vector is performed in half precision. In [3], authors mixed single and double precision, whereas we mix half and double precision on a more difficult matrix. We make the same observations as in Figures 1 and 2, and stress again that the sketch of the basis output by RHQR remains numerically orthogonal.

Figure 4 displays the performance of reconstructRHQR in single precision on the matrix C_{1200}, and is compared to Randomized Cholesky QR since both algorithms perform a single synchronization. As detailed before, the reconstructRHQR is solved with a backward solve using the scaling coefficients $\alpha_1, \ldots, \alpha_m$ retrieved from the Householder QR factorization of the sketch ΨW. The Randomized Cholesky QR is performed with a backward solve of the R factor retrieved from the QR factorization of the sketch. In Figure 4a, we see that both bases output by reconstructRHQR and Randomized Cholesky QR lose their sketch orthogonality, but the former has a much more favorable trajectory than the latter.



(a) Condition number (b) Accuracy of the factorization

Figure 4: Performance of reconstructRHQR compared to Randomized Cholesky QR



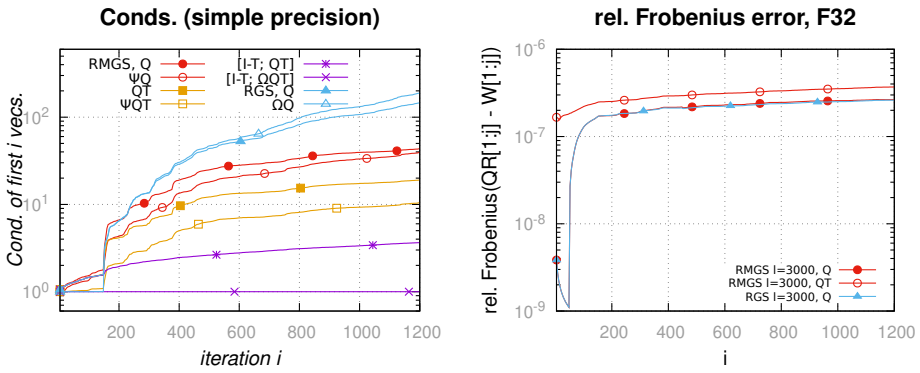
(a) Condition number Arnoldi basis (b) Accuracy of the Arnoldi relation (c) Convergence of GMRES

Figure 5: Performance of RHQR-GMRES compared to that of RGS-GMRES

In Figure 4b, we see that both algorithms compute an accurate factorization, with a slight advantage for Randomized Cholesky QR.

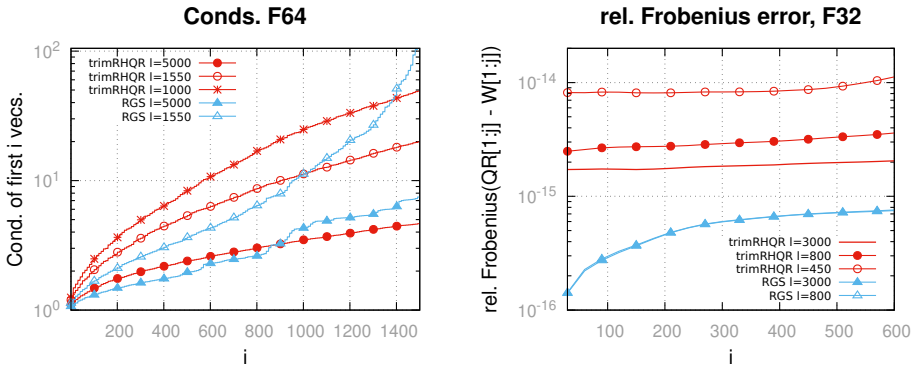
Figure 5 displays the performance of RHQR-GMRES (Algorithm 7) in single precision on the matrix *SiO2* from [8], compared to that of RGS-GMRES (based on [3, Algorithm 3]). The behavior of the two solvers is very similar. In Figure 5a, we display the condition number of $Q, \Psi Q$ (built by RHQR) and of $Q, \Omega Q$ (built by RGS). We see that both algorithms are able to produce a numerically orthogonal sketched Arnoldi basis. The condition number of the basis built by RHQR is slightly better than that built by RGS in the later iterations. In Figure 5b, we show the accuracy of the Arnoldi factorization $AQ_j = Q_{j+1}H_{j+1,j}$ computed by RHQR and RGS. As in the QR factorization of W, we see a slight advantage for RGS. Finally, in Figure 5c, we showcase the convergence of GMRES solvers with RHQR and RGS variations. In the early iterations, both solvers are indistinguishable. Before the residual stagnation, we see RHQR performing slightly better than RGS. When residual stagnation is reached, we see that RGS has a slight advantage over RHQR. On the more difficult *El3D* matrix, we observe the same results as in Figures 1 to 3, namely the numerical orthogonality of the sketched Arnoldi basis output by RHQR, the loss of orthogonality of that output by RGS, the accuracy of both factorizations with a slight advantage for RGS.

Figure 6 displays the performance of RMGS (Algorithm 8) in single precision, on the matrix C_{1200}. As pointed out in Section 5, the truly orthogonal matrix is $[I - T,\ \Omega QT]$, which in exact arithmetics is $[0_{m\times m};\ \Omega Q]$, i.e ΩQ would be orthogonal in exact arithmetics. In Figure 6a we showcase (in the order given by the plot's legend) the condition numbers of Q, ΩQ output by RMGS (i.e without any correction from T); the condition numbers of QT and ΩQT output by RMGS (a partial correction from T), the condition numbers of $[I - T;\ QT]$ and $[I - T;\ \Omega QT]$ output by RMGS (full correction from T), and



(a) Condition number (b) Accuracy of the factorization

Figure 6: Performance of RMGS compared to RGS



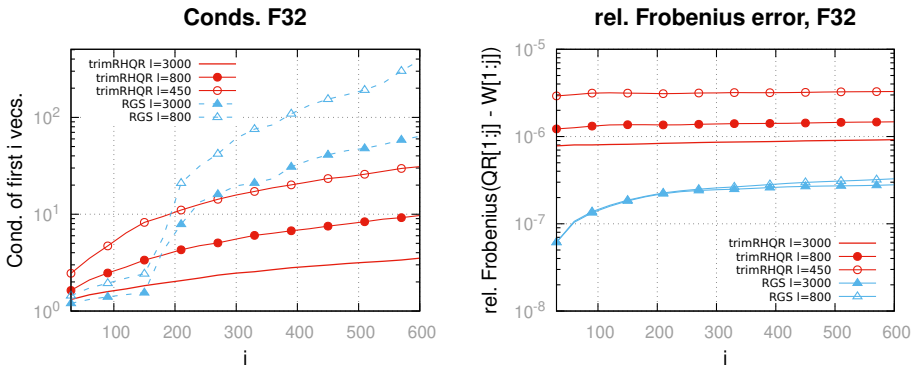
(a) Condition number of the bases output by trimRHQR and RGS (b) Factorization error of trimRHQR and RGS

Figure 7: Performance of trimRHQR and RGS in double precision

finally the comparison with the bases Q and ΩQ built by RGS. We see that the sketch of the basis built by RMGS with its full correction is numerically orthogonal. The basis built by RMGS, both with partial correction and without correction, suffers some instabilities when the matrix W becomes numerically singular, just like RGS. However, both bases built by RMGS are better conditioned than that built by RGS. In Figure 6b, we display the relative factorization error for the RMGS basis without correction, with partial correction, and we compare it to that of RGS. We see an interesting feature: while the basis with partial correction is the better conditioned of the three, the factorization is slightly les accurate than the one without correction. The factorization without correction is as accurate as that computed by RGS.

Figure 7 displays the accuracy obtained in double precision for both trimRHQR (Algorithms 10 and 11) and RGS, in terms of condition number of the computed basis and accuracy of the factorization, on the set of synthetic functions. Figure 7a displays in red (and circle points) the condition number of the basis obtained through RHQR from Algorithm 11, and in blue (and triangle points) that obtained by RGS for reference. In this experiment, the linear least squares problem of RGS is solved as before with LAPACK pivoted QR. We make the same global observations as in Figure 1. We note that the condition number of the basis output by trimRHQR is, in its stable regime, higher than that of both RHQR and RGS. Most importantly, we see that trimRHQR with a sampling size $\ell = 1000 < 1500 = m$ is stable, and stabler than RGS with a sampling size of $1550 > m$. We make the same observation in finite precision in Figure 8.

Figure 8 showcases the performance of Algorithm 11 in single precision. We make the same observations as in double precision (remark that trimRHQR with sampling size $\ell = 300 < 600$ outperforms



(a) Condition number of the bases output by trimRHQR and RGS

(b) Factorization error of trimRHQR and RGS

Figure 8: Performance of trimRHQR and RGS in single precision

RGS with $\ell = 800$).

8 Acknowledgements

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No 810367).

References

[1] Nir Ailon and Bernard Chazelle. "The fast Johnson–Lindenstrauss transform and approximate nearest neighbors". In: *SIAM Journal on computing* 39.1 (2009), pp. 302–322.

[2] Oleg Balabanov and Laura Grigori. "Randomized block Gram-Schmidt process for solution of linear systems and eigenvalue problems". In: *arXiv preprint arXiv:2111.14641* (2021).

[3] Oleg Balabanov and Laura Grigori. "Randomized Gram–Schmidt Process with Application to GMRES". In: *SIAM Journal on Scientific Computing* 44.3 (2022), A1450–A1474.

[4] Jesse L Barlow. "Block Modified Gram–Schmidt Algorithms and Their Analysis". In: *SIAM Journal on Matrix Analysis and Applications* 40.4 (2019), pp. 1257–1290.

[5] Åke Björck and Christopher C Paige. "Loss and recapture of orthogonality in the modified Gram–Schmidt algorithm". In: *SIAM journal on matrix analysis and applications* 13.1 (1992), pp. 176–190.

[6] Erin Carson, Kathryn Lund, Miroslav Rozložník, and Stephen Thomas. "Block Gram-Schmidt algorithms and their stability properties". In: *Linear Algebra and its Applications* 638 (2022), pp. 150–195.

[7] Anirban Dasgupta, Ravi Kumar, and Tamás Sarlós. "A sparse johnson: Lindenstrauss transform". In: *Proceedings of the forty-second ACM symposium on Theory of computing*. 2010, pp. 341–350.

[8] Timothy A Davis and Yifan Hu. "The University of Florida sparse matrix collection". In: *ACM Transactions on Mathematical Software (TOMS)* 38.1 (2011), pp. 1–25.

[9] James Demmel, Laura Grigori, Mark Hoemmen, and Julien Langou. "Communication-optimal parallel and sequential QR and LU factorizations". In: *SIAM Journal on Scientific Computing* 34.1 (2012), A206–A239.

[10] Nicholas J Higham. *Accuracy and stability of numerical algorithms*. SIAM, 2002.

[11] Christopher C Paige. "A useful form of unitary matrix obtained from any sequence of unit 2-norm n-vectors". In: *SIAM Journal on Matrix Analysis and Applications* 31.2 (2009), pp. 565–583.

[12] Chiara Puglisi. "Modification of the Householder method based on the compact WY representation". In: *SIAM Journal on Scientific and Statistical Computing* 13.3 (1992), pp. 723–726.

[13] Vladimir Rokhlin and Mark Tygert. "A fast randomized algorithm for overdetermined linear least-squares regression". In: *Proceedings of the National Academy of Sciences* 105.36 (2008), pp. 13212–13217.

[14] Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

[15] Robert Schreiber and Charles Van Loan. "A storage-efficient WY representation for products of Householder transformations". In: *SIAM Journal on Scientific and Statistical Computing* 10.1 (1989), pp. 53–57.

[16] JH Wilkinson. "The algebraic eigenvalue problem". In: *Handbook for Automatic Computation, Volume II, Linear Algebra*. Springer-Verlag New York, 1971.

[17] David P. Woodruff. "Sketching as a Tool for Numerical Linear Algebra". In: *CoRR* abs/1411.4357 (2014). arXiv: 1411.4357. URL: http://arxiv.org/abs/1411.4357.

Appendix : trimRHQR's compact formulas

Proof of Proposition 6.2 This is also proven by straightforward induction on j, however this one is somewhat trickier and we detail it here. For simplicity, we show the computation for the scaling where for all $j \in \{1, \cdots m\}$, $\|\Omega u_j\| = \sqrt{2}$. Also for simplicity, we will write $H_j := H(u_j, \Omega, j) = P(u_j, \Psi_j)$ for all $j \in \{1, \ldots, m\}$. For the case $m = 2$, multiplying H_1 and H_2, we observe that

$$H_1 H_2 = I_n - U_2 T_2 \mathrm{ut}\left((\Omega U_2)^t \Omega\right),$$

where ut denotes the upper triangular part and

$$T_2 = \begin{bmatrix} 1 & -\langle \Omega u_1, \Omega u_2 \rangle \\ 0 & 1 \end{bmatrix}.$$

Let us then suppose that

$$H_1 \cdots H_j = I - U_j T_j \mathrm{ut}\left((\Omega U_j)^t \Omega\right),$$

for some matrix $T_j \in \mathbb{R}^{j \times j}$. Multiplying by H_{j+1} on the right-side and using associativity, we obtain

$$\begin{aligned} &H_1 \cdots H_j H_{j+1} = \\ &I_n - u_{j+1}(\Omega u_{j+1})^t \begin{bmatrix} 0_{\ell \times j} & \Omega_{j+1} \end{bmatrix} - U_j T_j \mathrm{ut}\left((\Omega U_j)^t \Omega\right) + U_j T_j \mathrm{ut}\left((\Omega U_j)^t \Omega\right) u_{j+1}(\Omega u_{j+1})^t \begin{bmatrix} 0_{\ell \times j} & \Omega_{j+1} \end{bmatrix}. \end{aligned}$$

We first observe that

$$(\Omega u_{j+1})^t \begin{bmatrix} 0_j & \Omega_{j+1} \end{bmatrix} = \begin{bmatrix} 0_j & (\Omega u_{j+1})^t \Omega_{j+1} \end{bmatrix}.$$

Since the first j coordinates of u_{j+1} are zero, we also have:

$$\mathrm{ut}\left((\Omega U_j)^t \Omega\right) u_{j+1} = (\Omega U_j)^t \Omega u_{j+1},$$

so that the total composition writes

$$I_n - u_{j+1}\begin{bmatrix} 0_{\ell \times j} & (\Omega u_{j+1})^t \Omega_{j+1} \end{bmatrix} - U_j T_j \begin{bmatrix} (\Omega u_1)^t \Omega & & \\ 0_{1\times 1} & (\Omega u_2)^t \Omega_2 & \\ \vdots & & \\ 0_{1\times(j-1)} & & (\Omega u_j)^t \Omega_j \end{bmatrix} + U_j \left(T_j (\Omega U_j)^t \Omega u_{j+1}\right) \begin{bmatrix} 0_{1\times j} & (\Omega u_{j+1})^t \Omega_{j+1} \end{bmatrix}, \tag{39}$$

which can be factored as

$$H_1 \cdots H_j H_{j+1} = I - U_{j+1} \begin{bmatrix} T_j & -T_j(\Omega U_j)^t \Omega u_{j+1} \\ 0_{1\times j} & 1 \end{bmatrix} \mathrm{ut}\left((\Omega U_{j+1})^t \Omega\right).$$

The proposition follows by induction.

Proof of Proposition 6.5 : Apply the Woodburry-Morrison formula to $\mathcal{H}_j$ and $\mathcal{H}_j^{-1}$ and derive

$$\forall j \in \{1, \ldots, m\}, \quad \tilde{T}_j^t = -\left[I_j - T_j \mathrm{ut}\left((\Omega U_j)^t \Omega\right) U_j\right]^{-1} T_j. \tag{40}$$

Set

$$\Delta_j = (\Omega U_j)^t \Omega U_j - \mathrm{ut}\left((\Omega U_j)^t \Omega\right) U_j$$

then inject Δ_j in (40) to find

$$\tilde{T}_j^t = (T_j^{-t} - \Delta_j)^{-1}. \tag{41}$$

Finally, use (31).