ERROR ANALYSIS OF FLOATING-POINT ALGORITHMS TO COMPUTE INNER PRODUCTS AND THE QR FACTORIZATION, WITH AN APPLICATION TO LEAST SQUARE PROBLEMS*

YUELE HE†

Abstract. In the realm of computational linear algebra, understanding the behavior of rounding errors, the impact of random data, and the performance of algorithms to compute the QR decomposition are all crucial for enhancing numerical accuracy and algorithmic efficiency.

This investigation primarily addresses three core problems. Firstly, it explores the distribution and accumulation of rounding errors within a floating-point number system. Next, it considers how the distribution from which the entries of random vectors are sampled affects the relative error of their inner product. Lastly, it assesses the accuracy and algorithmic nuances of QR decomposition based on Householder reflection. The associated errors include the discrepancy between the product QR and the original matrix A, the distance from orthogonality of Q, and the residual error in least squares problems.

The research focuses on vectors and matrices whose elements are sampled from the uniform distribution. Varying sizes are considered, in order to understand size-dependent error patterns. Addressing floating-point rounding errors, this study establishes the relationship between relative error and single operation error. For inner product analysis, three error models are introduced, accounting for worst, probable, and mean-zero random vector scenarios. By employing logarithmic transformation for linear regression, it is observed that random vectors with non-zero mean yield error bounds proportional to square root of their length, whereas zero-mean vectors exhibit size-independent error bounds.

Regarding the QR decomposition, we found from numerical experiments, all three types of errors are proportional to $u(n)^a$, with a approximately ranging between 0.2-0.3. The block version of the algorithm exhibits proficiency in error types 2 and 3. While mixed precision mirrors the performance of low precision, it surpasses the latter in error type 1, with a slight error increase in larger sizes. This work thus provides comprehensive insights into the behavior of these linear algebra components under diverse conditions and precision considerations.

Key words. QR factorization, inner product, floating-point airhtmetic, rounding error, Householder reflector, mixed precision, block algorithm

AMS subject classifications. 65F25, 65F20

1. Introduction.

1.1. Rational for study. The widespread adoption of the floating-point number system has propelled computational capabilities to new heights. However, the representation of real numbers using floating-point numbers within computers introduces rounding errors that permeate numerical calculations [14]. Acknowledging these errors and their origins is essential for in-depth error analysis in numerical computations and the enhancement of numerical stability.

The inner product not only underpins matrix operations but also facilitates the transformation of linearly independent vectors into mutually orthogonal vectors. In the context of QR decomposition, pivotal in solving least squares problems, the orthogonal matrix is constructed from orthogonal Householder vectors, enhancing numerical stability and reducing error propagation when compared to conventional methods [1]. Moreover, QR decomposition finds application in principal component analysis, enabling the direct derivation of eigenvectors and eigenvalues from the decomposition [11].

1.2. Aims and Achievements. The primary aim of this paper is to dissect the influence of various factors on the computation error. Specifically, we investigate the effects of precision, the mean value of involved random variables, and the dimensions of vectors or matrices on the calculated error. This exploration encompasses three distinct types of errors: the error of a single operation in the floating-point system, the error introduced in vector inner products, and the error arising in QR decomposition.

For single operations, we discern distinct patterns in rounding errors across different ranges of random vector elements. Through relative error analysis, we pinpoint the factors leading to these observed trends. In the section on vector inner products, we unearth how the non zero mean values within vectors directly influences the relative error propagation.

In the section dedicated to QR decomposition, our analysis delves into the intricate interplay of varying precision levels, algorithmic approaches, and matrix dimensions. We found a trend that as matrix size escalates, error amplification surges more pronouncedly in the realm of high precision in contrast to both

*Work submitted as MiSCaDA dissertation in academic year 2022/2023.

†Durham University (yuele.he@durham.ac.uk)

mixed precision and low precision scenarios. The advantages of the block decomposition algorithm are as follows: the matrix Qobtains higher orthogenicity under low precision and mixed precision, and the residuals are smaller in the least squares problem.

The structure of the article is outlined as follows: We commence by providing an overview of the related work and foundational concepts in section 2. In section 3, we introduce two key variations of the QR factorization algorithm: the element version and the block version. Following the introductions, we conduct an analysis that encompasses error evaluation relevant to the QR decomposition process. In the realm of practical implementation, section 4 is dedicated to numerical computations involving inner products and QR decomposition. Leveraging the numpy library, we carry out these calculations and compare the obtained results with their analytical counterparts. Concluding our study in section 5, we consolidate our findings and insights from the preceding sections.

1.3. Notational Conventions. In this exposition, we adhere to certain notational conventions for clarity:

Lowercase letters, such as x, are employed to represent individual numbers or vectors. Uppercase letters, such as A, are employed to signify matrices. The symbols Q and R respectively denote orthogonal matrices and upper triangular matrices. If there is no hat mark above the letter, like x, it means the exact result, and if there is a hat mark, like $\hat{x}$, it means a computed result. Our discussions are grounded in floating-point arithmetic, in accordance with the latest revision [10] of the original IEEE 754-1985 Standard for Floating-Point Arithmetic [9]. Our indexing begins at 0, following the convention of the Python programming language. When addressing matrices, we follow the notation of the Python programming language. We use $i : j$ to denote the set of indices from i to j. For clarity, I would also add an example with rows an done with columns. For instance, $A[2 : 6, 1]$ denotes the sub-matrix of matrix A that contains the element of the column with index 1(second column) that belongs to rows with index 2 to 5 (third to sixth row).

2. Related work.

2.1. Floating point system. To represent a number y in a floating-point system $\mathcal{F}(\beta, t, e_{min}, e_{max})$, we can use the formula [13]

$$y = m \times \beta^{e-t+1}, \qquad \beta^{t-1} \leq m \leq \beta^{t-1}, \quad e_{min} \leq e \leq e_{max}, \tag{2.1}$$

where β is the base of the number system, m is the integral significand (with t significant digits), and e is the exponent. If a real number x cannot be represented exactly, then it has to be converted into a floating-point number. This conversion entails rounding the number to one of the two nearest floating-point numbers. The following shows that the relative error between a real number x and its floating-point approximation $fl(x)$ is bounded by the unit roundoff $u = \frac{1}{2}\beta^{1-t}$.

THEOREM 2.1 ([6, Theorem 2.2]). *Let $\mathcal{F}$ be a floating-point system, and let $fl : \mathbb{R} \to \mathcal{F}$ be the function that rounds a real to the closest floating-point number. If the real number x lies in the range of $\mathcal{F}$, then*

$$fl(x) = x(1 + \delta), \qquad |\delta| \leq u,$$

where δ is the rounding error and $u = \frac{1}{2}\delta^{1-t}$ is the unit roundoff.

Proof. Let x be a real number, so we can write

$$x = m \times \beta^{e-t+1}, \qquad \beta^{t-1} \leq m \leq \beta^t - 1.$$

We use $[m]$ to represent m rounded to its nearest integral number with significant digits t. Thus, the floating-point number closest to x is

$$fl(x) = [m] \times \beta^{e-t+1}, \qquad |m - [m]| \leq \frac{1}{2}.$$

The error between $|x|$ and its floating-point approximation $fl(x)$ can be written as

$$|x - fl(x)| \leq \frac{1}{2} \times \beta^{e-t+1}, \tag{2.2}$$

and dividing both sides by x, gives

$$\frac{|x - fl(x)|}{x} \leq \frac{\frac{1}{2} \times \beta^{e-t+1}}{m \times \beta^{e-t+1}} = \frac{1}{2m}.$$

Therefore, the floating-point approximation of x satisfies

$$fl(x) = x(1+\delta), \quad |\delta| \leq u. \tag{2.3}$$ ∎

2.2. Error in Floating-Point Operations. In a floating-point system $\mathcal{F}$, the result of an operation between two floating-point numbers x and y can be represented in two ways [6, Section 2.2].

STANDARD MODEL 1. *Standard model*

$$fl(x \ op \ y) = (x \ op \ y)(1+\delta) \ where \, |\delta| \leq u \ and \ op = +, -, \times, \div$$

STANDARD MODEL 2. *Alternative model*

$$fl(x \ op \ y) = \frac{(x \ op \ y)}{1+\delta} \ where \, |\delta| \leq u \ and \ op = +, -, \times, \div$$

Here, u is the unit roundoff, and x, y are elements of $\mathcal{F}$.

Standard Model 1 is widely used, especially in systems that adhere to the IEEE floating point standard [6, Section 2.2], as it provides accurate results for most calculations. These two models will be used in later proofs

2.3. Inner products in floating-point systems. Now let us consider how rounding errors affect the computation of the inner product of vectors in floating-point arithmetic. In this section, we will establish three types of rounding error bounds based on IEEE 754 standard for floating-point arithmetic. These bounds are defined as follows:

B1. Error Bound 1 [6]: The worst-case error bound is given by $\frac{nu}{1-nu}$. This error bound is applicable in all situations.

B2. Error Bound 2 [7]: The probabilistic bound is of the form $\lambda\sqrt{n}u+O(u^2)$ and holds with a probability of at least $P(\lambda) = 1 - 2\exp(-\frac{\lambda^2(1-u)^2}{2})$.

B3. Error Bound 3 [8]: The zero mean case is represented by u, which is independent of the size n. This bound holds when the elements have a zero mean.

THEOREM 2.2. *Consider the vectors* $x = [x_0, x_1, x_2, \ldots, x_{n-1}] \in \mathbb{R}^n$ *and* $y = [y_0, y_1, y_2, \ldots, y_{n-1}] \in \mathbb{R}^n$. *the computed inner products* $\hat{z}$ *satisfied* $\hat{z} = z(1+\theta_n)$, *where* $\theta_n \leq \frac{nu}{1-nu} =: \gamma_n$.

Let z be the inner product of x and y, defined by

$$z = \sum^{n} x_i y_i \tag{2.4}$$

For Standard Model 1, we have

$$fl(x_i y_i) = x_i y_i (1+\delta),$$

$$fl(x^T y) = x_1 y_1 \prod_{i=1}^{n}(1+\delta_{1i}) + \sum_{i=2}^{n}\left(x_i y_i \prod_{j=i}^{n+2}(1+\delta_{ij})\right), \quad \text{where } |\delta_{ij}| \leq u.$$

The error in the inner products can be represented as

$$\left|\frac{fl(xy) - xy}{xy}\right| \leq (1+u)^n.$$

Based on the binomial expansion, we have

$$(1+u)^n = \sum_{k=0}^{n} \binom{n}{k} u^k,$$

and the Taylor expansion gives

$$\frac{1}{1-nu} = \sum_{k=0}^{i} (nu)^k + O\left((nu)^i\right).$$

Since $\binom{n}{k} = \frac{n!}{k!(n-k)!} \leq n^k$, we have $(1+u)^n < \frac{1}{1-nu}$, and if $nu < 1$ the result for inner products in floating-point system can be written as

$$fl(z) = (1+\theta_n) \sum_{i+1}^{n} x_i y_i, \quad |\theta_n| \leq \frac{nu}{1-nu} =: \gamma_n.$$

Error in the inner products between vectors in a floating-point system can be represented as

$$\prod_{i=1}^{n} (1+\delta_i)^{\rho_i} = 1+\theta_n, \quad \theta_n \leq \gamma_n \tag{2.5}$$

where $|\delta_i| \leq u, \rho_i = \pm 1, nu < 1$, $\gamma_n := \frac{nu}{1-nu}$

THEOREM 2.3. *In a probabilistic sense, (2.5) can be represented as [7]*

$$\prod_{i=1}^{n} (1+\delta_i)^{\rho_i} = 1+\widetilde{\theta_n}, \qquad |\widetilde{\theta_n}| \leq \widetilde{\gamma_n}(\lambda), \tag{2.6}$$

where $\widetilde{\gamma_n}(\lambda) := \exp\left(\lambda\sqrt{n}u + \frac{nu^2}{1-u}\right) = \lambda\sqrt{n}u + O(u^2)$. *The probability of this inequality holding is at least* $p(\lambda) = 1 - 2\exp\left(-\frac{\lambda^2(1+u)^2}{2}\right)$.

Proof. Consider $\phi = \theta + 1 = \prod_{i=1}^{n}(1+\delta_i)^{\rho_i}$. Taking the logarithm of both sides gives

$$\log\theta = \sum_{i=1}^{n} \rho_i \log(1+\delta_i).$$

By utilizing the Taylor series expansions, we have:

$$\log(1+\delta_i) = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}\delta_i^k}{k},$$
$$\frac{1}{1-\delta_i} = \sum_{k=0}^{\infty} \delta_i^k,$$
$$\sum_{k=2}^{\infty} \delta_i^k = \frac{1}{1-\delta_i} - 1 - \delta_i = \frac{\delta_i^2}{1-\delta_i}.$$

When $\rho = 1$, we obtain,

$$\delta_i - \frac{\delta_i^2}{1-\delta_i} < \log(1+\delta_i) < \delta_i + \frac{\delta_i^2}{1-\delta_i},$$

and when $\rho = -1$, we have

$$-(\delta_i + \frac{\delta_i^2}{1-\delta_i}) < -\log(1+\delta_i) < -(\delta_i - \frac{\delta_i^2}{1-\delta_i})$$
$$-\delta_i - \frac{\delta_i^2}{1-\delta_i} < -\log(1+\delta_i) < -\delta_i + \frac{\delta_i^2}{1-\delta_i}$$

Combining these results, we conclude:

$$\rho_i\delta_i - \frac{\delta_i^2}{1-\delta_i} < \rho_i \log(1+\delta_i) < \rho_i\delta_i + \frac{\delta_i^2}{1-\delta_i},$$

where $\rho_i = \pm 1$.

Taking the absolute value, we have:

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

Now we invoke Hoeffding's inequality, which we briefly recall. Let $X_0, X_1, \ldots, X_{n-1}$ be independent random variables such that $a_i \leq X_i \leq b_i$ almost surely. Consider the sum of these random variables,

$$S_n = X_0, X_1, \ldots, X_{n-1}.$$

Hoeffding's theorem [15] states that, for all $t > 0$, one has

$$\Pr\left(S_n - \mathrm{E}[S_n] \geq t\right) \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right),$$
$$\Pr\left(|S_n - \mathrm{E}[S_n]| \geq t\right) \leq 2\exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).$$

where $E[S_n]$ is the expected value of S_n.

If $x_k = \rho_k \log(1+\delta_k)$, then $|x_k - x_{k_1}| \leq \frac{u^2}{1-u}$. Applying Hoeffding's inequality, we obtain:

$$\Pr(|\log\phi - E[\log\phi]| \geq t) \leq 2\exp\left(-\frac{t^2(1-u)^2}{2nu^2}\right).$$

Furthermore, simplifying the expression, we find

$$\begin{aligned} |E[\log\phi]| &= \left|E\left[\sum(\rho_i \log(1+\delta_i))\right]\right| \\ &\leq \sum E\left[(\rho_i \log(1+\delta_i))\right] \\ &\leq \sum_{n}^{} \frac{u^2}{1-u} = \frac{nu^2}{1-u}. \end{aligned}$$

Hence, we can derive

$$\begin{aligned} |\log\phi - E[\log\phi]| &\geq |\log\phi| - |E[\log\phi]| \\ &\geq |\log\phi| - \frac{nu^2}{1-u}. \end{aligned}$$

This leads to the inequality

$$\begin{aligned} \Pr\left(|\log\phi| - \frac{nu^2}{1-u} \geq t\right) &\leq \Pr\left(|\log\phi - E[\log\phi]| \geq t\right) \\ &\leq 2\exp\left(-\frac{t^2(1-u)^2}{2nu^2}\right). \end{aligned}$$

Consequently, we have

$$\Pr\left(|\log\phi| \leq t + \frac{nu^2}{1-u}\right) \geq 1 - 2\exp\left(-\frac{t^2(1-u)^2}{2nu^2}\right),$$

and if we set $t = \lambda\sqrt{n}u$, then we obtain

$$\Pr\left(|\log\phi| \leq \lambda\sqrt{n}u + \frac{nu^2}{1-u}\right) \geq 1 - 2\exp\left(-\frac{\lambda^2(1-u)^2}{2}\right).$$

This inequality can be further simplified to

$$|\log\phi| \le \lambda\sqrt{n}u + \frac{nu}{1-u},$$

which implies

$$\begin{aligned}
\exp(-\lambda\sqrt{n}u - \frac{nu}{1-u}) \le \phi \le \exp(\lambda\sqrt{n}u + \frac{nu}{1-u}),\\
\exp(-\lambda\sqrt{n}u - \frac{nu}{1-u}) - 1 \le \phi - 1 \le \exp(\lambda\sqrt{n}u + \frac{nu}{1-u}) - 1,\\
\exp(-\lambda\sqrt{n}u - \frac{nu}{1-u}) - 1 \le \theta \le \exp(\lambda\sqrt{n}u + \frac{nu}{1-u}) - 1.
\end{aligned}$$

Taking the absolute value, for $0 \le \lambda\sqrt{n}u + \frac{nu}{1-u}$, we have

$$\begin{aligned}
|\exp(-\lambda\sqrt{n}u - \frac{nu}{1-u}) - 1| &= 1 - \exp(-\lambda\sqrt{n}u - \frac{nu}{1-u})\\
&= (\exp(\lambda\sqrt{n}u + \frac{nu}{1-u}) - 1)\exp(-\lambda\sqrt{n}u - \frac{nu}{1-u})\\
&\le \exp(\lambda\sqrt{n}u + \frac{nu}{1-u}) - 1
\end{aligned}$$

Taking the absolute value, we have:

$$|\theta| \le \exp(\lambda\sqrt{n}u + \frac{nu}{1-u}) - 1 = \widetilde{\gamma_n}(\lambda)$$

∎

LEMMA 2.4. *Consider an element-wise independent random variable vector $x \in \mathbb{R}^n$, where each component x_j is drawn from a given distribution with a mean of μ_x and is bounded by $|x_j| \le C_x$, where C_x is a constant. Let $s = \sum_{j=1}^n x_j$, and let $\hat{s}$ be computed using recursive summation. The rounding errors δ_i are random variables with a mean of zero. For all index k, the δ_k are mean-independent of the preceding rounding errors and the data. This implies that $E(\delta_k|\delta_2,\ldots,\delta_{k-1},x_0,\ldots,x_{n-1}) = E[\delta_k] = 0$. For a given positive integer n, if there exists $\alpha \in [0,1]$ such that $(1-\alpha)\mu_{|x|}\sqrt{n} \ge \lambda C_x$, the backward error bound for $\hat{s}$ can be bounded as $\epsilon_{bwd}(\hat{s}) \le 1/(\alpha\mu_{|x|})$. [8, Corollary 2.10]*

In accordance with the implications of Lemma 2.4, when the mean μ_x is non-zero, the increase in backward error for the inner product is directly proportional to $\sqrt{n}$ times u. Conversely, when $\mu_x = 0$, the backward error for the inner product is approximately equal to the magnitude of the rounding error u, independent of the vector length n.

2.4. Householder Reflection: Algorithm and Geometric Interpretation. Given a vector $v \in \mathbb{R}^m$, a Householder reflection is defined as [4]

$$P(v) = I - \beta vv^T, \qquad \beta = \frac{2}{v^Tv}, \tag{2.7}$$

where I represents the identity matrix. The vector v is usually called the Householder vector.

From a geometric perspective, when a vector $x = \alpha v + \beta v_\perp$, where αv represents the component parallel to the Householder vector, βv represents the component perpendicular to it, and α and β as scaling factors. Upon subjecting x to a Householder transformation, the resultant reflected vector y is given by

$$\begin{aligned}
y &= P(v)x\\
&= x - \beta vv^Tx\\
&= \alpha v + \beta v_\perp - \frac{2}{v^Tv}vv^T\alpha v - \frac{2}{v^Tv}vv^T\beta v_\perp\\
&= \alpha v + \beta v_\perp - 2e_1v\\
&= -\alpha v + \beta v_\perp.
\end{aligned}$$

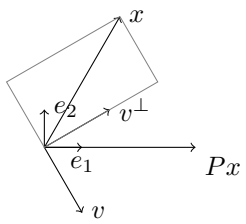


Fig. 1: Geometric Illustration of Vector Householder Reflection

Note that in this transformation, the component parallel to the Householder vector αv experiences a reversal, whereas the component perpendicular to the Householder vector $\beta v_\perp$ remains unchanged.

Using [4, Algorithm 5.1.1] [2], given a vector $x \in \mathbb{R}^m$, we can obtain a Householder vector $v \in \mathbb{R}^m$ and a coefficient $\beta \in \mathbb{R}$, such that the orthogonal matrix $P = I_m - \beta v v^T$ satisfies $Px = \|x\|_2 e_0$, where e_0 is the first basis vector.

Algorithm 2.1 Householder Transformation

function $[v, \beta] = House(x)$
$m = \text{length}(x)$
$\sigma = x[1:m]^T x[1:m]$
$v = [0, x[1:m]^T]$
if $\sigma = 0$ and $x[0] \geq 0$ **then**
  $\beta = 0$
else if $\sigma = 0$ and $x[0] < 0$ **then**
  $\beta = -2$
else
  $\mu = \sqrt{x(0)^2 + \sigma}$
  if $x(1) \leq 0$ **then**
    $v[0] = x[0] - \mu$
  else
    $v[0] = -\sigma/(x[0] + \mu)$
  end if
  $\beta = \frac{2v[0]^2}{\sigma + v[0]^2}$
  $v = v/v[0]$
end if
return v, β

From the perspective of matrices and vectors, applying a Householder matrix to the vector x is equivalent to representing x in the first vector of the canonical basis e_0, which means that all entries of the vector are set to zero except for the first.

Example: Vector Householder Reflection Visualization
Let $x = [2, 2\sqrt{3}]$. Using Algorithm 2.1, we have the Householder vector $v = [1, -\sqrt{3}]$, and the corresponding Householder matrix is $\begin{bmatrix} \frac{1}{2} & \frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & -\frac{1}{2}, \end{bmatrix}$ which gives $Px = [4, 0]$, as visually demonstrated in Figure 1.

2.5. Householder QR Factorization and Orthogonal Matrix Decomposition. An Householder matrix is orthogonal, since

$$\begin{aligned} PP^T &= (I - \beta vv^T)(I - \beta vv^T)^T \\ &= (I - \beta vv^T)(I - \beta vv^T) \\ &= I - \beta vv^T - \beta vv^T + \beta^2 vv^T vv^T \\ &= I - 2\beta vv^T + 2\beta vv^T = I, \end{aligned}$$

where in the last step we used the fact that $\beta = \frac{2}{v^T v}$.

Furthermore, we can show that any matrix A can be decomposed into the product of an orthogonal matrix Q and an upper triangular matrix R. This theorem is particularly significant, as it provides a versatile approach for solving the least squares problem while maintaining both numerical stability and computational efficiency.

THEOREM 2.5. *[4, Theorem 5.2.1] Let matrix $A \in \mathbb{R}^{m\times n}$. There exists an orthogonal matrix $Q \in \mathbb{R}^{m\times m}$ and an upper triangular matrix $R \in \mathbb{R}^{m\times n}$ such that $A = QR$.*

Proof. If $n = 1$, the matrix A is a vector. After Householder reflection, $PA = \|A\|_2 e_1$, so $A = P^T\|A\|_2 e_1$, where $e_1 = [1, 0, \ldots, 0]^T$ is the first vector of the canonical basis of $\mathbb{R}^n$. If $n > 1$, we can partition the matrix as

$$A = \left[\begin{array}{ccc|c} a_{11} & \cdots & & a_{1n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & \cdots & & a_{mn} \end{array}\right] = [A_1|v],$$

where A_1 can be written as $A_1 = Q_1 R_1$ by induction and $v = A[:, n]$.

Therefore, we can write

$$Q_1^T A = [Q_1^T A_1|Q_1^T v] = \left[\begin{array}{ccc|c} \sum_j q_{1j}a_{j1} & \cdots & & \sum_j q_{1j}a_{jn} \\ & \ddots & & \vdots \\ & & & \sum_j q_{mj}a_{jn} \end{array}\right] = [R_1|w] = \left[\begin{array}{c|c} R_1' & w_1 \\ 0 & w_2 \end{array}\right],$$

where R_1 is a square upper triangular matrix, $w_1 \in \mathbb{R}^n$ and $w_2 \in \mathbb{R}^{m-n}$. For the sub-column w_2, we can find a Householder matrix P, such that $P^T w_2 = \|w_2\|_2 e_1$. If we let $Q_2 = \begin{bmatrix} I & 0 \\ 0 & P_2 \end{bmatrix}$, we have

$$Q_2 Q_1^T A = \begin{bmatrix} I & 0 \\ 0 & P \end{bmatrix} \left[\begin{array}{c|c} R_1' & w_1 \\ 0 & w_2 \end{array}\right] = \begin{bmatrix} R_1' & w_1 \\ 0 & Pcw_2 \end{bmatrix} = R.$$

In the context of the Householder vector-based QR decomposition, utilizing the product of the matrices W and Y as a representation for the orthogonal matrix Q offers the advantage of simplifying calculations and improving numerical stability. The procedure for computing the matrices W and $Y$is outlined in Algorithm 2.1, from the reference [4].

Consider the factorization $Q = Q_1 \cdots Q_r$, where the matrices $Q_j = I_m - \beta_j v^j (v^j)^T$ are stored in factored form. This algorithm facilitates the computation of matrices $W \in \mathbb{R}^{m\times r}$ and $Y \in \mathbb{R}^{m\times r}$, which fulfill the relationship $Q = I_m - WY$.

2.6. LS problem. Now we consider the least-squares (LS) problem

$$\min_{x\in R^n} \|Ax - b\|_2. \tag{2.8}$$

If A is full rank, then x is a solution to (2.8) if and only if it satisfies the normal equations

$$\nabla\|Ax - b\|_2^2 = 2A^T(Ax - b) = 0, \tag{2.9}$$

as we now show.

Algorithm 2.2 Computation of matrix W, Y

$Y = v^1$
$W = \beta_1 v^1$
for j = 1:r **do**
  $z = \beta_j(I_m - WY^T)v^j$
  $W = [W|z]$
  $Y = [Y|v^j]$
end for
return W, Y ∎

Proof. If x did not satisfy the equation (2.8), then we could find a small enough constant α and a vector $z = A^T(Ax-b) = 0$ such that $\|A(x+\alpha z)-b)\|_2 < \|Ax-b\|_2$. If x satisfies the equation (2.8), then x is the minimum of $\|Ax-b\|_2$. In this case, the gradient at x must be zero, and we have

$$2A^TAx - 2A^Tb = 0 \Leftrightarrow A^TAx = A^Tb$$

If A is full rank, then A^TA is nonsingular, and we can find x as

$$x = (A^TA)^{-1}A^Tb.$$

Thus, we have shown that the solution x to (2.8) satisfies the normal equation $A^T(Ax-b) = 0$.

Conversely, assume that x is a solution to the normal equations (2.9). If x were not a minimizer for (2.8), there would exist a small enough constant α and a vector $z = A^T(Ax-b)$ such that $\|A(x+\alpha z)-b\|_2^2 < \|Ax-b\|_2^2$. We have that

$$\begin{aligned} \|A(x+\alpha z)-b\|_2^2 &= (A(x+\alpha z)-b))^T(A(x+\alpha z)-b)) \\ &= ((Ax-b)^T + (\alpha Az)^T)(Ax-b+\alpha Az) \\ &= \|Ax-b\|_2 + 2\alpha(Ax-b)^TAz + \alpha^2\|Az\|_2 \end{aligned} \tag{2.10}$$

Now let $z = -A^T(Ax-b)$. The second term $2\alpha(Ax-b)^TAz$ can be written as

$$\begin{aligned} 2\alpha(Ax-b)^TAz &= -2\alpha(Ax-b)^TAA^T(Ax-b) \\ &= -2\alpha\|A^T(Ax-b)\|_2 \end{aligned}$$

If the inequality $\|A(x+\alpha z)-b\|_2^2 < \|Ax-b\|_2^2$ were true, we would have that $2\alpha(Ax-b)^TAz+\alpha^2\|Az\|_2 < 0$. or equivalently, that $0 < \alpha < \frac{2\|A^T(Ax-b)\|_2^2}{\|AA^T(Ax-b)\|_2^2}$. This demonstrates that for a small enough positive constant α, and a vector $z = A^T(Ax-b)$, the inequality $\|A(x+\alpha z)-b)\|_2^2 < \|Ax-b\|_2^2$ holds when x does not satisfy the normal equation (2.9). ∎

If x and $x+\alpha z$ are both LS minimizers, where z is any vector, then $z \in \text{null}\, A$.

Proof. Based on the equation(2.10) and (2.8), the gradient of $\|A(x+\alpha z-b)\|_2^2$ can be written as

$$\nabla\|A(x+\alpha z)-b\|_2^2 = A^T(A(x+\alpha z)-b) = A^TAx - b + \alpha A^TAz = \alpha A^TAz = 0$$

Since $\alpha A^TAz = 0$, $z^TA^TAz = 0$ also is true for any vector z. $z^TA^TAz = \|Az\|_2$, and the vector norm Az is zero if and only if the vector Az is a zero vector, that is, $z \in \text{null}\, A$. ∎

3. Householder QR Factorization.

3.1. Algorithms at the element level. At the element level, the QR factorization is performed using traditional element-wise operations. In this approach, each element of the input matrix is considered independently, and Householder reflections are utilized to obtain the Q and R matrices. In order to calculate the QR factorization, we utilize Algorithm 2.1 (Householder Vector) and Algorithm 2.2 (W, Y matrix) from [12], followed by Algorithm 3.1 (Householder QR) for the element-wise computations. In this algorithm, we

Algorithm 3.1 QR Factorization

$m, n \leftarrow$ shape of A
$\bar{A} \leftarrow$ copy of A
$Q \leftarrow I_m$
for $j \leftarrow 0$ to $n-1$ **do**
  $v \leftarrow \bar{A}[j:, j]$
  $\hat{v}, \beta \leftarrow$ HouseholderVector(v)
  Update R: $\bar{A}[j:, j:] \leftarrow \bar{A}[j:, j:] - \beta \cdot \hat{v} \cdot \hat{v}^T \cdot \bar{A}[j:, j:]$
  if $j = 0$ **then**
    $Y \leftarrow \hat{v}$ (reshaped to a column vector)
    $W \leftarrow \beta \cdot \hat{v}$ (reshaped to a column vector)
  else
    $\hat{v} \leftarrow$ concatenate(zeros$(j), \hat{v})$ (reshaped to a column vector)
    $z \leftarrow \beta \cdot \hat{v} - \beta \cdot W \cdot Y^T \cdot \hat{v}$
    $W \leftarrow$ concatenate(W, z) (along the columns)
    $Y \leftarrow$ concatenate(Y, v_{hat}) (along the columns)
  end if
end for
$R \leftarrow$ upper triangular part of $\bar{A}$
$Q \leftarrow I_m - W \cdot Y^T$
return Q, R

denote the identity matrix of size m by I_m, and we denote the overwritten matrix $A_{overwritten}$ as $\bar{A}$. In practical computations, for memory conservation, $\bar{A}$ can be stored in A, if the latter is not needed once the QR factorization has been computed.

Consider a matrix $A \in \mathbb{R}^{m\times n}$, where $m \geq n$. We maintain a list all β values and $A_{overwritten}$ which represents the triangular matrix R and Householder vector v. With these preparations, we now have two distinct approaches to compute the orthogonal matrix Q: the direct calculation method and the WY matrix method.

In the direct calculation approach, we generate Householder matrices iteratively as subsection 2.4. In WY matrix method involves constructing a WY matrix and deriving the Householder matrix through $Q = I - WY^T$. Next, the FLOPs required for both methods would be analyzed. For the direct calculation, the FLOPs in constructing Householder Matrix H_j by the formula (2.7) are $2(m-j)^2$. In the products of H_j and H_{j+1} Proof 3, the calculation contribute $2(m-j)^2 + 2(m-j)^2$ FLOPs. The total FLOPs for the matrix product in the expression are given by $\sum_{j=1}^{n} 4(m-j)^2 + 2(m-j)^3$. Simplifying this expression by retaining only the term containing the highest order of m, we get $\sum_{j=1}^{n} 2m^3 = 2m^3 n$.

In the WY Matrix approach, We initiate our process by constructing the WY matrix, which is defined in Algorithm 2.2. The central computation revolves around the expression:

$$z = \beta_j (I_m - WY^T) v^{(j)} = \beta_j \hat{v} - \beta_j W (Y^T \hat{v})$$

When evaluating the vector z, the index j from 1 to $n-1$, and the FLOPs incurred at each iteration are $m - j + j^2(m-j) + (m-j+1)$. The cumulative FLOPs across all steps can be computed using:

$$\begin{aligned}\sum_{j=1}^{n-1} m - j + j^2(m-j) + (m-j+1) &= \sum_{j=1}^{n-1} 2m - 2j + 1 + j^2 m - j^3 \\ &\simeq 2mn - 2\frac{n^2}{2} + n + \frac{n^2 m}{3} - \frac{n^4}{4}\end{aligned}$$

Additionally, we proceed to the final step $Q = I_m - WY$, and the FLOPs are $m^2 n + mn$. The term containing the highest order of m is $\frac{m^2 n}{3}$, which is lower than the direct calculation method.

3.2. Algorithms at the block level. At the element level, the QR factorization is performed using traditional element-wise operations. In this approach, each element of the input matrix is considered

independently, and Householder creflections are utilized to obtain the Q and R matrices.

To facilitate parallel computations, we utilize the T4 GPU available in Google Colab to implement the QR factorization algorithm. This enables us to take advantage of the parallel processing capabilities of the GPU, thereby enhancing the computational efficiency of the QR factorization process.

To elucidate the process of performing QR decomposition using a partitioned matrix approach, let's explain how to implement parallel computation of block-wise QR decomposition.

For an arbitrary matrix A, it can be convenienty represented through partitioning as:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}.$$

To initiate the QR decomposition, we start with the matrix $\begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix}$, and in accordance with Theorem 2.5, we can identify an orthogonal matrix $Q^{(1)}$ that satisfied the equation:

$$Q^{(1)T} \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix} = \begin{bmatrix} R^{(1)} \\ 0 \end{bmatrix}.$$

Hence, we have:

$$Q^{(1)T} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} R^{(1)} & A_{12}^{(1)} \\ 0 & A_{22}^{(1)} \end{bmatrix}.$$

Subsequently, we determine $Q^{(2)}$ such that it satisfies $Q^{(2)T} A_{22}^{(1)} = R^{(2)}$. This step conclude the block QR factorization of matrix A, i.e. $A = QR$, where:

$$Q = \begin{bmatrix} Q_1^{(1)} & Q_2^{(1)} \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & Q^{(2)} \end{bmatrix} = \begin{bmatrix} Q_1^{(1)} & Q_2^{(1)} Q^{(2)} \end{bmatrix}$$

and

$$R = \begin{bmatrix} R^{(1)} & A_{12}^{(1)} \\ 0 & R^{(2)} \end{bmatrix}.$$

Thus, the block QR factorization involves a stepwise process. We initially partition the first column of the partitioning matrix using QR decomposition to acquire $(Q^{(1)})$. We then update matrix A using $(Q^{(1)T} A)$. Subsequently, we perform a QR factorization of the updated partitioned matrix, excluding the first row and column, and continue updating Q and A in the same manner. This process is repeated iteratively until matrix A transforms into a triangular form. We have observed that the updates to matrices Q and R are independent processes, suggesting the possibility of concurrent updates to enhance computational efficiency. To visualize this parallelization, we utilize the illustrative representation provided in Figure 2, which illustrates the process of parallelizing the block-by-block QR decomposition algorithm. The algorithm's steps are depicted in a flowchart, highlighting the simultaneous execution of two distinct parts. The Light Yellow arrows symbolize the progress of process 1, which involves updating the R matrix. Concurrently, the Pale Peach arrows indicate the advancement of process 2, responsible for updating the Q matrix. For detailed procedural insights into the block QR factorization with parallel updates, we refer to Algorithm Algorithm 3.2. This approach capitalizes on the inherent parallelism within the QR factorization process, ultimately accelerating the overall computation.

3.3. Matrix Reconstruction Error. One of the key measures of the QR factorization accuracy is the error of reconstructing the original matrix from the computed Q and R matrices. The error $\|A - QR\|_F$ is computed as the Euclidean norm of the difference between A and the product of Q and R.

This analysis aims to elucidate the error accumulation behavior inherent in the update process of the triangular matrix R and orthogonal matrix Q using the QR factorization algorithm. We use the error bound presented in Item B2. to represent errors. In this section, the variable $\delta(|\delta| \leq u)$ is utilized to denote the rounding error resulting from a single operation, where u represents the level of precision. Additionally, error in inner products of a vector with k elements are donated as $\theta_k(|\theta_k| \leq \gamma_k)$.

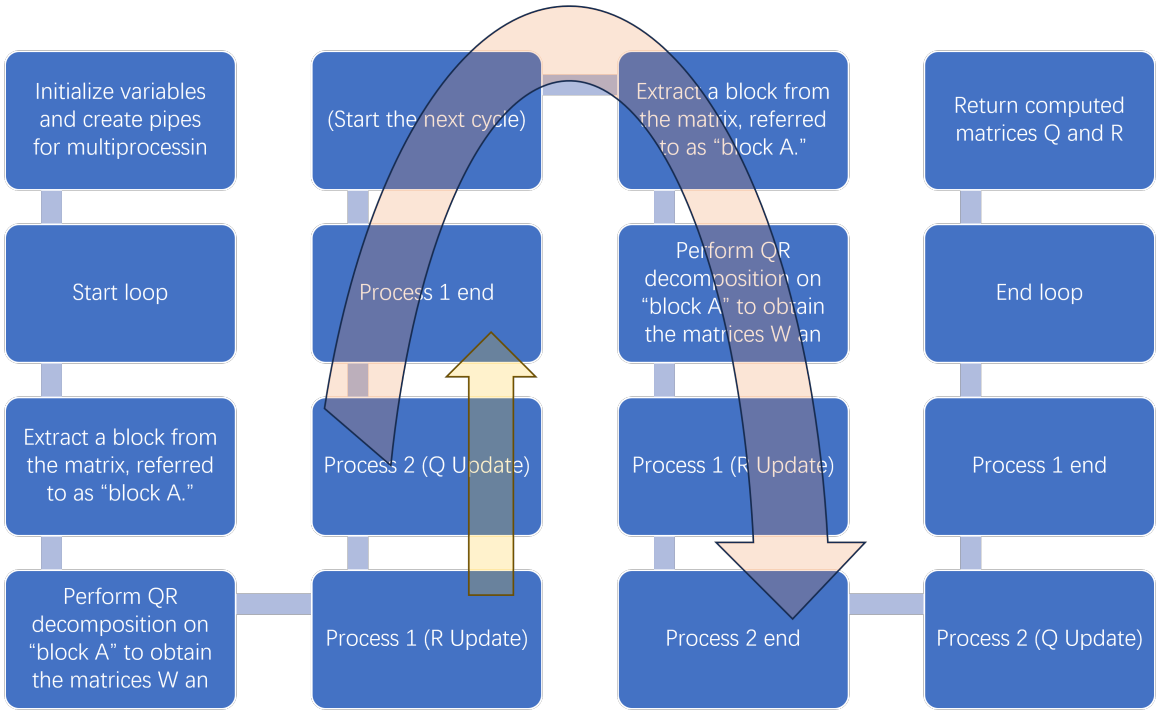


Fig. 2: Parallel Block-by-Block QR Decomposition Process

Algorithm 3.2 Block Householder Reflection

$m, n \leftarrow$ shape of A
$Q_{\text{block}} \leftarrow$ identity matrix of size $m \times m$
$R_{\text{block}} \leftarrow$ copy of matrix A
$\lambda_{\text{val}} \leftarrow 0$
$r \leftarrow \sqrt{m}$
while $\lambda_{\text{val}} < n$ **do**
  $\tau \leftarrow \min(\lambda_{\text{val}} + r, n)$
  $A_{\text{block}} \leftarrow R_{\text{block}}[\lambda_{\text{val}} : m, \lambda_{\text{val}} : \tau]$
  $\bar{A}, \beta_{\text{list}} \leftarrow$ qr_factorization_overwritten(A_{block}, ty)
  if $\lambda_{\text{val}} \neq 0$ **then**
    process2.join()
  end if
  $W, Y \leftarrow$ WY_matrix$(\bar{A}, \beta_{\text{list}}, ty)$
  process1 $\leftarrow$ updatedR$(R_{\text{block}}[\lambda_{\text{val}} : m, \tau - r : n], W, Y,)$
  process2 $\leftarrow$ updatedQ$(Q_{\text{block}}[:, \lambda_{\text{val}} : m], W, Y,)$
  process1.start()
  process2.start()
  $R_{\text{block}}[\lambda_{\text{val}} : m, \tau - r : n] \leftarrow$ process1 result
  $Q_{\text{block}}[:, \lambda_{\text{val}} : m] \leftarrow$ process2 result
  process1.join()
  $\lambda_{\text{val}} \leftarrow \tau$
end while
$R_{\text{block}} \leftarrow$ upper_triangular(R_{block})

return $Q_{\text{block}}, R_{\text{block}}$

As shown in Lemma 2.2 from [16], the rules summarize how to accumulate error represented by θ and γ in a uniform precision setting. arithmetis operations between bounded terms, θ_k are:

$$(1+\theta_k)(1+\theta_j) = (1+\theta_{k+j}) \qquad and \qquad \frac{1+\theta_k}{1+\theta_j} = \begin{cases} 1+\theta_{k+j}, & j \le k \\ 1+\theta_{k+2j}, & j > k \end{cases}$$

if $\max_{j,k} u \le \frac{1}{2}$ and $n \le \frac{1}{uk}$, the operations on the bounds γ are:

$$\begin{aligned} \gamma_k\gamma_j &\le \gamma_{\min(k,j)}, \qquad & n\gamma_k &\le \gamma_{nk]}, \\ \gamma_k + u &\le \gamma_{k+1}, \qquad & \gamma_k + \gamma_j + \gamma_k\gamma_j &\le \gamma_{k+j]}, \end{aligned}$$

where $|\theta_k| \le \gamma_k$, and $\gamma_n = \exp(\lambda\sqrt{n}u + \frac{nu}{1-u}) - 1 = \lambda\sqrt{n}u + O(u^2)$

The process uses the function $House(x)$ to calculate the Householder vector, denoted as v, and the parameter β as outlined in Algorithm 2.1. In this analysis, we assume that all elements of vector x are non-zero. Initially, we set v equal to x. Then, we calculate σ as $x[1:m]^T x[1:m]$, representing its computational result as $\hat{\sigma}(1+\theta^{(\sigma)})$. We use the same notation to represent the computational result in subsequent calculations. Similarly, we compute μ as $x^T x$. The calculations of both σ and μ involve inner products of vectors, so we have:

$$\begin{aligned} \hat{v}[0] &= -\frac{\hat{\sigma}}{(x[0]+\hat{\mu})(1+\delta_1)}(1+\delta_2) \\ &= -\frac{\sigma(1+\theta^{(\sigma)})}{(x[0]+\mu(1+\theta^{(\mu)}))(1+\delta_1)}(1+\delta_2) \\ &= -\frac{\sigma}{x[0]+\mu}(1+\theta^{(v[0])}) \end{aligned}$$

Here, $\theta^{(\sigma)} = \theta_{m-1}$, $\theta^{(\mu)} = \theta_m$, $\delta_1 \le u$, and $\delta_2 \le u$. The next step involves computing the error bound for $\theta^{(v[0])}$. The error bound for $\theta^{(v[0])}$ can be expressed as:

$$\begin{aligned} |\theta^{(v[0])}| &= \left| \frac{(1+\theta^{(\sigma)})(1+\delta_2)(x[0]+\mu)}{(x[0]+u(1+\theta^{(\mu)]}))(1+\theta_1)} - 1 \right| \\ &\le \left| \frac{(1+\theta^{(\sigma)})(1+\delta_2)}{(1+\theta^{(\mu)})(1+\delta_1)} - 1 \right| \\ &\le |1+\theta_{m+2m+3} - 1| \\ &\le |\theta_{3m+3}| \\ &\le \gamma_{3m+3} \end{aligned}$$

Next, we calculate $\beta = \frac{2v[0]v[0]}{\sigma + v[0]v[0]}$. The computation for $\hat{\beta}$ is given by:

$$\begin{aligned} \hat{\beta} &= \frac{2\hat{v[0]}\hat{v[0]}\prod_{i=1}^{3}(1+\delta_i)}{(\hat{\sigma} + \hat{v[0]}\hat{v[0]})\prod_{i=1}^{2}(1+\delta_i)} \\ &= \frac{2v[0]v[0](1+\theta_1^{(v[0])})(1+\theta_2^{(v[0])})\prod_{i=1}^{3}(1+\delta_i)}{\left(\sigma(1+\theta^{(\sigma)}) + v[0]v[0](1+\theta_3^{(v[0])})(1+\theta_4^{(v[0])})\right)\prod_{i=1}^{2}(1+\delta_i)} \end{aligned}$$

Further, we compute the error bound for $\theta^{(\beta)}$ as:

$$|\theta^{(\beta)}| \le |\theta_{20m+21}|.$$

Finally, we update v by assigning $v/v[0]$ to it. Since v is a copy of x, except for $v[0]$, so $|\theta^{(v[j])}| \le |\theta^{(v[0])}|$, where $v[j]$ represents any element vector v, and the corresponding error bound expressed as $|\theta^{(v[j])}| \le |\theta_{3m+4}|$.

Let's consider the procedure for updating the matrix R and analyzing the error bounds of the computed triangular matrix $\hat{R} = R + \Delta R$, where R is the result of the QR decomposition of the matrix $A \in \mathbb{R}^{m\times n}$

using the algorithm described in Algorithm 3.1. In each step of the algorithm, the matrix R is updated column by column. To do this, we utilize the Householder transformation with a computed vector v and scalar β.

For the specific partition of the matrix R, denoted as $R[j:,j:]$, which is currently under computation, we will refer to it as R_{block}. We perform the update $R_{block} = R'_{block} - \beta vv^T R'_{block}$, where v and β are obtained through the function $House(A[j,j:])$.The error analysis of this process involves considering the effect of these updates on the accuracy of the computed triangular matrix. For the j-th column, the error bound after a single iteration can be expressed as

$$\begin{aligned}
\|\Delta R\|_F &= \left\| \left(R'_{block} - \hat{\beta}\hat{v}\hat{v}^T R'_{block} \prod_{i=1}^{3}(1+\delta_i) \right)(1+\delta_4) - (R'_{block} - \beta vv^T R'_{block}) \right\|_F \\
&\leq \left\| (R'_{block} - \beta vv^T R'_{block})(1+\theta_{26(m-j)+31}) - (R'_{block} + \beta vv^T R'_{block}) \right\|_F \\
&\leq \left\| (R'_{block} - \beta vv^T R'_{block})(\theta_{26(m-j)+31}) \right\|_F \\
&\leq |\theta_{26(m-j)+31}| \, \|R\|_F
\end{aligned}$$

Therefore, for any element of matrix $R[\hat{j},k] = R[j,k](1+\theta^{(R[j,k])})$, the condition $|\theta^{(R[j,k])}| \leq |\theta_{26(m-j)+31}|$ holds true.

As the algorithm progresses through each column, the last column (n-th column) of R, denoted as $R[n,n:]$, undergoes the most iterations, namely n iterations. Thus, after n iterations, the error bound for the n-th column can be bounded as follows:

$$\begin{aligned}
\|\hat{R'_{block}}\|_F &\leq \| \prod_{j=1}^{n} |1+\theta_{26(m-j)+31}| \|R'_{block}\|_F \\
&\leq |1+\theta_{\sum_{j=1}^{n}(26(m-j)+31)}| \|R'_{block}\|_F \\
&\leq |1+\theta_{\sum_{j=1}^{n}(26m+31)}| \|R'_{block}\|_F \\
&\leq |1+\theta_{(26mn+31)}| \|R'_{block}\|_F \\
&\leq |(1+\theta_{26mn})(1+\theta_{31})| \|R'_{block}\|_F \\
&\leq |(1+\gamma_{26mn})(1+\gamma_{31})| \|R'_{block}\|_F \\
&\leq |(1+\lambda\sqrt{26}u\sqrt{mn})(1+\lambda\sqrt{31}u)| \|R'_{block}\|_F \\
&\leq (1+C_1\sqrt{mn}+C_2)\|R'_{block}\|_F + O(u^2)
\end{aligned} \tag{3.1}$$

Therefore, for each element R_{jk} in the matrix R, the error can be bounded as $\|\Delta R_{jk}\|_F \leq (C_1\sqrt{mn} + C_2)\|R_{jk}\|_F + O(u^2)$, where $C_1 = \lambda\sqrt{26}u(1+\lambda\sqrt{31}u)$, and $C_2 = \lambda\sqrt{31}u$.

In the paper [3], a similar outcome is reported: each element R_{jk} of the matrix R satisfies the error bound: $\|\Delta R_{jk}\|_F \leq C_3\lambda\sqrt{n}\gamma_m\|R_{jk}\|_F + O(u^2)$.

3.4. Orthogonality Error. Another important error measure in QR factorization is the orthogonality error $\|Q^TQ - I\|_F$. It quantifies the deviation of the computed Q matrix from an orthogonal matrix. The error is calculated as the Euclidean norm of the difference between the product of Q's transpose and Q and the identity matrix.

To begin, we analyze the error bounds for the matrices $\Delta W = \hat{W} - W$ and $\Delta Y = \hat{Y} - Y$. For the matrix W, each column is constructed using the Householder vector v, thereby attributing errors to the Householder vectors. It's important to note that longer Householder vectors introduce larger errors, as described by Equation 3.3. Consequently, the error bound for ΔW can be represented as $\|\Delta W\|_F \leq \max(\theta^{(v)})\|W\|_F \leq \theta_{3m+4}\|W\|_F$.

Shifting our focus to matrix Y, its columns are formed by $z = \beta v - \beta W(Y^T v)$. Introducing $\hat{z} = z + \Delta z$, we have

$$\|\hat{z}\|_F \leq \begin{cases} (1+\theta_{27m-20j+30})\|z\|_F, & j=0 \\ (1+\theta_{24m-20j+26})(1+\Delta z_{j-1}/\|z_{j-1}\|_F)\|z\|_F, & j>0 \end{cases},$$

where j signifies the current column index. Consequently, we have

$$\begin{aligned}\|\hat{z}\|_F &\leq (1+\theta_{27m-20j+30+\sum_{j=1}^{n-1}(24m-20j+30)})\|z\|_F \\ &\leq (1+\theta_{24mn-10nn+3m+40n)})\|z\|_F \\ &\leq (1+\theta_{24mn+3m+40n})\|z\|_F \\ &\leq (1+\lambda(\sqrt{24mn}+\sqrt{3m}+\sqrt{40n})u)\|z\|_F + O(u^2) \\ &\leq (1+C_4\sqrt{mn}+C_5\sqrt{m}+C_6\sqrt{n})\|z\|_F + O(u^2)).\end{aligned}$$

Consequently, the error bound for ΔY can be represented as $\|\Delta Y\|_F \leq (C_4\sqrt{mn}+C_5\sqrt{m}+C_6\sqrt{n})\|Y\|_F + O(u^2))$, where $C_4 = 2\sqrt{6}\lambda u$, $C_4 = \sqrt{3}\lambda u$, and $C_4 = 2\sqrt{10}\lambda u$.

based on Proof 3, we compute $\Delta Q = \hat{Q} - Q$.

$$\begin{aligned}\|\Delta Q\|_F &= \|(I-\hat{W}\hat{Y}^T(1+\theta_n))(1+\delta) - (I-WY^T)\|_F \\ &\leq \|I - W(1+\|\frac{\Delta W}{W}\|_F)Y^T(1+\|\frac{\Delta Y}{Y}\|_F)(1+\theta_{n+1}) - (I-WY^T)\|_F \\ &\leq |(1+\theta_{3m+4})(1+\theta_{24mn+3m+40n})(1+\theta_{n+1}) - 1|\|Q\|_F \\ &\leq |2\gamma_{3m+4} + \gamma_{24mn+3m+40n} + 4\gamma_{n+1}|\|Q\|_F\end{aligned} \tag{3.2}$$

Considering (3.3), we proceed to evaluate the Frobenius norm of the expression $\|\hat{Q}^T\hat{Q} - I\|_F$ as follows:

$$\begin{aligned}\|\hat{Q}^T\hat{Q} - I\|_F &= \|\hat{Q}^T\hat{Q} - I\|_F \\ &\leq \|Q^T(1+\|\frac{\Delta Q}{Q}\|_F)Q(1+\|\frac{\Delta Q}{Q}\|_F) - I\|_F \\ &\leq (2\|\frac{\Delta Q}{Q}\|_F + \|\frac{\Delta Q}{Q}\|_F\|\frac{\Delta Q}{Q}\|_F)\|Q^TQ\|_F \\ &\leq |(11\gamma_{3m+4} + 3\gamma_{24mn+3m+40n} + 48\gamma_{n+1})|\|Q^TQ\|_F \\ &\leq |(11\gamma_{3m+4} + 3\gamma_{24mn+3m+40n} + 48\gamma_{n+1})|\|Q^TQ\|_F \\ &\leq \lambda u(6\sqrt{6mn} + 14\sqrt{3m} + (6\sqrt{10}+4\sqrt{3})\sqrt{n} + 26)\|Q^TQ\|_F + O(u^2)\end{aligned} \tag{3.3}$$

3.5. Least Squares Solution Error. The residual error $\|Ax-b\|_2$ in solving the least squares problem using QR factorization is given by the Euclidean norm of the difference between the computed solution Ax and the target vector b. Utilizing QR factorization for a given matrix $A \in \mathbb{R}^{m\times n}$, we transform the problem to $\min\|Ax-b\|_2$ into $\min\|Rx-y\|_2$, where R is the upper triangular factor obtained from the QR factorization and $y = Q^Tb$. The computational result can be represent as $\min\|\hat{R}x - y\|_2 = \min\|Rx + \Delta Rx - y - \Delta y\|_2 \leq \min\|Rx-y\|_2 + \|\Delta Rx - \Delta y\|_2$, where ΔRx represents the additional term introduced due to computational approximations. This term is bounded by $\|\Delta Rx\|_2 \leq ((C_1\sqrt{mn}+C_2)\|R\|_F\|x\|_2 + O(u^2))$, where C_1 and C_2 are constants (3.1). Similarly, $\|\Delta y\|_2 \leq (C_3\sqrt{mn})\|Q\|_F + O(u^2)$, where C_3 are constants(3.2). Therefore, for $\min\|Rx-y\|_2$, we have

$$\min\|Rx-y\|_2 \leq (C_4\sqrt{mn}+C_5)\max(\|R\|_F\|x\|_2, \|Q\|_F) + O(u^2) \tag{3.4}$$

4. Numerical Experiments. The section of numerical computation is devoted to the meticulous examination of error propagation and precision considerations inherent in computational linear algebra algorithms. In this section, we delve into various aspects of error analysis, encompassing inner products, QR factorization, orthogonal matrices, and solutions to LS problems.

4.1. Error in inner products. We will begin by analyzing the error propagation of a single operation, specifically when computing accumulating random floating-point numbers. In particular, we will consider the operation $c = c+x$. To investigate this, we utilize the NumPy library [5] to simulate rounding errors. In this context, we establish a reference point by considering the calculation result with high precision (`np.float64`) as the reference result. By quantifying the difference between the computation outcome using low precision (`np.float32`) and the high precision reference, we estimate the relative error of the low-precision result.

Our findings are visualized through a dual-panel plot in Figure 3b. The graph's left panel displays a heatmap of the errors alongside their corresponding steps. Dotted vertical lines are situated at positions of 2^n, where n is a positive integer. Meanwhile, the right panel showcases a histogram representing the distribution of errors, complemented by a kernel density estimation curve.

Through simulations, we examine rounding errors for two distinct scenarios: random vectors with elements sampled from the interval $[0, 2]$ and random vectors with elements spanning the interval $[-1, 1]$.

For random vectors with elements between 0 and 2, our results show a sharp increase in the rounding error at power of 2, where the error reaches approximately 6×10^{-8}, followed by a gradual decrease to approximately 3×10^{-8}. Moreover, the error distribution appears symmetric with respect to 0.

For random vectors with elements in the range $[-1, 1]$, we observe that the error distribution is relatively unaffected by the number of steps. Although the distribution of errors in this case is similar to the distribution seen in random vectors with shape and element range $[0, 2]$, the errors' magnitudes exhibit a greater concentration near zero.

If we employ `numpy.isclose` to ascertain the exact results, where $|a - b| \leq (a_{tol} + r_{tol} \cdot |b|)$, with $r_{tol} = 10^{-5}$ and $a_{tol} = 10^{-8}$. For arrays with non-zero means, the probability of achieving exact results is approximately 1.698%. This probability drops significantly to 1.335×10^{-3}% for arrays with mean zero. Analyzing mean errors, we note values of -5.977×10^{-12} for arrays with non-zero means and 3.452×10^{-12} for arrays with a mean of zero.

It is essential to recognize that the mean error and the probability of obtaining accurate results can vary across different sets of random numbers. However, through numerous experiments, a consistent pattern emerges: arrays with a mean of zero tend to yield accurate results with a probability about 100 times higher than the other array.

In the upcoming explanation, we will clarify why the relative error initially reaches its peak at 2^n and subsequently decreases to half of its peak value during the steps from 2^n to 2^{n+1}.

Let x_i is a random variable which adheres to a uniform distribution over the interval $[0, 2]$. To represent the cumulative sum of x_i, we start with two versions: the exact result $s_0 = x_0$ and $s_{i+1} = \hat{s_i} + x_{i+1}$, as well as the computational result $\hat{s_0} = x_0$ and $\hat{s_{i+1}} = fl(\hat{s_i} + x_{i+1})$.

When the number of steps reaches 2^n, due to the uniform distribution of x_i, s_i tends to be around 2^n. We assume $s_i = 2^n - \epsilon$ and $s_{i+1} = 2^n + \epsilon$, where ϵ is a positive number. For s_i, its absolute error can be represent as $|s_i - \hat{s_1}| \leq \frac{1}{2} \times \beta^{e-t}$ and for s_{i+1} absolute error is bounded by $|s_{i+1} - \hat{s_{i+1}}| \leq \frac{1}{2} \times \beta^{e-t+1}$ (refer to equation2.2).

The relative err for s_i can be written as $err_i = \frac{|s_i - \hat{s_i}|}{s_i} \leq \frac{\frac{1}{2} \times \beta^{e-t}}{2^n - \epsilon}$. Similarly, the relative err for s_{i+1} is given by $err_{i+1} = \frac{|s_{i+1} - \hat{s_{i+1}}|}{s_{i+1}} \leq \frac{\frac{1}{2} \times \beta^{e-t+1}}{2^n + \epsilon}$.

It becomes apparent that the relative error at $s_i$is twice that at s_{i+1}, when ϵapproaches 0^+. As for s_{i+1}, the relative error bound decreases as ϵincreases to 2^n. It eventually reaches a value of $\frac{\frac{1}{2} \times \beta^{e-t+1}}{2^n + 2^n} = \frac{\frac{1}{2} \times \beta^{e-t}}{2^n}$, which equals half the value achieved as ϵapproaches 0^+.

Next, the relationship between the inner product error bound and the vector length will be discussedFigure 4b. We focus on three specific precisions: float32, float64, and mixed precision. The mixed-precision approach involves using a mixed fused multiply-add (FMA) operation, where C=C+AB uses high precision for C and low precision for A and B. We use the calculation results under float128 precision as reference values. Infer the relationship between inner product error and vector length by fitting a logarithmic model.

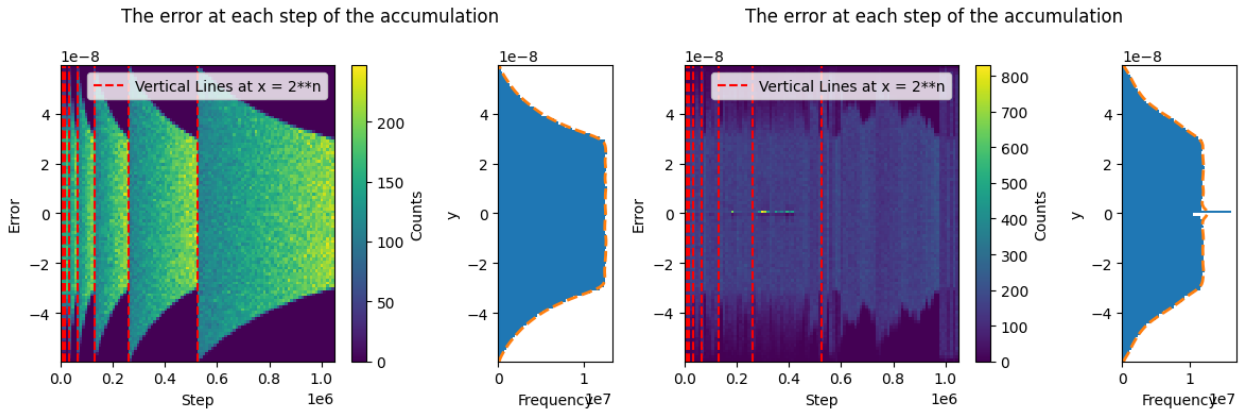
We will first generate random vectors of different lengths to calculate the inner product, and then compare the calculated results with reference values to determine the relative error at each length. For each length n, we will generate 20 different random vectors to calculate the relative error.

Then we go to the maximum relative error at each length for fitting error bounds. We use a logarithmic transformation for modeling the error bound:

$$\log(\text{error Bound}) = k \log(n) + c,$$

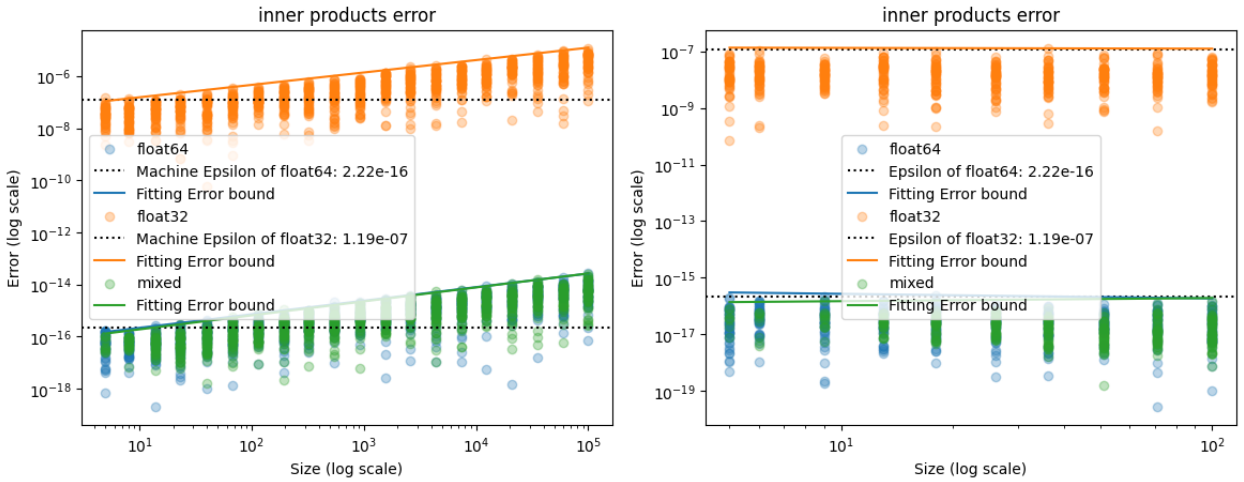
where both k and c denote constants. We will then lift the resulting fitted curve to ensure it lies above all data points. At the same time, we add the machine epsilon as a reference in the graph to represent the relationship between accuracy and error.

our findings reveal the following patterns:



(a) Elements in [0,2] (b) Elements in [-1,1]

Fig. 3: Error in Single Operation



(a) Elements in [0,2] (b) Elements in [-1,1]

Fig. 4: Error in Inner Products

Non-Zero Mean Vectors: Among vectors with non-zero means across the three precisions, the relative error demonstrates a relationship that scales proportionally with the square root of the vector's length.

Zero Mean Vectors: For vectors with an average element value of 0, the relative error remains constant and is unaffected by the vector's length.

These observations highlight the suitability of different error estimation scenarios for random vectors:Item B2. scenario is particularly well-suited for random vectors with non-zero mean values, where the relative error follows a proportional relationship. The outcome is more optimistic than estimated, indicating that this case provides a favorable error estimation for such vectors.On the other hand, B3. scenario aligns with random vectors possessing a mean element value of 0. The constant relative error it yields is independent of the vector's length. Here, the estimation in Item B1. proves to be more pessimistic, making Item B2. and B3. more accurate choices for these random vectors.

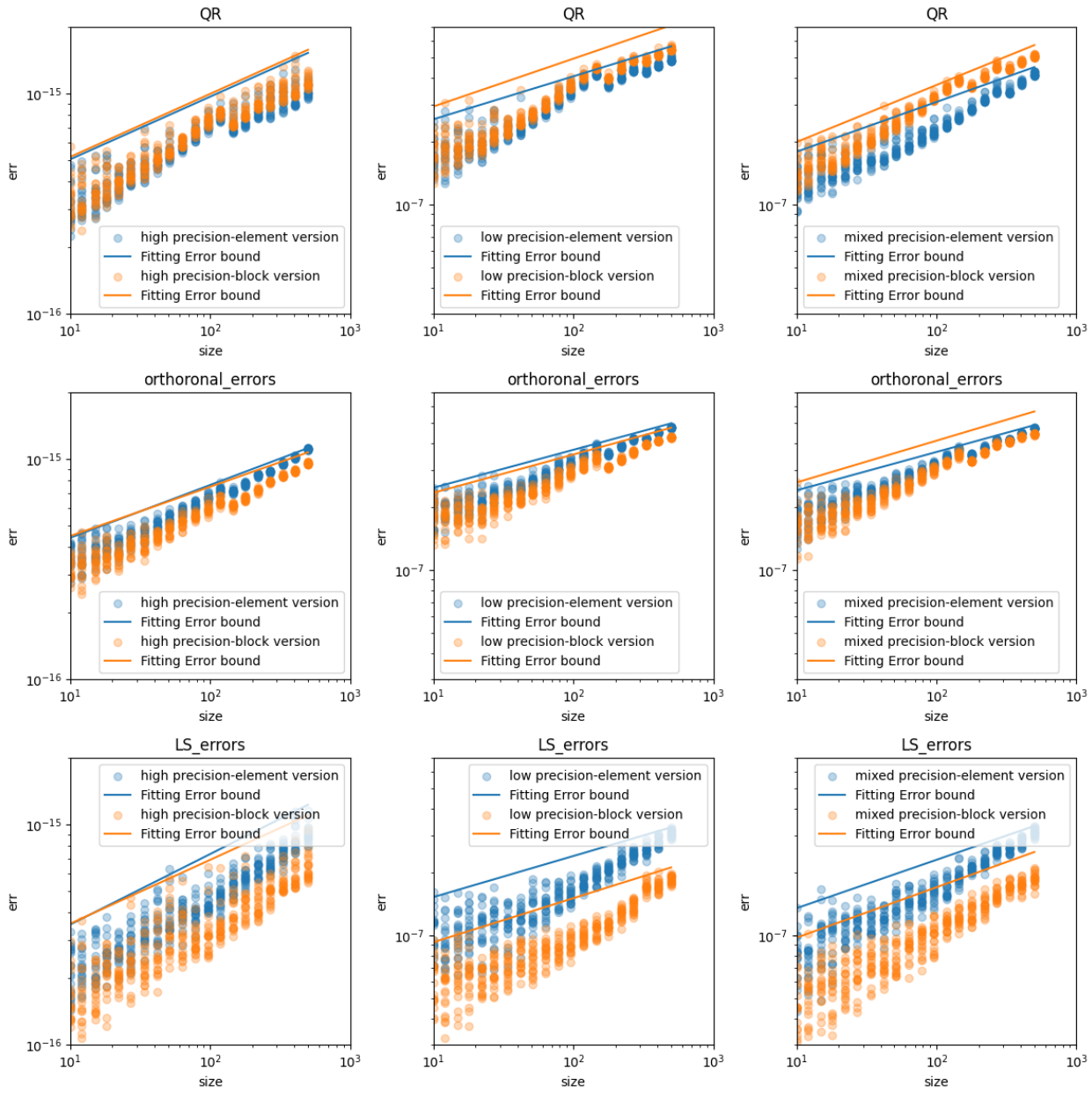


Fig. 5: Error Analysis in QR Decomposition for Block and Element Decomposition Across Various Precisions

4.2. Error in QR factorization. We initiate the process by generating a random matrix $A \in \mathbb{R}^{2n\times n}$ for the purpose of QR decomposition. Our objective revolves around dissecting the errors prevalent in both the element-wise and block-wise algorithms. We meticulously examine two precision paradigms: single floating-point precision and mixed precision. Within this context, we simulate and analyze three distinct QR decomposition errors: the Matrix Reconstruction Error $\frac{\|A-Q\hat{R}\|_F}{\|A\|}$, the Orthogonality Error $\frac{\|\hat{Q}^T\hat{Q}-I\|_F}{\|Q\|_F}$, and the Least Squares Solution Error $\frac{\|Rx-Q^Tb\|_2}{\|b\|_2}$.

Particularly noteworthy is our employment of mixed-precision FMA operations. This entails replacing the computational element of single-precision product addition in the program. The analysis spans matrices characterized by both with zero mean and non-zero mean elements, elucidating similar outcomes. As a result, our focus is primarily directed towards the scenario where the element mean is not zero. In examining the

QR	high	element	Slope	0.28
			Interce pt	-35.88
		block	Slope	0.29
			Interce pt	-35.86
	low	element	Slope	0.20
			Interce pt	-15.65
		block	Slope	0.23
			Interce pt	-15.57
	mixed	element	Slope	0.24
			Interce pt	-16.09
		block	Slope	0.27
			Interce pt	-16.06

orthoron al errors	high	element	Slope	0.24
			Intercept	-35.91
		block	Slope	0.22
			Intercept	-35.86
	low	element	Slope	0.18
			Intercept	-15.63
		block	Slope	0.18
			Intercept	-15.69
	mixed	element	Slope	0.18
			Intercept	-15.67
		block	Slope	0.20
			Intercept	-15.62

LS_errors	high	element	Slope	0.32
			Intercept	-36.32
		block	Slope	0.29
			Intercept	-36.24
	low	element	Slope	0.20
			Intercept	-16.15
		block	Slope	0.21
			Intercept	-16.68
	mixed	element	Slope	0.23
			Intercept	-16.35
		block	Slope	0.24
			Intercept	-16.70

Fig. 6: Linear Fit Parameters for Error Analysis

Least Squares Solution Error, denoted as $\frac{\|Rx-Q^Tb\|_2}{\|b\|_2}$, we establish the exact solution as $x = \text{ones}(2n)$, wherein $\text{ones}(2n)$ symbolizes a vector of length $2n$, with all its elements set to one.

Our exploration of inner product errors adopts a parallel methodology, involving the application of logarithms to both the error value and the matrix size n, followed by a linear fitFigure 5. The resulting formulation $\log(err) = k\log(n) + \log(c)$ translated from $err = c \times n^k$.

Upon dissecting the slope of the fitted line k (reflecting the exponent of n), Across all three error types, the proportionality to the power of 0.2-0.3 is consistently observed concerning the matrix size Figure 6. Notably, the realm of high precision accentuates error amplification in contrast to mixed precision and low precision domains. this underscores that the influence of size on error is less pronounced than initially anticipated (3.1), (3.3), (3.2).

Digging further into the constant term c of the line (corresponding to multiples of n), The constant term for high precision is roughly twice that of low precision. This observation aligns with the nature of floating-point systems, where the unit rounding error of float32 stands around 1.2×10^{-7}, and that of float64 is approximately $2.2 \times 10^-16 \simeq (1.2 \times 10^{-7})^2$, affirming the anticipated relationship between error and accuracy (3.1), (3.3), (3.4).

The advantages of the block decomposition algorithm are that the matrix Q obtains high orthogonality in low and mixed precision, and the residuals are smaller in the least squares problem.

5. Conclusion. In conclusion, our focus on tall-and-skinny matrices, where the number of rows (m) exceeds the number of columns (n), led us to explore two versions of the Householder reflection-based algorithms: the element-wise decomposition approach and the block-wise decomposition approach. In the case of the element-wise approach, we evaluated the FLOPs for computing the orthogonal matrix Q using the WY matrix, where the leading term in the complexity is $m^2n/3$. Directly updating Q incurs $2m^3n$ FLOPs. Consequently, the utilization of the WY matrix proves advantageous for enhanced speed and error reduction. For the block-wise decomposition version, we employed parallelization techniques to enhance computational efficiency, concurrently updating the Q and R matrices.

Our error analysis encompassed both the Householder reflection error and an examination of errors within the R matrix resulting from the QR decomposition. We dissected three errors: the Matrix Reconstruction Error $\|A-Q\hat{R}\|_F$, the Orthogonality Error $\|\hat{Q}^T\hat{Q}-I\|_F$, and the Least Squares Solution Error $\|Rx-Q^Tb\|_2$. Importantly, we established that for $\frac{\|\Delta R\|_F}{\|R\|_F}$, $\frac{\|\Delta Q^T\Delta Q-I\|_F}{\|Q\|_F}$, and $\frac{\|min(\Delta Rx-\Delta Q^Tb\|_2)}{min(\|Rx-b\|_2)}$, upper bounds could be determined as $\frac{\|\Delta R\|_F}{\|R\|_F}$, $\frac{\|\Delta Q^T\Delta Q-I\|_F}{\|Q\|_F}$, and $\frac{\|min(\Delta Rx-\Delta Q^Tb\|_2)}{min(\|Rx-b\|_2)}$, where C represents a constant.

In our numerical simulations using random vectors and matrices with a uniform distribution via numpy, we observed that the mean values of vectors and matrices significantly influence error accumulation. For instance, the relative error tends to be significantly lower when the mean of arrays is close to zero. We explained the phenomena behind the relative error reaching its peak at 2^n and subsequently decreasing to half of its peak value during the transition from 2^n to $2^n + 1$. This phenomenon is linked to a step change in the absolute error bound at 2^n.

Furthermore, our numerical simulations of vector inner products showcased that a logarithmic transformation fits the error bound model, represented as $\log(err) = k\log(n) + \log(c)$. For vectors with non-zero means, the relative error scales proportionally with the square root of the vector's length. Conversely, vectors with an average element value of 0 exhibit a constant relative error that remains unaffected by the vector's length. In our numerical simulations of QR factorization, the three distinct error types demonstrated a proportional relationship with the unit rounding error u. The errors exhibited proportionality to the power within the range of 0.2 to 0.3 concerning matrix size, as depicted in Figure 6. This observation highlights a more pronounced error amplification within the realm of high precision as matrix size increases. Notably, the block-wise decomposition algorithm demonstrated its advantages by achieving higher positivity of the Q matrix under low and mixed precision scenarios, along with reduced residuals in the least squares problem.

REFERENCES

[1] Ballard, J. Demmel, L. Grigori, M. Jacquelin, H. D. Nguyen, and E. Solomonik, *Reconstructing householder vectors from tall-skinny qr grey*, 2013, https://api.semanticscholar.org/CorpusID:31186228.

[2] J. Ballingall, *Householder vector algorithm in Golub and Van Loan.* Mathematics Stack Exchange, https://math.stackexchange.com/q/4590331. Version of 03 December 2022.

[3] M. P. Connolly and N. J. Higham, *Probabilistic rounding error analysis of householder qr factorization*, SIAM Journal on Matrix Analysis and Applications, 44 (2023), pp. 1146–1163, https://doi.org/10.1137/22M1514817, https://doi.org/10.1137/22M1514817, https://arxiv.org/abs/https://doi.org/10.1137/22M1514817.

[4] G. H. Golub and C. F. Van Loan, *Matrix Computations*, John Hopkins University Press, 2013.

[5] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, *Array programming with NumPy*, Nature, 585 (2020), pp. 357–362, https://doi.org/10.1038/s41586-020-2649-2, https://doi.org/10.1038/s41586-020-2649-2.

[6] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, Other Titles in Applied Mathematics, Society for Industrial and Applied Mathematics, 2nd ed., 2002, https://books.google.co.uk/books?id=epilvM5MMxwC.

[7] N. J. Higham and T. Mary, *A new approach to probabilistic rounding error analysis*, SIAM Journal on Scientific Computing, 41 (2019), pp. A2815–A2835.

[8] N. J. Higham and T. Mary, *Sharper probabilistic backward error analysis for basic linear algebra kernels with random data*, SIAM Journal on Scientific Computing, 42 (2020), pp. A3427–A3446, https://doi.org/10.1137/20M1314355, https://doi.org/10.1137/20M1314355, https://arxiv.org/abs/https://doi.org/10.1137/20M1314355.

[9] *IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Standard 754-1985*, Institute of Electrical and Electronics Engineers, Piscataway, NJ, USA, Oct. 1985, https://doi.org/10.1109/IEEESTD.1985.82928. Reprinted in SIGPLAN Notices, 22(2):9–25, 1987.

[10] *IEEE Standard for Floating-Point Arithmetic, IEEE Std 754-2019 (revision of IEEE Std 754-2008)*, Institute of Electrical and Electronics Engineers, Piscataway, NJ, USA, July 2019, https://doi.org/10.1109/IEEESTD.2019.8766229.

[11] A. Kulkarni and T. Vaccon, *Super-linear convergence in the p-adic qr-algorithm*, Linear and Multilinear Algebra, 70 (2020), pp. 7778 – 7806, https://api.semanticscholar.org/CorpusID:221397683.

[12] M. Martel, *Compressed matrix computations*, 2022, https://arxiv.org/abs/2202.13007.

[13] J.-M. Muller, N. Brunie, F. de Dinechin, C.-P. Jeannerod, M. Joldes, V. Lefèvre, G. Melquiond, N. Revol, and S. Torres, *Handbook of Floating-Point Arithmetic, 2nd edition*, Birkhäuser Boston, 2018. ACM G.1.0; G.1.2; G.4; B.2.0; B.2.4; F.2.1., ISBN 978-3-319-76525-9.

[14] S. M. Rump, T. Ogita, Y. Morikura, and S. Oishi, *Interval arithmetic with fixed rounding mode*, Nonlinear Theory and Its Applications, IEICE, 7 (2016), pp. 362–373, https://doi.org/10.1587/nolta.7.362.

[15] R. Vershynin, *High-Dimensional Probability: An Introduction with Applications in Data Science*, Cambridge Series in Statistical and Probabilistic Mathematics, Cambridge University Press, 2018, https://doi.org/10.1017/9781108231596.

[16] L. M. Yang, A. Fox, and G. Sanders, *Rounding error analysis of mixed precision block householder qr algorithms*, 2021, https://arxiv.org/abs/1912.06217.