ROUNDING ERROR ANALYSIS OF MIXED-PRECISION HOUSEHOLDER QR ALGORITHMS

L. MINAH YANG, ALYSON FOX, AND GEOFFREY SANDERS

Abstract. Although mixed precision arithmetic has recently garnered interest for training dense neural networks, many other applications could benefit from the speed-ups and lower storage if applied appropriately. The growing interest in employing mixed precision computations motivates the need for rounding error analysis that properly handles behavior from mixed precision arithmetic. We present a framework for mixed precision analysis that builds on the foundations of rounding error analysis presented in [13] and demonstrate its practicality by applying the analysis to various Householder QR Algorithms.

1. Introduction.

2. Background: Build up to rounding error analysis for inner products.

3. Algorithms and existing round-off error analyses.

We introduce the Householder QR factorization algorithm (HQR) in subsection 3.1 and two block variants that use HQR within the block in subsections 3.2 and 3.3. The blocked HQR (BQR) in subsection 3.2 partitions the columns of the target matrix and utilizes mainly level-3 BLAS operations and is a well-known algorithm that uses the WY representation of [4]. In contrast, the Tall-and-Skinny QR (TSQR) in subsection 3.3 partitions rows of the matrix and takes a communication-avoiding divide-and-conquer approach that can be easily parallelized (see [7]). We also present the crucial results in standard rounding error analysis of these algorithms that excludes any mixed-precision assumptions. These building steps of round-off error analysis will be easily tweaked for various mixed-precision assumptions in section 4.

3.1. Householder QR (HQR).

The HQR algorithm uses Householder transformations to zero out elements below the diagonal of a matrix (see [16]). We present this as zeroing out all but the first element of some vector, $\mathbf{x}\in\mathbb{R}^m$.

Lemma 3.1. *Given vector $\mathbf{x}\in\mathbb{R}^m$, there exist Householder vector, $\mathbf{v}$, and Householder transformation matrix, $\mathbf{P}_{\mathbf{v}}$, such that $\mathbf{P}_{\mathbf{v}}$ zeros out $\mathbf{x}$ below the first element.*

$$\begin{aligned}\sigma &= -\operatorname{sign}(\mathbf{x}_1)\|\mathbf{x}\|_2, \quad \mathbf{v}=\mathbf{x}-\sigma\hat{e_1},\\ \beta &= \frac{2}{\mathbf{v}^\top\mathbf{v}} = -\frac{1}{\sigma\mathbf{v}_1}, \qquad \mathbf{P}_{\mathbf{v}} = \mathbf{I}_m - \beta\mathbf{v}\mathbf{v}^\top.\end{aligned} \tag{3.1}$$

The transformed vector, $\mathbf{P}_{\mathbf{v}}\mathbf{x}$, has the same 2-norm as $\mathbf{x}$ since Householder transformations are orthogonal: $\mathbf{P}_{\mathbf{v}}\mathbf{x}=\sigma\hat{\mathbf{e}_1}$. In addition, $\mathbf{P}_{\mathbf{v}}$ is symmetric and orthogonal, $\mathbf{P}_{\mathbf{v}}=\mathbf{P}_{\mathbf{v}}^\top=\mathbf{P}_{\mathbf{v}}^{-1}$.

3.1.1. HQR: Algorithm.

Given $\mathbf{A}\in\mathbb{R}^{m\times n}$ and Lemma 3.1, HQR is done by repeating the following processes until only an upper triangle matrix remains. For $i=1,2,\cdots,n$,
Step 1) Compute $\mathbf{v}$ and β that zeros out the i^{th} column of $\mathbf{A}$ beneath a_{ii} (see alg. 1), and
Step 2) Apply $\mathbf{P}_{\mathbf{v}}$ to the bottom right partition, $\mathbf{A}[i:m,i:n]$ (lines 4-6 of alg. 2).

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 and was supported by the LLNL-LDRD Program under Project No. 17-SI-004, LLNL-JRNL-795525-DRAFT.

Consider the following 4-by-3 matrix example adapted from [13]. Let $\mathbf{P_i}$ represent the i^{th} Householder transformation of this algorithm.

$$\mathbf{A} = \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \xrightarrow{\text{apply } \mathbf{P_1} \text{ to } \mathbf{A}} \left[\begin{array}{c|cc} \times & \times & \times \\ \hline 0 & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{array}\right] \xrightarrow{\text{apply } \mathbf{P_2} \text{ to } \mathbf{P_1A}}$$

$$\left[\begin{array}{cc|c} \times & \times & \times \\ 0 & \times & \times \\ \hline 0 & 0 & \times \\ 0 & 0 & \times \end{array}\right] \xrightarrow{\text{apply } \mathbf{P_3} \text{ to } \mathbf{P_2P_1A}} \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \\ 0 & 0 & 0 \end{bmatrix} = \mathbf{P_3P_2P_1A} =: \mathbf{R}$$

Then, the $\mathbf{Q}$ factor for a full QR factorization is $\mathbf{Q} := \mathbf{P_1P_2P_3}$ since $\mathbf{P_i}$'s are symmetric, and the thin factors for a general matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ are

$$\mathbf{Q}_{\text{thin}} = \mathbf{P_1}\cdots\mathbf{P_n}\mathbf{I}_{m\times n} \quad \text{and} \quad \mathbf{R}_{\text{thin}} = \mathbf{I}_{m\times n}^{\top}\mathbf{P_n}\cdots\mathbf{P_1}\mathbf{A}. \tag{3.2}$$

Algorithm 1: β, $\mathbf{v}$, σ = $\texttt{hh\_vec}(\mathbf{x})$. Given a vector $\mathbf{x} \in \mathbb{R}^n$, return $\mathbf{v}$, β, σ that satisfy $(I - \beta\mathbf{v}\mathbf{v}^\top)\mathbf{x} = \sigma\hat{e_1}$ and $\mathbf{v}_1 = 1$ (see [2, 13]).

Input: $\mathbf{x} \in \mathbb{R}^m$
Output: $\mathbf{v} \in \mathbb{R}^m$, and $\sigma, \beta \in \mathbb{R}$ such that $(I - \beta\mathbf{v}\mathbf{v}^\top)\mathbf{x} = \pm\|\mathbf{x}\|_2\hat{e_1} = \sigma\hat{e_1}$

1. $\mathbf{v} \leftarrow \texttt{copy}(\mathbf{x})$
2. $\sigma \leftarrow -\text{sign}(\mathbf{x}_1)\|\mathbf{x}\|_2$
3. $\mathbf{v}_1 \leftarrow \mathbf{x}_1 - \sigma$
4. $\beta \leftarrow -\frac{\mathbf{v}_1}{\sigma}$
5. **return** β, $\mathbf{v}/\mathbf{v}_1$, σ

Algorithm 2: $\mathbf{V}$, $\boldsymbol{\beta}$, $\mathbf{R}$ = $\texttt{HQR2}(A)$. A Level-2 BLAS implementation of the Householder QR algorithm. Given a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ where $m \geq n$, return matrix $\mathbf{V} \in \mathbb{R}^{m\times n}$, vector $\boldsymbol{\beta} \in \mathbb{R}^n$, and upper triangular matrix $\mathbf{R}$. An orthogonal matrix $\mathbf{Q}$ can be generated from $\mathbf{V}$ and $\boldsymbol{\beta}$, and $\mathbf{QR} = \mathbf{A}$.

Input: $A \in \mathbb{R}^{m\times n}$ where $m \geq n$.
Output: $\mathbf{V}$,$\boldsymbol{\beta}$, $\mathbf{R}$

1. $\mathbf{V}, \boldsymbol{\beta} \leftarrow \mathbf{0}_{m\times n}, \mathbf{0}_m$
2. **for** $i = 1 : n$ **do**
3. $\quad\mathbf{v}, \beta, \sigma \leftarrow \text{hh\_vec}(\mathbf{A}[i : \text{end}, i])$
4. $\quad\mathbf{V}[i : \text{end}, i]$, $\boldsymbol{\beta}_i$, $\mathbf{A}[i, i] \leftarrow \mathbf{v}, \beta, \sigma$
5. $\quad\mathbf{A}[i + 1 : \text{end}, i] \leftarrow \text{zeros}(m - i)$
6. $\quad\mathbf{A}[i : \text{end}, i + 1 : \text{end}] \leftarrow \mathbf{A}[i : \text{end}, i + 1 : \text{end}] - \beta\mathbf{v}\mathbf{v}^\top\mathbf{A}[i : \text{end}, i + 1 : \text{end}]$
7. **return** $\mathbf{V}$, $\boldsymbol{\beta}$, $\mathbf{A}[1 : n, 1 : n]$

3.1.2. HQR: Rounding Error Analysis. Now we present an error analysis for alg. 2 by keeping track of the different operations of alg. 1 and alg. 2.

Calculating the i^{th} Householder vector and constant. In alg. 2, the i^{th} Householder vector shares all but the first component with the target column, $\mathbf{A}[i:m,i]$. We first calculate σ as is implemented in line 2 of alg. 1.

$$\mathrm{fl}(\sigma)=\hat{\sigma}=\mathrm{fl}(-\mathrm{sign}(\mathbf{A}_{\mathrm{i,i}})\|\mathbf{A}[\mathrm{i:m,i}]\|_2)=\sigma+\Delta\sigma,\quad |\Delta\sigma|\leq\gamma_{\mathrm{m-i+1}}|\sigma|. \tag{3.3}$$

Note that the backward error incurred here is simply that an inner product of a vector in $\mathbb{R}^{m-i+1}$ with itself. Let $\tilde{\mathbf{v}}_1\equiv\mathbf{A}_{i,i}-\sigma$, the penultimate value $\mathbf{v}_1$. The subtraction adds a single additional rounding error via

$$\mathrm{fl}(\tilde{\mathbf{v}}_1)=\tilde{\mathbf{v}}_1+\Delta\tilde{\mathbf{v}}_1=(1+\delta)(\mathbf{A}_{i,i}-\sigma-\Delta\sigma)=(1+\tilde{\theta}_{m-i+2})(\mathbf{A}_{i,i}-\sigma)$$

where the last equality is granted because the sign of σ is chosen to prevent cancellation. For the sake of simplicity, we write $|\Delta\tilde{\mathbf{v}}_1|\leq\tilde{\gamma}_{m-i+1}|\tilde{\mathbf{v}}_1|$ even though a tighter relative upper bound is θ_{m-i+2} We sweep that minor difference (in comparison to $\mathcal{O}(m-i)$) under the our use of the $\tilde{\gamma}$ notation defined in **??**. Since alg. 1 normalizes the Householder vector so that its first component is 1, the remaining components of $\mathbf{v}$ are divided by $\mathrm{fl}(\tilde{\mathbf{v}}_1)$ incurring another single rounding error. As a result, the rounding errors in $\mathbf{v}$ are

$$\mathrm{fl}(\mathbf{v}_j)=\mathbf{v}_j+\Delta\mathbf{v}_j \text{ where } |\Delta\mathbf{v}_j|\leq\begin{cases}0, & j=1\\ \tilde{\gamma}_{m-i+1}|\mathbf{v}_j|, & j=2:m-i+1.\end{cases} \tag{3.4}$$

Next, we consider the Householder constant, β, as is computed in line 4 of alg. 1.

$$\hat{\beta}=\mathrm{fl}\left(-\frac{\tilde{\mathbf{v}}_1}{\hat{\sigma}}\right)=-(1+\delta)\frac{\tilde{\mathbf{v}}_1+\boldsymbol{\Delta}\tilde{\mathbf{v}}_1}{\sigma+\Delta\sigma} \tag{3.5}$$

$$=\frac{(1+\delta)(1+\theta_{m-i+1})}{(1+\theta_{m-i+2})}\beta=(1+\theta_{3(m-i+2)})\beta \tag{3.6}$$

$$=\beta+\Delta\beta,\ \text{where } |\Delta\beta|\leq\tilde{\gamma}_{m-i+1}\beta. \tag{3.7}$$

We have shown (3.5) to keep our analysis simple in section 4 and (3.6) and (3.7) show that the error incurred from calculating of $\|\mathbf{A}[i:m,i]\|_2$ accounts for the vast majority of the rounding error so far.

Applying a Single Householder Transformation. Now we consider lines 4-6 of alg. 2. Since the entries in $\mathbf{A}[i+1:m,i]$ are simply zeroed out and $\mathbf{A}_{i,i}$ is replaced by σ, we only need to calculate the errors for applying a Householder transformation with the computed Householder vector and constant. This is the most crucial building block of the rounding error analysis for any variant of HQR because the $\mathbf{Q}$ factor is formed by applying the Householder transformations to the identity and both of the blocked versions in subsection 3.2 and subsection 3.3 require efficient implementations of this step. In this section, we only consider a level-2 BLAS implementation of applying the Householder transformation, but in subsection 3.2 we introduce a level-3 BLAS implementation.

A Householder transformation is applied through a series of inner and outer products, since Householder matrices are rank-1 updates of the identity. That is, computing $\mathbf{P}_{\mathbf{v}}\mathbf{x}$ for any $\mathbf{x}\in\mathbb{R}^m$ is as simple as computing $\mathbf{y}:=\mathbf{x}-(\beta\mathbf{v}^\top\mathbf{x})\mathbf{v}$. Let us assume that $\mathbf{x}$ is an exact vector and there were errors incurred in forming $\mathbf{v}$ and β. The errors incurred from computing $\mathbf{v}$ and β need to be included in addition to the new rounding errors accumulating from the action of applying $\mathbf{P}_{\mathbf{v}}$ to a column. In practice, $\mathbf{x}$ would be a column in $\mathbf{A}^{(i-1)}[i+1:m, i+1:n]$, where the superscript $(i-1)$ indicates that this submatrix of $\mathbf{A}$ has already been transformed by $i-1$ Householder transformations that zeroed out components below $\mathbf{A}_{j,j}$ for $j = 1 : i-1$. We show the error for forming $\mathrm{fl}\left(\hat{\mathbf{v}}^\top \mathbf{x}\right)$ where we continue to let $\mathbf{v}, \mathbf{x} \in \mathbb{R}^{m-i+1}$ as would be in the i^{th} iteration of the for-loop in alg. 2:

$$\mathrm{fl}\left(\hat{\mathbf{v}}^\top \mathbf{x}\right) = (1+\theta_{m-i+1})(\mathbf{v}+\boldsymbol{\Delta}\mathbf{v})^\top \mathbf{x}.$$

Set $\mathbf{w} := \beta \mathbf{v}^\top \mathbf{x} \mathbf{v}$. Then,

$$\hat{\mathbf{w}} = (1+\theta_{m-i+1})(1+\delta)(1+\tilde{\delta})(\beta+\Delta\beta)(\mathbf{v}+\boldsymbol{\Delta}\mathbf{v})^\top \mathbf{x}(\mathbf{v}+\boldsymbol{\Delta}\mathbf{v}),$$

where θ_{m-i+1} is from computing the inner product $\hat{\mathbf{v}}^\top \mathbf{x}$, and δ and $\tilde{\delta}$ are from multiplying β, $\mathrm{fl}(\hat{\mathbf{v}}^\top \mathbf{x})$, and $\hat{\mathbf{v}}$ together. Finally, we can add in the vector subtraction operation and complete the rounding error analysis of applying a Householder transformation to any vector:

$$\mathrm{fl}(\mathbf{x} - \hat{\mathbf{w}}) = (1+\delta)(\mathbf{x} - \mathbf{w} - \boldsymbol{\Delta}\mathbf{w}) = (1+\tilde{\theta}_{m-i+1})\mathbf{y}. \tag{3.8}$$

We can easily switch between forward and errors from (3.8) via

$$\mathbf{y} + \boldsymbol{\Delta}\mathbf{y} = (1+\tilde{\theta}_{m-i+1})\mathbf{y} = (1+\tilde{\theta}_{m-i+1})\mathbf{P}_{\mathbf{v}}\mathbf{x} = (\mathbf{P}_{\mathbf{v}} + \boldsymbol{\Delta}\mathbf{P}_{\mathbf{v}})\mathbf{x},$$

where $|\boldsymbol{\Delta}\mathbf{y}| \leq \tilde{\gamma}_{m-i+1}|\mathbf{y}|$ and $|\boldsymbol{\Delta}\mathbf{P}_{\mathbf{v}}| \leq \tilde{\gamma}_{m-i+1}|\mathbf{P}_{\mathbf{v}}|$.

Even though we never explicitly form $\mathbf{P}_{\mathbf{v}}$, forming the normwise error bound for this matrix makes the analysis for HQR simpler. Therefore, we now transition from componentwise error to matrix norm errors: the 2-norm and the Frobenius norm.

First, we transition from componentwise forward error to the 2-norm forward error via

$$\|\boldsymbol{\Delta}\mathbf{y}\|_2 = \left(\sum_{i=1}^m \boldsymbol{\Delta}\mathbf{y}_i^2\right)^{1/2} \leq \left((\tilde{\gamma}_{m-i+1})^2 \sum_{i=1}^m |\mathbf{y}_i|^2\right)^{1/2} = \tilde{\gamma}_{m-i+1}\|\mathbf{y}\|_2. \tag{3.9}$$

In exact arithmetic, we are guaranteed $\|\mathbf{y}\|_2 = \|\mathbf{P}_{\mathbf{v}}\mathbf{x}\|_2 \leq \|\mathbf{P}\|_2\|\mathbf{x}\|_2 = \|\mathbf{x}\|_2$ since $\mathbf{P}_{\mathbf{v}}$ is orthogonal and preserves norms. Combining this with (3.9) we find

$$\frac{\|\boldsymbol{\Delta}\mathbf{y}\|_2}{\|\mathbf{x}\|_2} \leq \tilde{\gamma}_{m-i+1}. \tag{3.10}$$

Now we convert this to a normwise backward error. Since $\boldsymbol{\Delta}\mathbf{P}$ is exactly $\frac{1}{\mathbf{x}^\top\mathbf{x}}\boldsymbol{\Delta}\mathbf{y}\mathbf{x}^\top$, we can compute its Frobenius norm by using $\boldsymbol{\Delta}\mathbf{P}_{ij} = \frac{1}{\|\mathbf{x}\|_2^2}\boldsymbol{\Delta}\mathbf{y}_i\mathbf{x}_j$,

$$\|\boldsymbol{\Delta}\mathbf{P}\|_F = \left(\sum_{i=1}^m\sum_{j=1}^m\left(\frac{1}{\|\mathbf{x}\|_2^2}\boldsymbol{\Delta}\mathbf{y}_i\mathbf{x}_j\right)^2\right)^{1/2} = \frac{\|\boldsymbol{\Delta}\mathbf{y}\|_2}{\|\mathbf{x}\|_2} \leq \tilde{\gamma}_{m-i+1},$$

where the last inequality is a direct application of (3.10). We summarize these results in Lemma 3.2.

LEMMA 3.2. *Let $\mathbf{x} \in \mathbb{R}^m$ and consider the computation of $\hat{\mathbf{y}} = \mathrm{fl}(\mathbf{P}_{\mathbf{v}}\mathbf{x})$ via*

$$\mathbf{y} + \boldsymbol{\Delta}\mathbf{y} = \mathrm{fl}(\mathbf{P}_{\mathbf{v}}\mathbf{x}) = \mathrm{fl}(\mathbf{x} - \hat{\beta}\hat{\mathbf{v}}\hat{\mathbf{v}}^\top\mathbf{x})$$

and rounding errors incurred in forming $\hat{\mathbf{v}}$ and $\hat{\beta}$ are expressed componentwise via $\hat{\mathbf{v}} = \mathbf{v} + \Delta\mathbf{v}$ and $\hat{\beta} = \beta + \Delta\beta$. Let us write the componentwise forward error bound as $|\boldsymbol{\Delta}\mathbf{y}| \leq \gamma_y|\mathbf{y}|$. Then, the normwise forward and backward errors are

$$\|\Delta\mathbf{y}\|_2 \leq \gamma_y\|\mathbf{y}\|_2, \quad \|\mathbf{P}_{\mathbf{v}}\|_F \leq \gamma_y.$$

Note that in a uniform precision setting this bound is represented as $\gamma_y = \tilde{\gamma}_m$, where the majority of the round-off errors are attributed to inner product computations for forming $\hat{\beta}$ and $\mathbf{v}$.

Applying many successive Householder transformations. Consider applying a sequence of transformations in the set $\{\mathbf{P_i}\}_{i=1}^r \subset \mathbb{R}^{m\times m}$ to $\mathbf{x} \in \mathbb{R}^m$, where $\mathbf{P_i}$'s are all Householder transformations. This is directly applicable to HQR as $\mathbf{Q} = \mathbf{P_1}\cdots\mathbf{P_n}\mathbf{I}$ and $\mathbf{R} = \mathbf{Q}^\top\mathbf{A} = \mathbf{P_n}\cdots\mathbf{P_1}\mathbf{A}$. Let us define

$$\mathbf{Q} + \boldsymbol{\Delta}\mathbf{Q'} \equiv \prod_{i=1}^{r}\left(\mathbf{P}_i + \boldsymbol{\Delta}\mathbf{P}_i\right)$$

in the context of applying this matrix to a vector, $\mathbf{x} \in \mathbb{R}^m$, where $\boldsymbol{\Delta}\mathbf{Q'}^\top$ represents the backward error of forming $\mathbf{R}$, instead of the forward error of the $\mathbf{Q}$ factor. The forward error for $\mathbf{Q}$ is denoted as $\boldsymbol{\Delta}\mathbf{Q} \equiv \mathrm{fl}(\mathbf{Q}) - \mathbf{Q}$ where $\mathrm{fl}(\mathbf{Q})$ is formed via HQR. That is, if $\mathbf{y} = \mathbf{Q}^\top\mathbf{x}$, then $\mathrm{fl}(\mathbf{y}) = \mathbf{y} + \boldsymbol{\Delta}\mathbf{y} = (\mathbf{Q} + \boldsymbol{\Delta}\mathbf{Q'})^\top\mathbf{x}$. Even though an efficient implementation would use that $\mathbf{P_i}$'s are applied to successively shorter vectors ($\mathbf{P_i}$ is left multiplied to $\mathbf{A}[i:m, i+1:n]$, which is equivalent to $n-i$ vectors of length $m-i+1$), we assume $\{\mathbf{P_i}\}_{i=1}^r \subset \mathbb{R}^{m\times m}$ to allow for a simpler analysis while forming a looser bound. We will now use Lemma 3.7 from [13] to bound $\boldsymbol{\Delta}\mathbf{Q'}$ with the Frobenius norm.

$$\begin{aligned}
\|\boldsymbol{\Delta}\mathbf{Q'}^\top\|_F &= \left\|\prod_{r=1}^{1}\left(\mathbf{P}_i + \boldsymbol{\Delta}\mathbf{P}_i\right) - \prod_{i=r}^{1}\mathbf{P}_i\right\|_F, \\
&\leq \left(\prod_{i=1}^{r}(1+\tilde{\gamma}_m) - 1\right)\prod_{i=r}^{1}\|\mathbf{P}_i\|_2 = (1+\tilde{\gamma}_m)^r - 1.
\end{aligned}$$

The last equality results from the orthogonality of Householder matrices, and we further reduce the last term. Generalizing the last rule in Lemma **??** yields

$$(1+\tilde{\gamma}_m)^r = (1+\tilde{\gamma}_m)^{r-2}(1+\tilde{\gamma}_m)(1+\tilde{\gamma}_m) \leq (1+\tilde{\gamma}_m)^{r-2}(1+\tilde{\gamma}_{2m}) \leq \cdots \leq (1+\tilde{\gamma}_{rm}).$$

Now we will use the following equivalent algebraic inequalities to get the final result.

$$0 < a < b < 1 \Leftrightarrow 1-a > 1-b \Leftrightarrow \frac{1}{1-a} < \frac{1}{1-b} \Leftrightarrow \frac{a}{1-a} < \frac{b}{1-b} \tag{3.11}$$

In addition, we assume $r\tilde{\gamma}_m < \frac{1}{2}$, such that

$$\begin{aligned}
(1+\tilde{\gamma}_m)^r - 1 &\leq \gamma_w^{(r\tilde{z})} = \frac{r\tilde{z}u_w}{1-r\tilde{z}u_w} \qquad \text{(by definition)} \qquad &(3.12)\\
&\leq \frac{r\tilde{\gamma}_m}{1-r\tilde{\gamma}_m}, \text{ since } r\tilde{z}u_w < r\tilde{\gamma}_m \qquad \text{(by Equation 3.11)} &(3.13)\\
&\leq 2r\tilde{\gamma}_m \qquad \left(\text{since } r\tilde{\gamma}_m < \frac{1}{2} \text{ implies } \frac{1}{1-r\tilde{\gamma}_m} < 2\right) &(3.14)\\
&= r\tilde{\gamma}_m, &(3.15)
\end{aligned}$$

Therefore, we have $(1+\tilde{\gamma}_m)^r - 1 \leq r\tilde{\gamma}_m$ and

$$\|\Delta \mathbf{Q'}\|_2 \leq \|\Delta \mathbf{Q'}\|_F = \|\Delta \mathbf{Q'}^\top\|_F \leq r\tilde{\gamma}_m \tag{3.16}$$

In this current uniform precision error analysis, the important quantity $\tilde{\gamma}_m$ is derived from the backward error of applying one Householder transformation. To easily generalize this section for mixed-precision analysis, we benefit from alternatively denoting this quantity as $\tilde{\gamma}_{\mathbf{P}}$ with the understanding that $\tilde{\gamma}_{\mathbf{P}}$ will be some combination of $\tilde{\gamma}$'s of differing precisions. Equation (3.15) would then be

$$(1+\tilde{\gamma}_{\mathbf{P}})^r - 1 \leq r\tilde{\gamma}_{\mathbf{P}}. \tag{3.17}$$

Next, we apply (3.16) to the i^{th} columns of $\mathbf{Q}, \mathbf{R}$ and set $r = n$ for a full rank matrix, $\mathbf{A}$. Then,

$$\|\mathbf{\Delta R}[:,i]\|_2 = \|\Delta \mathbf{Q'}^\top \mathbf{A}[:,i]\|_2 \leq \|\Delta \mathbf{Q'}\|_2 \|\mathbf{A}[:,i]\|_2 \leq n\tilde{\gamma}_m \|\mathbf{A}[:,i]\|_2,$$
$$\|\mathbf{\Delta Q}[:,i]\|_2 = \|\Delta \mathbf{Q'}\mathbf{I}[:,i]\|_2 \leq \|\Delta \mathbf{Q'}\|_2 \leq n\tilde{\gamma}_m.$$

These columnwise bounds can now be transformed into matrix norms as follows:

$$\|\mathbf{\Delta R}\|_F = \left(\sum_{i=1}^n \|\mathbf{\Delta R}[:,i]\|_2^2\right)^{1/2} \leq \left(\sum_{i=1}^n n^2\tilde{\gamma}_m^2 \|\mathbf{A}[:,i]\|_2^2\right)^{1/2} = n\tilde{\gamma}_m \|\mathbf{A}\|_F,$$
$$\|\mathbf{\Delta Q}\|_F = \left(\sum_{i=1}^n \|\mathbf{\Delta Q}[:,i]\|_2^2\right)^{1/2} \leq \left(\sum_{i=1}^n \tilde{\gamma}_m^2\right)^{1/2} = n^{3/2}\tilde{\gamma}_m.$$

We gather these results into Theorem 3.3.

THEOREM 3.3. *Let $\mathbf{A} \in \mathbb{R}^{m\times n}$ with $m \geq n$ have full rank, n. Let $\hat{\mathbf{Q}} \in \mathbb{R}^{m\times n}$ and $\hat{\mathbf{R}} \in \mathbb{R}^{n\times n}$ be the thin QR factors of $\mathbf{A}$ obtained via alg. 2, defined via*

$$\hat{\mathbf{R}} = \mathbf{R} + \mathbf{\Delta R} = \mathrm{fl}(\hat{\mathbf{P}}_n \cdots \hat{\mathbf{P}}_1 \mathbf{A}), \;\; n\tilde{\gamma}_m \|\mathbf{A}\|_F$$
$$\hat{\mathbf{Q}} = \mathbf{Q} + \mathbf{\Delta Q} = \mathrm{fl}(\hat{\mathbf{P}}_1 \cdots \hat{\mathbf{P}}_n \mathbf{I}), \;\; \|\mathbf{\Delta Q}\|_F \leq n^{3/2}\tilde{\gamma}_m.$$

Let $\mathbf{A} + \mathbf{\Delta A} = \hat{\mathbf{Q}}\hat{\mathbf{R}}$, where $\hat{\mathbf{Q}}$ and $\hat{\mathbf{R}}$ are obtained via Algorithm 2. Then the backward error is

$$\|\mathbf{\Delta A}\|_F \leq n^{3/2}\tilde{\gamma}_m \|\mathbf{A}\|_F. \tag{3.18}$$

The content of this section is largely derived directly from [13], but we kept the analysis general by employing quantities denoted via $\Delta\beta$, $\mathbf{\Delta v}$, $\tilde{\gamma}_y$, and $\tilde{\gamma}_{\mathbf{P}}$. These quantities account for various forward and backward errors formed in computing essential components of HQR, namely the Householder constant and vector, as well as normwise errors of the action of applying Householder transformations. In the next sections, we present blocked variants of HQR that use alg. 2.

3.2. Block HQR with partitioned columns (BQR). We refer to the blocked variant of HQR where the columns are partitioned as BQR. Note that this algorithm relies on the WY representation described in [4] instead of the storage-efficient version of [19], which is widely implemented.

3.2.1. The WY Representation. A convenient matrix representation that accumulates r Householder reflectors is known as the WY representation.

LEMMA 3.4. *Suppose* $\mathbf{Q} = \mathbf{I}_m - \mathbf{W}\mathbf{Y}^\top \in \mathbb{R}^{m\times m}$ *is an orthogonal matrix with* $\mathbf{W}, \mathbf{Y} \in \mathbb{R}^{m\times j}$*. If* $\mathbf{P} = \mathbf{I}_m - \beta\mathbf{v}\mathbf{v}^\top$ *with* $\mathbf{v} \in \mathbb{R}^m$ *and* $\mathbf{z} = \beta\mathbf{Q}\mathbf{v}$*, then*

$$\mathbf{Q}_+ = \mathbf{Q}\mathbf{P} = \mathbf{I} - \mathbf{W}_+\mathbf{Y}_+^\top,$$

where $\mathbf{W}_+ = [\mathbf{W}|\mathbf{z}]$ *and* $\mathbf{Y}_+ = [\mathbf{Y}|\mathbf{v}]$ *are each* m*-by-*$(j+1)$*.*

If $\mathbf{Q}$ was already the accumulation of j Householder transformations, then Lemma 3.4 shows us a clever way to build the WY representation of successive Householder transformations. Let us now show the proof for Lemma 3.4.

Proof. A direct right multiplication of $\mathbf{P} := \mathbf{I}_m - \beta\mathbf{v}\mathbf{v}^\top$ onto $\mathbf{Q}$ can be written as

$$\mathbf{Q}\mathbf{P} = \mathbf{Q} - \beta\mathbf{Q}\mathbf{v}\mathbf{v}^\top.$$

Let us use the WY representation of $\mathbf{Q}$.

$$\mathbf{Q}\mathbf{P} = \mathbf{I}_m - \mathbf{W}\mathbf{Y}^\top - \beta\mathbf{Q}\mathbf{v}\mathbf{v}^\top = \mathbf{I}_m - \mathbf{W}\mathbf{Y}^\top - \mathbf{z}\mathbf{v}^\top$$

Now note that the two subtracted terms are exactly the updated WY factors:

$$\mathbf{W}_+\mathbf{Y}_+^\top = [\mathbf{W} \quad \mathbf{z}]\begin{bmatrix}\mathbf{Y}^\top \\ \mathbf{v}^\top\end{bmatrix} = \mathbf{W}\mathbf{Y}^\top + \mathbf{z}\mathbf{v}^\top.$$ ∎

With the correct initialization of $\mathbf{W}$ and $\mathbf{Y}$, we can build the WY representation of successive Householder transformations as shown in Algorithm 3.

Algorithm 3: $\mathbf{W}, \mathbf{Y} \leftarrow$ `buidlWY`$(V, \boldsymbol{\beta})$: Given a set of householder vectors $\{\mathbf{V}[:, i]\}_{i=1}^r$ and their corresponding constants $\{\boldsymbol{\beta}_i\}_{i=1}^r$, form the final $\mathbf{W}$ and $\mathbf{Y}$ factors of the WY representation of $\mathbf{P}_1 \cdots \mathbf{P}_r$, where $\mathbf{P}_i := \mathbf{I}_m - \boldsymbol{\beta}_i\mathbf{v}_i\mathbf{v}_i^\top$

Input: $\mathbf{V} \in \mathbb{R}^{m\times r}$, $\boldsymbol{\beta} \in \mathbb{R}^r$ where $m > r$.
Output: $\mathbf{W}, \mathbf{Y}$

Initialize: $\mathbf{W} := \boldsymbol{\beta}_1\mathbf{V}[:, 1]$ and $\mathbf{Y} := \mathbf{V}[:, 1]$.
for $j = 2 : r$ **do**
  $\mathbf{z} \leftarrow \boldsymbol{\beta}_j\left[\mathbf{V}[:, j] - \mathbf{W}\left(\mathbf{Y}^\top\mathbf{V}[:, j]\right)\right]$
  $\mathbf{W} \leftarrow [\mathbf{W} \quad \mathbf{z}]$ `/* Update` $\mathbf{W}$`. */`
  $\mathbf{Y} \leftarrow [\mathbf{Y} \quad \mathbf{V}[:, j]]$ `/* Update` $\mathbf{Y}$`. */`
  `//` $\mathbf{W}$ `and` $\mathbf{Y}$ `are now` m`-by-`j `matrices.`
return $\mathbf{W}, \mathbf{Y}$

In the traditional HQR, $\mathbf{A}$ is transformed into an upper triangular matrix $\mathbf{R}$ by first computing the Householder transformation to zero out a column below the diagonal, then applying that Householder transformation to all of the remaining columns to the right. For example, the k^{th} Householder transformation finds an $m-k+1$ length Householder vector, $\mathbf{v}_k$, and applies it to an $(m-k+1)$-by-$(n-k)$ matrix. The bulk of FLOPs of this step (line 6 in alg. 2) requires two Level-2

BLAS operations when computed efficiently, which are $\mathbf{C} := \mathbf{v}_k^\top \mathbf{A}_{k:m,k+1:n}\mathbb{R}^{1\times(n-k)}$ and $\mathbf{vC}$, an outer product.

In BQR, the columns of $\mathbf{A}$ are partitioned by groups of r with $\mathbf{A} = [\mathbf{C}_1 \cdots \mathbf{C}_N]$ except for the last block which is $\mathbf{C}_N = \mathbf{A}[:, (N-1)r+1:n]$ and $N = \lceil \frac{n}{r} \rceil$. The first block is triangularized using HQR and the WY representation of $\mathbf{P}_1 \cdots \mathbf{P}_r = \mathbf{I}_m - \mathbf{W}_1\mathbf{Y}_1^\top$ is built at the end. Both of these operations are rich in Level-2 BLAS operations. Then, $\mathbf{I}_m - \mathbf{Y}_1\mathbf{W}_1^\top = \mathbf{P}_r \cdots \mathbf{P}_1$ is applied to $[\mathbf{C}_2 \cdots \mathbf{C}_N]$ with two Level-3 BLAS operations:

1. $A := \mathbf{W}_1^\top [\mathbf{C}_2 \cdots \mathbf{C}_N]$ is a matrix-matrix multiply with m-length inner products.
2. $[\mathbf{C}_2 \cdots \mathbf{C}_N] - \mathbf{Y}_1 A$ is a matrix-matrix multiply with subtraction where the product $\mathbf{Y}_1 A$ computes r-length inner products.

We are now ready to triangularize the second block and update rows $r+1:m$ of $[\mathbf{C}_3 \cdots \mathbf{C}_N]$, and so on. Algorithm 4 shows the pseudoalgorithm of the described procedure and performs approximately $1 - \mathcal{O}(1/N)$ fraction of FLOPs in Level-3 BLAS operations (see section 5.2.3 of [10]).

Algorithm 4: $\mathbf{Q}, \mathbf{R} \leftarrow$ `blockHQR`$(\mathbf{A}, r)$: Perform Householder QR factorization of matrix $\mathbf{A}$ with column partitions of size r.

Input: $\mathbf{A} \in \mathbb{R}^{m\times n}$, $r \in \mathbb{R}$ where $r < n$.
Output: $\mathbf{Q}, \mathbf{R}$

$N = \lceil \frac{n}{r} \rceil$
// Let $n_i = ri$ for $i = 1 : N-1$ and $n_N = n$.
for $i = 1 : N$ **do**
  $\mathbf{V}_i, \boldsymbol{\beta}_i, \mathbf{A}_{n_{i-1}+1:m, n_{i-1}+1:n_i} \leftarrow$ `hhQR`$(\mathbf{A}_{n_{i-1}:m, n_{i-1}+1:n_i})$ /* Algorithm 2 */
  $\mathbf{W}_i, \mathbf{Y}_i \leftarrow$ `buildWY`$(\mathbf{V}_i, \boldsymbol{\beta}_i)$ /* Algorithm 3 */
  if $i < N$ **then**
    $\mathbf{A}_{n_i+1:m, n_i+1:n}$ -= $\mathbf{Y}_i \left(\mathbf{W}_i^\top \mathbf{A}_{n_i+1:m, n_i+1:n}\right)$ /* update the rest: BLAS-3 */
// $\mathbf{A}$ has been transformed into $\mathbf{R} = \mathbf{Q}^\top \mathbf{A}$.
// Now build $\mathbf{Q}$.
$\mathbf{Q} \leftarrow \mathbf{I}$ /* $\mathbf{I}_m$ if full QR, and $\mathbf{I}_{m\times n}$ if thin QR. */
for $i = N : -1 : 1$ **do**
  $\mathbf{Q}_{n_{i-1}+1:m, n_{i-1}+1:n}$-= $\mathbf{W}_i \left(\mathbf{Y}_i^\top \mathbf{Q}_{n_{i-1}+1:m, n_{i-1}+1:n}\right)$ /* BLAS-3 */
return $\mathbf{Q}, \mathbf{A}$

3.3. Block HQR with partitioned rows : Tall-and-Skinny QR (TSQR).

4. Mixed-precision error analysis.

4.1. Round down at the end of the factorization.

4.2. Round down at block-level (BLAS-3).

4.3. Round down at inner-product level (BLAS-2).

5. Numerical Experiments.

6. Conclusion. Though the use of lower precision naturally reduces the bandwidth and storage needs, the development of GPUs to optimize low precision floating point arithmetic have accelerated the interest in half precision and mixed-precision algorithms. Loss in precision, stability, and representable range offset for those advantages, but these shortcomings may have little to no impact in some applications. It may even be possible to navigate around those drawbacks with algorithmic design.

The existing rounding error analysis cannot accurately bound the behavior of mixed-precision arithmetic. We have developed a new framework for mixed-precision rounding error analysis and applied it to HQR, a widely used linear algebra routine, and implemented it in an iterative eigensolver in the context of spectral clustering. The mixed-precision error analysis builds from the inner product routine, which can be applied to many other linear algebra tools as well. The new error bounds more accurately describe how rounding errors are accumulated in mixed-precision settings. We also found that TSQR, a communication-avoiding, easily parallelizable QR factorization algorithm for tall-and-skinny matrices, can outperform HQR in mixed-precision settings for ill-conditioned, extremely overdetermined cases, which suggests that some algorithms are more robust against lower precision arithmetic. As QR factorizations of tall-and-skinny matrices are common in spectral clustering, we experimented with introducing mixed-precision settings into graph partitioning problems. In particular, we applied DBSCAN to the spectral basis of a graph identified via subspace iteration that used our simulated mixed-precision HQR, which yielded clustering results tantamount to results from employing double-precision entirely.

Although this work is focused on QR factorizations and applications in spectral clustering, the mixed precision round-off error analysis can be applied to other tasks and applications that can benefit from employing low precision computations. While the emergence of technology that support low precision floats combats issues dealing with storage, now we need to consider how low precision affects stability of numerical algorithms.

Future work is needed to test larger, more ill-conditioned problems with different mixed-precision settings, and to explore other divide-and-conquer methods like TSQR that can harness parallel capabilities of GPUs while withstanding lower precisions.

REFERENCES

[1] A. Abdelfattah, S. Tomov, and J. Dongarra, *Fast batched matrix multiplication for small sizes using half-precision arithmetic on GPUs*, in 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS), May 2019, pp. 111–122, https://doi.org/10.1109/IPDPS.2019.00022.

[2] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. J. Dongarra, J. Du Croz, S. Hammarling, A. Greenbaum, A. McKenney, and D. Sorensen, *LAPACK Users' Guide (Third Ed.)*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999; also available online from http://www.netlib.org.

[3] J. Appleyard and S. Yokim, *Programming Tensor Cores in CUDA 9*, 2017, https://devblogs.nvidia.com/programming-tensor-cores-cuda-9/ (accessed 2018-07-30).

[4] C. Bischof and C. Van Loan, *The WY Representation for Products of Householder Matrices*, SIAM Journal on Scientific and Statistical Computing, 8 (1987), pp. s2–s13, https://doi.org/10.1137/0908009.

[5] M. Courbariaux, Y. Bengio, and J.-P. David, *Training deep neural networks with low precision multiplications*, arXiv preprint, arXiv:1412.7024, (2014).

[6] M. Courbariaux, J.-P. David, and Y. Bengio, *Low precision storage for deep learning*, arXiv preprint arXiv:1412.7024, (2014).

[7] J. Demmel, I. Dumitriu, and O. Holtz, *Fast linear algebra is stable*, Numerische Mathematik, 108 (2007), pp. 59–91, https://doi.org/10.1007/s00211-007-0114-x, https://arxiv.org/abs/0612264.

[8] J. Demmel, L. Grigori, M. Hoemmen, and J. Langou, *Communication-optimal parallel and sequential QR and LU factorizations*, SIAM Journal on Scientific Computing, 34 (2012), https://doi.org/10.1137/080731992, https://arxiv.org/abs/0808.2664.

[9] M. Fagan, J. Schlachter, K. Yoshii, S. Leyffer, K. Palem, M. Snir, S. M. Wild, and C. Enz, *Overcoming the power wall by exploiting inexactness and emerging COTS architectural features: Trading precision for improving application quality*, in 2016 29th IEEE International System-on-Chip Conference (SOCC), Sep.

2016, pp. 241–246, https://doi.org/10.1109/SOCC.2016.7905477.
[10] G. H. Golub and C. F. Van Loan, *Matrix computations*, JHU press, 4 ed., 2013.
[11] A. Haidar, A. Abdelfattah, M. Zounon, P. Wu, S. Pranesh, S. Tomov, and J. Dongarra, *The Design of Fast and Energy-Efficient Linear Solvers: On the Potential of Half-Precision Arithmetic and Iterative Refinement Techniques*, June 2018, pp. 586–600, https://doi.org/10.1007/978-3-319-93698-7_45.
[12] A. Haidar, S. Tomov, J. Dongarra, and N. J. Higham, *Harnessing GPU tensor cores for fast fp16 arithmetic to speed up mixed-precision iterative refinement solvers*, in Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis, SC '18, Piscataway, NJ, USA, 2018, IEEE Press, pp. 47:1–47:11, https://doi.org/10.1109/SC.2018.00050, https://doi.org/10.1109/SC.2018.00050.
[13] N. J. Higham, *Accuracy and Stability of Numerical Methods*, 2002, https://doi.org/10.2307/2669725.
[14] N. J. Higham and T. Mary, *A New Approach to Probabilistic Rounding Error Analysis*, SIAM Journal on Scientific Computing, 41 (2019), pp. A2815–A2835, https://doi.org/10.1137/18M1226312, https://epubs.siam.org/doi/10.1137/18M1226312.
[15] N. J. Higham and S. Pranesh, *Simulating Low Precision Floating-Point Arithmetic*, SIAM Journal on Scientific Computing, 41 (2019), pp. C585–C602, https://doi.org/10.1137/19M1251308, https://epubs.siam.org/doi/10.1137/19M1251308.
[16] A. S. Householder, *Unitary triangularization of a nonsymmetric matrix*, Journal of the ACM (JACM), 5 (1958), pp. 339–342.
[17] I. C. F. Ipsen and H. Zhou, *Probabilistic Error Analysis for Inner Products*, (2019), http://arxiv.org/abs/1906.10465, https://arxiv.org/abs/1906.10465.
[18] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu, *Mixed precision training*, in International Conference on Learning Representations, 2018, https://openreview.net/forum?id=r1gs9JgRZ.
[19] R. Schreiber and C. Van Loan, *A Storage-Efficient WY Representation for Products of Householder Transformations*, SIAM Journal on Scientific and Statistical Computing, 10 (1989), pp. 53–57, https://doi.org/10.1137/0910005.
[20] G. Tagliavini, S. Mach, D. Rossi, A. Marongiu, and L. Benin, *A transprecision floating-point platform for ultra-low power computing*, in 2018 Design, Automation Test in Europe Conference Exhibition (DATE), March 2018, pp. 1051–1056, https://doi.org/10.23919/DATE.2018.8342167.
[21] U. von Luxburg, *A tutorial on spectral clustering*, Statistics and Computing, 17 (2007), pp. 395–416, https://doi.org/10.1007/s11222-007-9033-z, https://doi.org/10.1007/s11222-007-9033-z.