Notes on Householder QR Factorization

Robert A. van de Geijn
Department of Computer Science
The University of Texas at Austin
Austin, TX 78712
rvdg@cs.utexas.edu

September 21, 2014

1 Motivation

A fundamental problem to avoid in numerical codes is the situation where one starts with large values and one ends up with small values with large relative errors in them. This is known as catastrophic cancellation. The Gram-Schmidt algorithms can inherently fall victim to this: column a_j is successively reduced in length as components in the directions of $\{q_0, \ldots, q_{j-1}\}$ are subtracted, leaving a small vector if a_j was almost in the span of the first j columns of A. Application of a unitary transformation to a matrix or vector inherently preserves length. Thus, it would be beneficial if the QR factorization can be implementated as the successive application of unitary transformations. The Householder QR factorization accomplishes this.

The first fundamental insight is that the product of unitary matrices is itself unitary. If, given $A \in \mathbb{C}^{m\times n}$ (with $m \geq n$), one could find a sequence of unitary matrices, $\{H_0, \ldots, H_{n-1}\}$, such that

$$H_{n-1} \cdots H_0 A = \left(\frac{R}{0} \right),$$

where $R \in \mathbb{C}^{m\times n}$ is upper triangular, then

$$H_{n-1} \cdots H_0 A = \underbrace{H_0 \cdots H_{n-1}}_{Q} \left(\frac{R}{0} \right) = Q \left(\frac{R}{0} \right) = \left(\begin{array}{c|c} Q_L & Q_R \end{array} \right) \left(\frac{R}{0} \right) = Q_L R,$$

where Q_L equals the first n columns of A. Then $A = Q_L R$ is the QR factorization of A. The second fundamental insight will be that the desired unitary transformations $\{H_0, \ldots, H_{n-1}\}$ can be computed and applied cheaply.

2 Householder Transformations (Reflectors)

2.1 The general case

In this section we discuss *Householder transformations*, also referred to as *reflectors*.

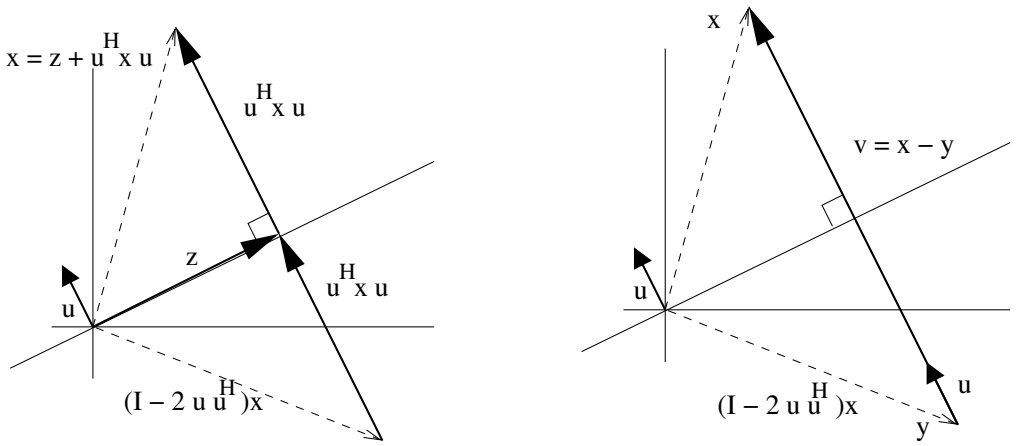

Figure 1: Left: Illustration that shows how, given vectors x and unit length vector u, the subspace orthogonal to u becomes a mirror for reflecting x represented by the transformation $(I - 2uu^H)$. Right: Illustration that shows how to compute u given vectors x and y with $\|x\|_2 = \|y\|_2$.

Definition 1. Let $u \in \mathbb{C}^n$ be a vector of unit length ($\|u\|_2 = 1$). Then $H = I - 2uu^H$ is said to be a reflector or Householder transformation.

We observe:

- Any vector z that is perpendicular to u is left unchanged:

$$(I - 2uu^H)z = z - 2u(u^H z) = z.$$

- Any vector x can be written as $x = z + u^H xu$ where z is perpendicular to u and $u^H xu$ is the component of x in the direction of u. Then

$$\begin{aligned}(I - 2uu^H)x &= (I - 2uu^H)(z + u^H xu) = z + u^H xu - 2u \underbrace{u^H z}_{0} - 2uu^H u^H xu \\ &= z + u^H xu - 2u^H x \underbrace{u^H u}_{1} u = z - u^H xu.\end{aligned}$$

This can be interpreted as follows: The space perpendicular to u acts as a "mirror": any vector in that space (along the mirror) is not reflected, while any other vector has the component that is orthogonal to the space (the component outside, orthogonal to, the mirror) reversed in direction, as illustrated in Figure 1. Notice that a reflection preserves the length of the vector.

Exercise 2. Show that if H is a reflector, then

- $HH = I$ (reflecting the reflection of a vector results in the original vector),

- $H = H^H$, and
- $H^H H = I$ (a reflection is a unitary matrix and thus preserves the norm).

Next, let us ask the question of how to reflect a given $x \in \mathbb{C}^n$ into another vector $y \in \mathbb{C}^n$ with $\|x\|_2 = \|y\|_2$. In other words, how do we compute vector u so that $(I - 2uu^H)x = y$. From our discussion above, we need to find a vector u that is perpendicular to the space with respect to which we will reflect. From Figure 1(right) we notice that the vector from y to x, $v = x - y$, is perpendicular to the desired space. Thus, u must equal a unit vector in the direction v: $u = v/\|v\|_2$.

Remark 3. In subsequent discussion we will prefer to give Householder transformations as $I - uu^H/\tau$, where $\tau = u^H u/2$ so that u needs no longer be a unit vector, just a direction. The reason for this will become obvious later.

In the next subsection, we will need to find a Householder transformation H that maps a vector x to a multiple of the first unit basis vector (e_0).

Let us first discuss how to find H in the case where $x \in \mathbb{R}^n$. We seek v so that $(I - \frac{2}{v^T v} vv^T)x = \pm\|x\|_2 e_0$. Since the resulting vector that we want is $y = \pm\|x\|_2 e_0$, we must choose $v = x - y = x \mp \|x\|_2 e_0$.

Exercise 4. Show that if $x \in \mathbb{R}^n$, $v = x \mp \|x\|_2 e_0$, and $\tau = v^T v/2$ then $(I - \frac{1}{\tau} vv^T)x = \pm\|x\|_2 e_0$.

In practice, we choose $v = x + \text{sign}(\chi_1)\|x\|_2 e_0$ where χ_1 denotes the first element of x. The reason is as follows: the first element of v, ν_1, will be $\nu_1 = \chi_1 \mp \|x\|_2$. If χ_1 is positive and $\|x\|_2$ is almost equal to χ_1, then $\chi_1 - \|x\|_2$ is a small number and if there is error in χ_1 and/or $\|x\|_2$, this error becomes large *relative* to the result $\chi_1 - \|x\|_2$, due to catastrophic cancellation. Regardless of whether χ_1 is positive or negative, we can avoid this by choosing $x = \chi_1 + \text{sign}(\chi_1)\|x\|_2 e_0$.

2.2 As implemented for the Householder QR factorization (real case)

Next, we discuss a slight variant on the above discussion that is used in practice. To do so, we view x as a vector that consists of its first element, χ_1, and the rest of the vector, x_2: More precisely, partition

$$x = \begin{pmatrix} \chi_1 \\ \hline x_2 \end{pmatrix},$$

where χ_1 equals the first element of x and x_2 is the rest of x. Then we will wish to find a Householder vector $u = \begin{pmatrix} 1 \\ u_2 \end{pmatrix}$ so that

$$\left(I - \frac{1}{\tau} \begin{pmatrix} 1 \\ u_2 \end{pmatrix} \begin{pmatrix} 1 \\ u_2 \end{pmatrix}^T \right) \begin{pmatrix} \chi_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \pm \|x\|_2 \\ 0 \end{pmatrix}.$$

Notice that y in the previous discussion equals the vector $\begin{pmatrix} \pm\|x\|_2 \\ 0 \end{pmatrix}$, so the direction of u is given by

$$v = \begin{pmatrix} \chi_1 \mp \|x\|_2 \\ x_2 \end{pmatrix}.$$

We now wish to normalize this vector so its first entry equals "1":

$$u = \frac{v}{\nu_1} = \frac{1}{\chi_1 \mp \|x\|_2} \begin{pmatrix} \chi_1 \mp \|x\|_2 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ x_2/\nu_1 \end{pmatrix}.$$

where $\nu_1 = \chi_1 \mp \|x\|_2$ equals the first element of v. (Note that if $\nu_1 = 0$ then u_2 can be set to 0.)

2.3 The complex case (optional)

Next, let us work out the complex case, dealing explicitly with x as a vector that consists of its first element, χ_1, and the rest of the vector, x_2: More precisely, partition

$$x = \begin{pmatrix} \chi_1 \\ x_2 \end{pmatrix},$$

where χ_1 equals the first element of x and x_2 is the rest of x. Then we will wish to find a Householder vector $u = \begin{pmatrix} 1 \\ u_2 \end{pmatrix}$ so that

$$\left(I - \frac{1}{\tau} \begin{pmatrix} 1 \\ u_2 \end{pmatrix} \begin{pmatrix} 1 \\ u_2 \end{pmatrix}^H \right) \begin{pmatrix} \chi_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \circledpm \|x\|_2 \\ 0 \end{pmatrix}.$$

Here $\circledpm$ denotes a complex scalar on the complex unit circle. By the same argument as before

$$v = \begin{pmatrix} \chi_1 - \circledpm \|x\|_2 \\ x_2 \end{pmatrix}.$$

We now wish to normalize this vector so its first entry equals "1":

$$u = \frac{v}{\|v\|_2} = \frac{1}{\chi_1 - \circledpm \|x\|_2} \begin{pmatrix} \chi_1 - \circledpm \|x\|_2 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ x_2/\nu_1 \end{pmatrix}.$$

where $\nu_1 = \chi_1 - \circledpm \|x\|_2$. (If $\nu_1 = 0$ then we set u_2 to 0.)

Algorithm: $\left[\begin{pmatrix}\rho\\ u_2\end{pmatrix}, \tau\right] = \text{HOUSEV}\left(\begin{pmatrix}\chi_1\\ x_2\end{pmatrix}\right)$	
	$\chi_2 := \|x_2\|_2$
	$\alpha := \left\|\begin{pmatrix}\chi_1\\ \chi_2\end{pmatrix}\right\|_2 (= \|x\|_2)$
$\rho = -\text{sign}(\chi_1)\|x\|_2$	$\rho := -\text{sign}(\chi_1)\alpha$
$\nu_1 = \chi_1 + \text{sign}(\chi_1)\|x\|_2$	$\nu_1 := \chi_1 - \rho$
$u_2 = x_2/\nu_1$	$u_2 := x_2/\nu_1$
	$\chi_2 = \chi_2/\|\nu_1\| (= \|u_2\|_2)$
$\tau = (1 + u_2^H u_2)/2$	$\tau = (1 + \chi_2^2)/2$

Figure 2: Computing the Householder transformation. Left: simple formulation. Right: efficient computation. **Note: I have not completely double-checked these formulas for the complex case. They work for the real case.**

Exercise 5. Verify that

$$\left(I - \frac{1}{\tau}\begin{pmatrix}1\\ u_2\end{pmatrix}\begin{pmatrix}1\\ u_2\end{pmatrix}^H\right)\begin{pmatrix}\chi_1\\ x_2\end{pmatrix} = \begin{pmatrix}\rho\\ 0\end{pmatrix}$$

where $\tau = u^H u/2 = (1 + u_2^H u_2)/2$ and $\rho = \oplus\!\!\!\!-\|x\|_2$.
Hint: $\rho\bar{\rho} = |\rho|^2 = \|x\|_2^2$ since H preserves the norm. Also, $\|x\|_2^2 = |\chi_1|^2 + \|x_2\|_2^2$ and $\sqrt{\frac{z}{\bar{z}}} = \frac{z}{|z|}$.

Again, the choice $\oplus\!\!\!\!-$ is important. For the complex case we choose $\oplus\!\!\!\!- = -sign(\chi_1) = \frac{\chi_1}{|\chi_1|}$

2.4 A routine for computing the Householder vector

We will refer to the vector

$$\begin{pmatrix}1\\ u_2\end{pmatrix}$$

as the Householder vector that reflects x into $\oplus\!\!\!\!-\|x\|_2 e_0$ and introduce the notation

$$\left[\begin{pmatrix}\rho\\ u_2\end{pmatrix}, \tau\right] := \text{HOUSEV}\left(\begin{pmatrix}\chi_1\\ x_2\end{pmatrix}\right)$$

as the computation of the above mentioned vector u_2, and scalars ρ and τ, from vector x. We will use the notation $H(x)$ for the transformation $I - \frac{1}{\tau}uu^H$ where u and τ are computed by HOUSEV(x).

3 Householder QR Factorization

Let A be an $m \times n$ with $m \geq n$. We will now show how to compute $A \rightarrow QR$, the QR factorization, as a sequence of Householder transformations applied to A, which eventually zeroes out all elements of that matrix below the diagonal. The process is illustrated in Figure 3.

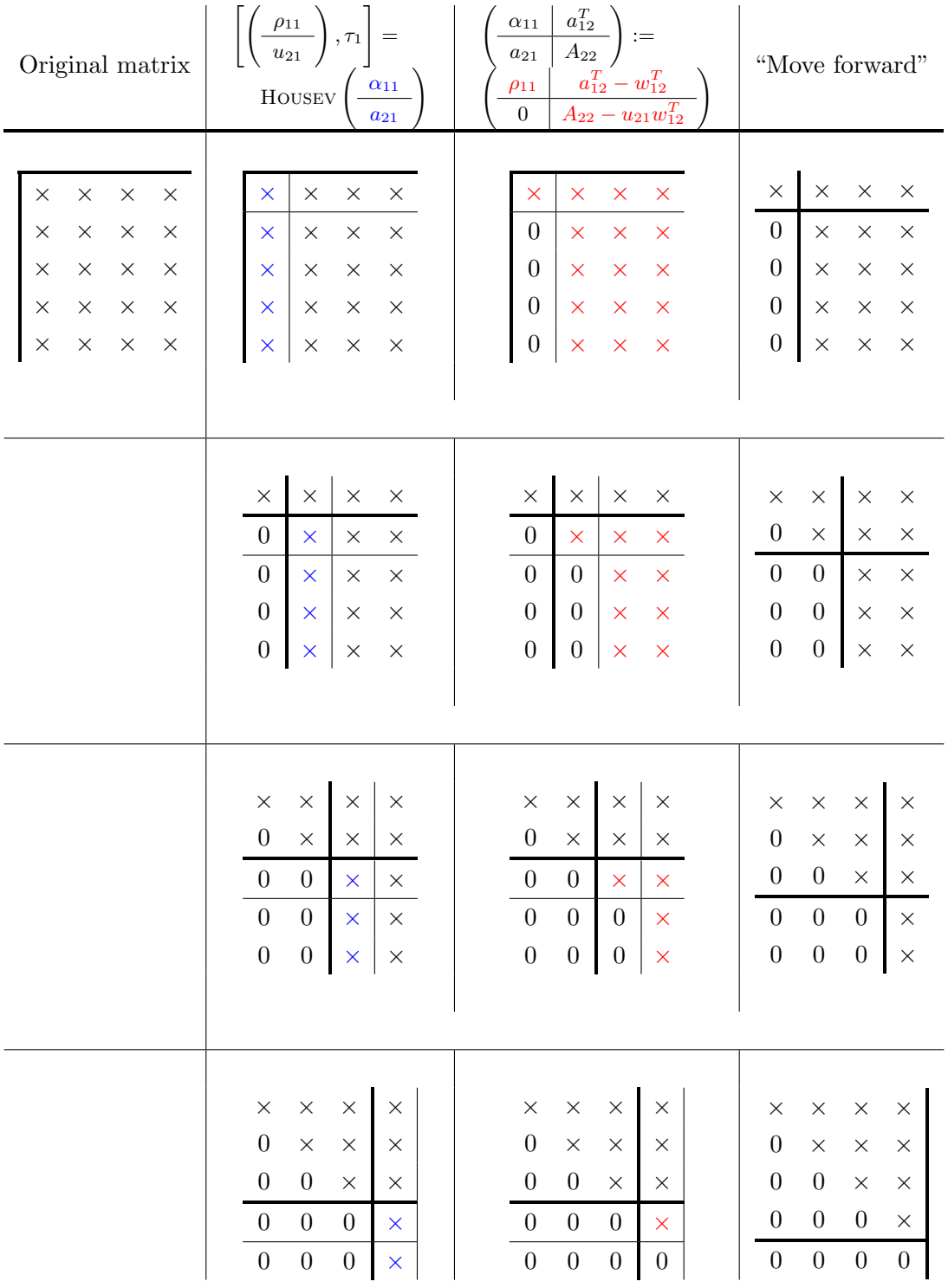

Figure 3: Illustration of Householder QR factorization.

In the first iteration, we partition

$$A \rightarrow \left(\begin{array}{c|c} \alpha_{11} & a_{12}^T \\ \hline a_{21} & A_{22} \end{array}\right).$$

Let

$$\left[\left(\frac{\rho_{11}}{u_{21}}\right), \tau_1\right] = \textsc{Housev}\left(\frac{\alpha_{11}}{a_{21}}\right)$$

be the Householder transform computed from the first column of A. Then applying this Householder transform to A yields

$$\begin{array}{rcl} \left(\begin{array}{c|c} \alpha_{11} & a_{12}^T \\ \hline a_{21} & A_{22} \end{array}\right) & := & \left(I - \frac{1}{\tau_1}\left(\frac{1}{u_2}\right)\left(\frac{1}{u_2}\right)^H\right)\left(\begin{array}{c|c} \alpha_{11} & a_{12}^T \\ \hline a_{21} & A_{22} \end{array}\right) \\ & = & \left(\begin{array}{c|c} \rho_{11} & a_{12}^T - w_{12}^T \\ \hline 0 & A_{22} - u_{21} w_{12}^T \end{array}\right), \end{array}$$

where $w_{12}^T = (a_{12}^T + u_{21}^H A_{22})/\tau_1$. Computation of a full QR factorization of A will now proceed with the updated matrix A_{22}.

Now let us assume that after k iterations of the algorithm matrix A contains

$$A \rightarrow \left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & A_{BR} \end{array}\right) = \left(\begin{array}{c|c|c} R_{00} & r_{01} & R_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & a_{21} & A_{22} \end{array}\right),$$

where R_{TL} and R_{00} are $k \times k$ upper triangular matrices. Let

$$\left[\left(\frac{\rho_{11}}{u_{21}}\right), \tau_1\right] = \textsc{Housev}\left(\frac{\alpha_{11}}{a_{21}}\right).$$

and update

$$\begin{array}{rcl} A & := & \left(\begin{array}{c|c} I & 0 \\ \hline 0 & \left(I - \frac{1}{\tau_1}\left(\frac{1}{u_2}\right)\left(\frac{1}{u_2}\right)^H\right) \end{array}\right)\left(\begin{array}{c|c|c} R_{00} & r_{01} & R_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & a_{21} & A_{22} \end{array}\right) \\ & = & \left(I - \frac{1}{\tau_1}\left(\begin{array}{c} 0 \\ \hline 1 \\ \hline u_2 \end{array}\right)\left(\begin{array}{c} 0 \\ \hline 1 \\ \hline u_2 \end{array}\right)^H\right)\left(\begin{array}{c|c|c} R_{00} & r_{01} & R_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ \hline 0 & a_{21} & A_{22} \end{array}\right) \\ & = & \left(\begin{array}{c|c|c} R_{00} & r_{01} & R_{02} \\ \hline 0 & \rho_{11} & a_{12}^T - w_{12}^T \\ \hline 0 & 0 & A_{22} - u_{21} w_{12}^T \end{array}\right), \end{array}$$

where again $w_{12}^T = (a_{12}^T + u_{21}^H A_{22})/\tau_1$.

Let

$$H_k = \left(I - \frac{1}{\tau_1} \left(\begin{array}{c} 0_k \\ \hline 1 \\ \hline u_{21} \end{array} \right) \left(\begin{array}{c} 0_k \\ \hline 1 \\ \hline u_{21} \end{array} \right)^H \right)$$

be the Householder transform so computed during the $(k+1)$st iteration. Then upon completion matrix A contains

$$R = \left(\begin{array}{c} R_{TL} \\ \hline 0 \end{array} \right) = H_{n-1} \cdots H_1 H_0 \hat{A}$$

where $\hat{A}$ denotes the original contents of A and R_{TL} is an upper triangular matrix. Rearranging this we find that

$$\hat{A} = H_0 H_1 \cdots H_{n-1} R$$

which shows that if $Q = H_0 H_1 \cdots H_{n-1}$ then $\hat{A} = QR$.

Exercise 6. Show that

$$\left(\begin{array}{c|c} I & 0 \\ \hline 0 & \left(I - \frac{1}{\tau_1} \left(\begin{array}{c} 1 \\ \hline u_2 \end{array} \right) \left(\begin{array}{c} 1 \\ \hline u_2 \end{array} \right)^H \right) \end{array} \right) = \left(I - \frac{1}{\tau_1} \left(\begin{array}{c} 0 \\ \hline 1 \\ \hline u_2 \end{array} \right) \left(\begin{array}{c} 0 \\ \hline 1 \\ \hline u_2 \end{array} \right)^H \right).$$

Typically, the algorithm overwrites the original matrix A with the upper triangular matrix, and at each step u_{21} is stored over the elements that become zero, thus overwriting a_{21}. (It is for this reason that the first element of u was normalized to equal "1".) In this case Q is usually not explicitly formed as it can be stored as the separate Householder vectors below the diagonal of the overwritten matrix. The algorithm that overwrites A in this manner is given in Fig. 4.

We will let

$$[\{U \backslash R\}, t] = \text{HouseholderQR}(A)$$

denote the operation that computes the QR factorization of $m \times n$ matrix A, with $m \geq n$, via Householder transformations. It returns the Householder vectors and matrix R in the first argument and the vector of scalars "τ_i" that are computed as part of the Householder transformations in t.

Theorem 7. Given $A \in \mathbb{C}^{m \times n}$ the cost of the algorithm in Figure 4 is given by

$$C_{\text{HQR}}(m,n) \approx 2mn^2 - \frac{2}{3}n^3 \text{ flops.}$$

Proof: The bulk of the computation is in $w_{12}^T = (a_{12}^T + u_{21}^H A_{22})/\tau_1$ and $A_{22} - u_{21} w_{12}^T$. During the kth iteration (when R_{TL} is $k \times k$), this means a matrix-vector multiplication ($u_{21}^H A_{22}$) and rank-1

Algorithm: $[A, t] = \text{HOUSEHOLDERQR}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$ and $t \rightarrow \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right)$

where A_{TL} is 0×0 and t_T has 0 elements

while $n(A_{BR}) \neq 0$ **do**

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right) \text{ and } \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right) \rightarrow \left(\begin{array}{c} t_0 \\ \hline \tau_1 \\ \hline t_2 \end{array}\right)$$

where α_{11} and τ_1 are scalars

$$\left[\left(\begin{array}{c} \alpha_{11} \\ \hline a_{21} \end{array}\right), \tau_1\right] := \left[\left(\begin{array}{c} \rho_{11} \\ \hline u_{21} \end{array}\right), \tau_1\right] = \text{HOUSEV}\left(\begin{array}{c} \alpha_{11} \\ \hline a_{21} \end{array}\right)$$

Update $\left(\begin{array}{c} a_{12}^T \\ \hline A_{22} \end{array}\right) := \left(I - \frac{1}{\tau_1}\left(\begin{array}{c} 1 \\ \hline u_{21} \end{array}\right)\left(\begin{array}{c|c} 1 & u_{21}^H \end{array}\right)\right)\left(\begin{array}{c} a_{12}^T \\ \hline A_{22} \end{array}\right)$

via the steps

- $w_{12}^T := (a_{12}^T + a_{21}^H A_{22})/\tau_1$
- $\left(\begin{array}{c} a_{12}^T \\ \hline A_{22} \end{array}\right) := \left(\begin{array}{c} a_{12}^T - w_{12}^T \\ \hline A_{22} - a_{21} w_{12}^T \end{array}\right)$

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right) \text{ and } \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right) \leftarrow \left(\begin{array}{c} t_0 \\ \hline \tau_1 \\ \hline t_2 \end{array}\right)$$

endwhile

Figure 4: Unblocked Householder transformation based QR factorization.

update with matrix A_{22} which is of size approximately $(m-k) \times (n-k)$ for a cost of $4(m-k)(n-k)$ flops. Thus the total cost is approximately

$$\begin{aligned} \sum_{k=0}^{n-1} 4(m-k)(n-k) &= 4\sum_{j=0}^{n-1}(m-n+j)j = 4(m-n)\sum_{j=0}^{n-1} j + 4\sum_{j=0}^{n-1} j^2 \\ &= 2(m-n)n(n-1) + 4\sum_{j=0}^{n-1} j^2 \\ &\approx 2(m-n)n^2 + 4\int_0^n x^2 dx = 2mn^2 - 2n^3 + \frac{4}{3}n^3 = 2mn^2 - \frac{2}{3}n^3. \end{aligned}$$

□

4 Forming Q

Given $A \in \mathbb{C}^{m \times n}$, let $[A, t] = \text{HouseholderQR}(A)$ yield the matrix A with the Householder vectors stored below the diagonal, R stored on and above the diagonal, and the τ_i stored in vector t. We

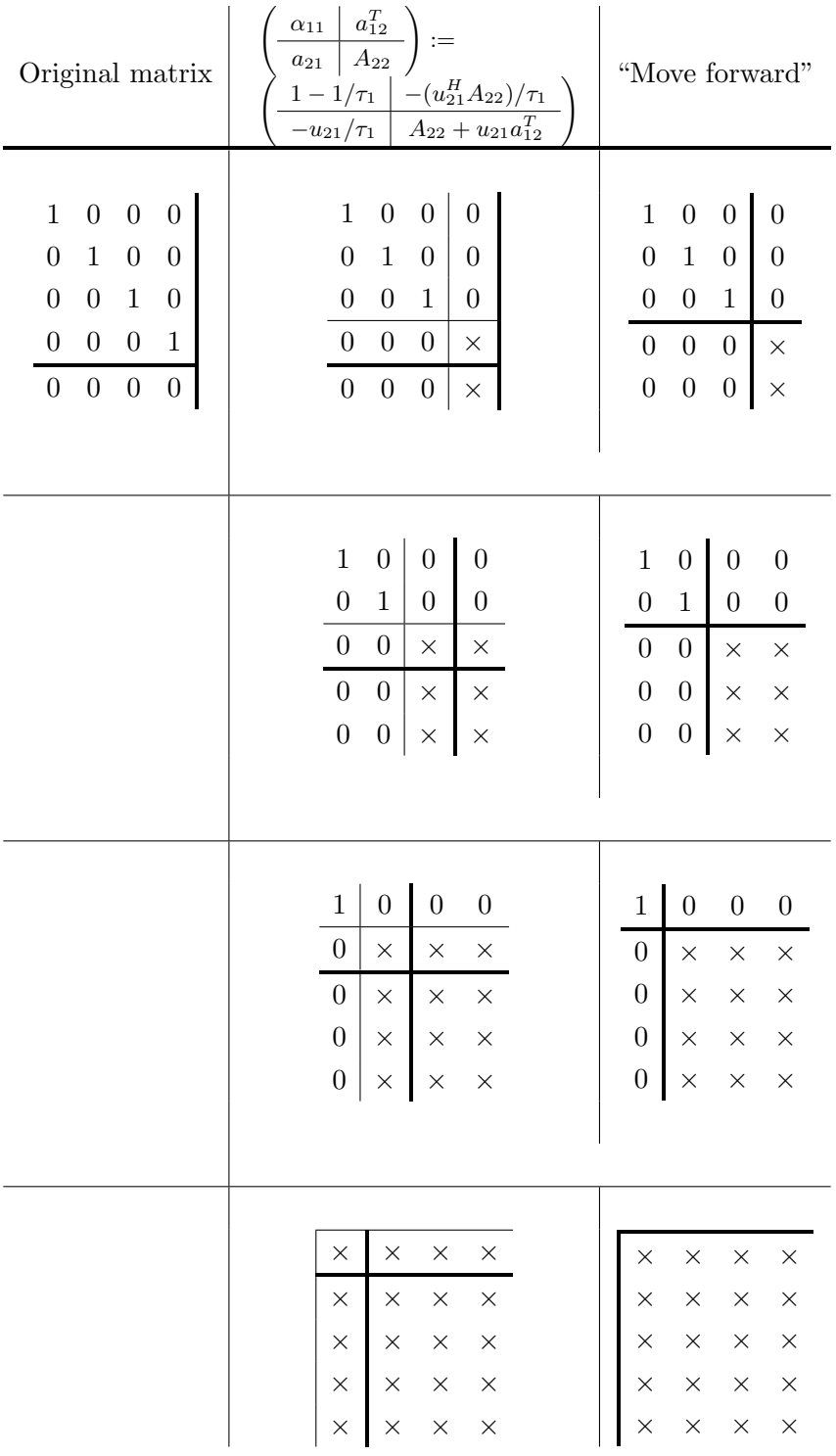

Figure 5: Illustration of the computation of Q.

now discuss how to form the first n columns of $Q = H_0 H_1 \cdots H_{n-1}$. The computation is illustrated in Figure **??**.

Notice that to pick out the first n columns we must form

$$Q \begin{pmatrix} I_{n\times n} \\ 0 \end{pmatrix} = H_0 \cdots H_{n-1} \begin{pmatrix} I_{n\times n} \\ 0 \end{pmatrix} = H_0 \cdots H_{k-1} \underbrace{H_k \cdots H_{n-1} \left(\begin{array}{c} I_{n\times n} \\ \hline 0 \end{array}\right)}_{B_k}.$$

where B_k is defined as indicated.

Lemma 8. B_k has the form

$$B_k = H_k \cdots H_{n-1} \left(\begin{array}{c} I_{n\times n} \\ \hline 0 \end{array}\right) = \left(\begin{array}{c|c} I_{k\times k} & 0 \\ \hline 0 & \tilde{B}_k \end{array}\right).$$

Proof: The proof of this is by induction on k:

- **Base case:** $k = n$. Then $B_n = \left(\begin{array}{c} I_{n\times n} \\ \hline 0 \end{array}\right)$, which has the desired form.
- **Inductive step:** Assume the result is true for B_k. We show it is true for B_{k-1}:

$$\begin{aligned}
B_{k-1} &= H_{k-1} H_k \cdots H_{n-1} \left(\begin{array}{c} I_{n\times n} \\ \hline 0 \end{array}\right) = H_{k-1} B_k = H_{k-1} \left(\begin{array}{c|c} I_{k\times k} & 0 \\ \hline 0 & \tilde{B}_k \end{array}\right). \\
&= \left(\begin{array}{c|c} I_{(k-1)\times(k-1)} & 0 \\ \hline 0 & I - \frac{1}{\tau_k} \left(\begin{array}{c} 1 \\ \hline u_k \end{array}\right) \left(\begin{array}{c|c} 1 & u_k^H \end{array}\right) \end{array}\right) \left(\begin{array}{c|c|c} I_{(k-1)\times(k-1)} & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & \tilde{B}_k \end{array}\right) \\
&= \left(\begin{array}{c|c} I_{(k-1)\times(k-1)} & 0 \\ \hline 0 & \left(I - \frac{1}{\tau_k} \left(\begin{array}{c} 1 \\ \hline u_k \end{array}\right) \left(\begin{array}{c|c} 1 & u_k^H \end{array}\right)\right) \left(\begin{array}{c|c} 1 & 0 \\ \hline 0 & \tilde{B}_k \end{array}\right) \end{array}\right) \\
&= \left(\begin{array}{c|c} I_{(k-1)\times(k-1)} & 0 \\ \hline 0 & \left(\begin{array}{c|c} 1 & 0 \\ \hline 0 & \tilde{B}_k \end{array}\right) - \left(\begin{array}{c} 1 \\ \hline u_k \end{array}\right) \left(\begin{array}{c|c} 1/\tau_k & y_k^T \end{array}\right) \end{array}\right) \quad \text{where } y_k^T = u_k^H \tilde{B}_k / \tau_k \\
&= \left(\begin{array}{c|c} I_{(k-1)\times(k-1)} & 0 \\ \hline 0 & \left(\begin{array}{c|c} 1 - 1/\tau_k & -y_k^T \\ \hline -u_k/\tau_k & B_k - u_k y_k^T \end{array}\right) \end{array}\right) \\
&= \left(\begin{array}{c|c|c} I_{(k-1)\times(k-1)} & 0 & 0 \\ \hline 0 & 1 - 1/\tau_k & -y_k^T \\ \hline 0 & -u_k/\tau_k & B_k - u_k y_k^T \end{array}\right) = \left(\begin{array}{c|c} I_{(k-1)\times(k-1)} & 0 \\ \hline 0 & \tilde{B}_{k-1} \end{array}\right).
\end{aligned}$$

Algorithm: $[A] = \text{FORMQ}(A, t)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$ and $t \rightarrow \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right)$

where A_{TL} is $n(A) \times n(A)$ and t_T has $n(A)$ elements

while $n(A_{TR}) \neq 0$ **do**

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right) \text{ and } \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right) \rightarrow \left(\begin{array}{c} t_0 \\ \hline \tau_1 \\ \hline t_2 \end{array}\right)$$

where α_{11} and τ_1 are scalars

Update $\left(\begin{array}{c|c} \alpha_{11} & a_{12}^T \\ \hline a_{21} & A_{22} \end{array}\right) := \left(I - \frac{1}{\tau_1}\left(\begin{array}{c} 1 \\ \hline u_{21} \end{array}\right)\left(\begin{array}{c|c} 1 & u_{21}^H \end{array}\right)\right)\left(\begin{array}{c|c} 1 & 0 \\ \hline 0 & A_{22} \end{array}\right)$

via the steps

- $\alpha_{11} := 1 - 1/\tau_1$
- $a_{12}^T := -(a_{21}^H A_{22})/\tau_1$
- $A_{22} := A_{22} + a_{21} a_{12}^T$
- $a_{21} := -a_{21}/\tau_1$

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right) \text{ and } \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right) \leftarrow \left(\begin{array}{c} t_0 \\ \hline \tau_1 \\ \hline t_2 \end{array}\right)$$

endwhile

Figure 6: Algorithm for overwriting A with Q from the Householder transformations stored as Householder vectors below the diagonal of A (as produced by $[A, t] = \text{HouseholderQR}(A)$).

- **By the Principle of Mathematical Induction** the result holds for $B_0, \ldots, B_n$.

□

Theorem 9. Given $[A, t] = \text{HouseholderQR}(A)$ from Figure 4, the algorithm in Figure 6 overwrites A with the first $n = n(A)$ columns of Q as defined by the Householder transformations stored below the diagonal of A and in the vector t.

Proof: The algorithm is justified by the proof of Lemma 8. □

Theorem 10. Given $A \in \mathbb{C}^{m \times n}$ the cost of the algorithm in Figure 6 is given by

$$C_{\text{FormQ}}(m, n) \approx 2mn^2 - \frac{2}{3}n^3 \text{ flops.}$$

Algorithm: $[y] = \text{APPLYQT}(A, t, y)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$, $t \rightarrow \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right)$, and $y \rightarrow \left(\begin{array}{c} y_T \\ \hline y_B \end{array}\right)$

where A_{TL} is 0×0 and t_T, y_T has 0 elements

while $n(A_{BR}) \neq 0$ **do**

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right), \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right) \rightarrow \left(\begin{array}{c} t_0 \\ \hline \tau_1 \\ \hline t_2 \end{array}\right), \text{ and } \left(\begin{array}{c} y_T \\ \hline y_B \end{array}\right) \rightarrow \left(\begin{array}{c} y_0 \\ \hline \psi_1 \\ \hline y_2 \end{array}\right)$$

where α_{11}, τ_1, and ψ_1 are scalars

Update $\left(\begin{array}{c} \psi_1 \\ \hline y_2 \end{array}\right) := \left(I - \frac{1}{\tau_1}\left(\begin{array}{c} 1 \\ \hline u_{21} \end{array}\right)\left(\begin{array}{c|c} 1 & u_{21}^H \end{array}\right)\right)\left(\begin{array}{c} \psi_1 \\ \hline y_2 \end{array}\right)$

via the steps

- $\omega_1 := (\psi_1 + a_{21}^H y_2)/\tau_1$
- $\left(\begin{array}{c} \psi_1 \\ \hline y_2 \end{array}\right) := \left(\begin{array}{c} \psi_1 - \omega_1 \\ \hline y_2 - \omega_1 u_2 \end{array}\right)$

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right), \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right) \leftarrow \left(\begin{array}{c} t_0 \\ \hline \tau_1 \\ \hline t_2 \end{array}\right), \text{ and } \left(\begin{array}{c} y_T \\ \hline y_B \end{array}\right) \leftarrow \left(\begin{array}{c} y_0 \\ \hline \psi_1 \\ \hline y_2 \end{array}\right)$$

endwhile

Figure 7: Algorithm for computing $y := H_{n-1} \cdots H_0 y$ given the output from routine HouseholderQR.

Proof: Hence the proof for Theorem 7 can be easily modified to establish this result. □

Exercise 11. If $m = n$ then Q could be accumulated by the sequence

$$Q = (\cdots((IH_0)H_1)\cdots H_{n-1}).$$

Give a high-level reason why this would be (much) more expensive than the algorithm in Figure 6.

5 Applying Q^H

In a future Note, we will see that the QR factorization is used to solve the linear least-squares problem. To do so, we need to be able to compute $\hat{y} = Q^H y$ where $Q^H = H_{n-1} \cdots H_0$.

Let us start by computing $H_0 y$:

$$\left(I - \frac{1}{\tau_1} \left(\frac{1}{u_2} \right) \left(\frac{1}{u_2} \right)^H \right) \left(\frac{\psi_1}{y_2} \right) = \left(\frac{\psi_1}{y_2} \right) - \left(\frac{1}{u_2} \right) \underbrace{\left(\frac{1}{u_2} \right)^H \left(\frac{\psi_1}{y_2} \right) / \tau_1}_{\omega_1}$$

$$= \left(\frac{\psi_1}{y_2} \right) - \omega_1 \left(\frac{1}{u_2} \right) = \left(\frac{\psi_1 - \omega_1}{y_2 - \omega_1 u_2} \right).$$

More generally, let us compute $H_k y$:

$$\left(I - \frac{1}{\tau_1} \left(\frac{0}{\frac{1}{u_2}} \right) \left(\frac{0}{\frac{1}{u_2}} \right)^H \right) \left(\frac{y_0}{\frac{\psi_1}{y_2}} \right) = \left(\frac{y_0}{\frac{\psi_1 - \omega_1}{y_2 - \omega_1 u_2}} \right),$$

where $\omega_1 = (\psi_1 + u_2^H y_2)/\tau_1$. This motivates the algorithm in Figure 7 for computing $y := H_{n-1} \cdots H_0 y$ given the output matrix A and vector t from routine HouseholderQR.

The cost of this algorithm can be analyzed as follows: When y_T is of length k, the bulk of the computation is in an inner product with vectors of length $m - k$ (to compute ω_1) and an axpy operation with vectors of length $m - k$ to subsequently update ψ_1 and y_2. Thus, the cost is approximately given by

$$\sum_{k=0}^{n-1} 4(m-k) \approx 4mn - 2n^2.$$

Notice that this is *much* cheaper than forming Q and then multiplying.