8.3 Householder QR factorization

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 cancelation. 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. Thus, if, given $A \in \mathbb{C}^{m\times n}$ 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 is upper triangular, then

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

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 is that the desired unitary transformations $H_0, \ldots, H_{n-1}$ can be computed and applied cheaply.

8.3.1 Householder transformations (reflectors)

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

Definition 8.6 *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 $x = z + u^H x u$ where z is perpendicular to u and $u^H x u$ is the component of x in the direction of u. Then

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

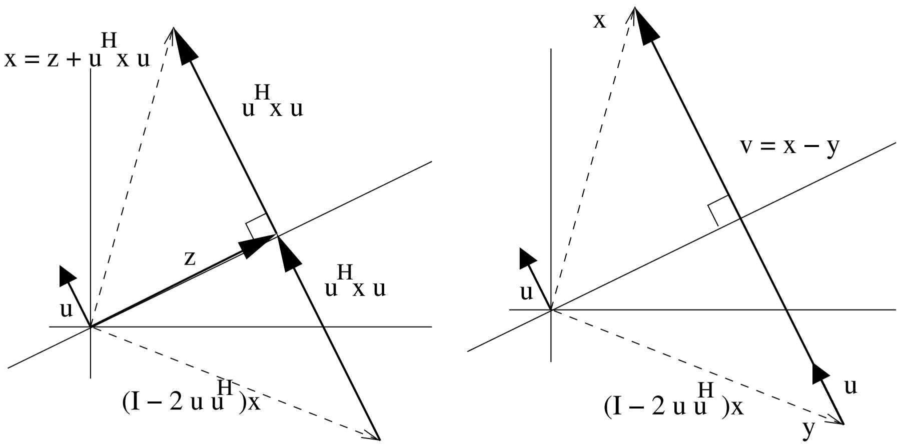

Figure 8.8: ??

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 Fig. 8.8. Notice that a reflection preserves the length of the vector.

Exercise 8.7 *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 $y \in \mathbb{C}^n$, where $\|x\|_2 = \|y\|_2$. 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 Fig. 8.8(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 8.8 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 wish to find a Householder transformation, H, for a vector, x, such that Hx equals a vector with zeroes below the top element. More precisely, partition

$$x = \left(\begin{array}{c} \chi_1 \\ \hline x_2 \end{array} \right),$$

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 $u = \left(\begin{array}{c} 1 \\ \hline u_2 \end{array}\right)$ so that

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

Here $\oplus$ denotes a complex scalar on the complex unit circle[1] Notice that this means that y in the previous discussion equals the vector $\left(\begin{array}{c} \oplus\|x\|_2 \\ \hline 0 \end{array}\right)$ so that the direction of u is given by

$$v = \left(\begin{array}{c} \chi_1 - \oplus\|x\|_2 \\ \hline x_2 \end{array}\right).$$

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

$$u = \frac{v}{\|v\|_2} = \frac{1}{\chi_1 - \oplus\|x\|_2} \left(\begin{array}{c} \chi_1 - \oplus\|x\|_2 \\ \hline x_2 \end{array}\right) = \left(\begin{array}{c} 1 \\ \hline x_2/\nu_1 \end{array}\right).$$

where $\nu_1 = \chi_1 - \oplus\|x\|_2$. (Note that if $\nu_1 = 0$ then u_2 can be set to 0.)

Exercise 8.9 *Verify that*

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

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* $\sqrt{\frac{z}{\bar{z}}} = fracz|z|$.

The choice $\oplus$ is important when computer arithmetic is used and roundoff error is a concern. In practice one chooses $\nu_1 = \chi_1 + \text{sign}(\chi_1)\|x\|_2$ to avoid creating catastrophic cancelation so that $\oplus = -sign(\chi_1)$ ($-\oplus$ points in the opposite direction, in the complex plane, as χ_1). If χ_1 is real valued, this simply means that sign(χ_1) equals the sign of χ_1 in the usual sense. The reason is as follows (again restricting our discussion to real numbers): 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 $\chi_1 - \|x\|_2$. Regardless of whether χ_1 is positive, negative, or complex valued, this can be avoided by using $\nu_1 = \chi_1 + \text{sign}(\chi_1)\|x\|_2$. It is not hard to see that $\text{sign}(\chi_1) = \chi_1/|\chi_1|$.

Let us introduce the notation

$$\left[\left(\begin{array}{c} \rho \\ \hline u_2 \end{array}\right), \tau \right] := \text{Housev}\left(\left(\begin{array}{c} \chi_1 \\ \hline x_2 \end{array}\right) \right)$$

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).

[1]For those who have problems thinking in terms of complex numbers, pretend this entire discussion deals with real numbers and treat $\oplus$ as $\pm$.

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

Figure 8.9: 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.**

8.3.2 Algorithms

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.

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

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

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

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

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

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

and update

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

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 8.10 *Show that*

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

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

```
Algorithm: $[A,t] = \text{URt}(A)$
Partition $A \rightarrow \left(\begin{array}{c|c} \{U\backslash R\}_{TL} & R_{TR} \\ \hline U_{BL} & A_{BR} \end{array}\right)$ and $t \rightarrow \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right)$
    where $\{U\backslash R\}_{TL}$ is $0 \times 0$ and $t_T$ has 0 elements
while $n(A_{BR}) \neq 0$ do
    Repartition
        $\left(\begin{array}{c|c} \{U\backslash R\}_{TL} & R_{TR} \\ \hline U_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} \{U\backslash R\}_{00} & r_{01} & R_{02} \\ \hline u_{10}^T & \alpha_{11} & a_{12}^T \\ \hline U_{20} & a_{21} & A_{22} \end{array}\right)$ 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 $\alpha_{11}$ and $\tau_1$ are scalars

    $\left[\left(\begin{array}{c} \rho_{11} \\ \hline u_{21} \end{array}\right), \tau_1\right] := \text{Housev}\left(\begin{array}{c} \alpha_{11} \\ \hline a_{21} \end{array}\right)$
    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
            • $w_{12}^T := (a_{12}^T + u_{21}^H A_{22})/\tau_1$
            • $\left(\begin{array}{c} a_{12}^T \\ \hline A_{22} \end{array}\right) := \left(\begin{array}{c} a_{12}^T - w_{12}^T \\ \hline A_{22} - u_{21} w_{12}^T \end{array}\right)$

    Continue with
        $\left(\begin{array}{c|c} \{U\backslash R\}_{TL} & R_{TR} \\ \hline U_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} \{U\backslash R\}_{00} & r_{01} & R_{02} \\ \hline u_{10}^T & \rho_{11} & r_{12}^T \\ \hline U_{20} & u_{21} & A_{22} \end{array}\right)$ 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 8.10: Unblocked Householder transformation based QR factorization.

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. 8.10.

We will let

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

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 8.11 *Given $A \in \mathbb{C}^{m\times n}$ the cost of the algorithm in Figure 8.10 is given by*

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

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 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

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

8.3.3 Forming Q

Given $A \in \mathbb{C}^{m\times n}$, let $[A,t] = \mathrm{URt}(A)$ yield the matrix A with the Householder vectors stored below the diagona, R stored on and above the diagonal, and the τ_i stored in vector t. We now discuss how to form the first n columns of $Q = H_0 H_1 \cdots H_{n-1}$. 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 8.12 *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}:

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

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

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

Theorem 8.13 *Given $[A, t] = \mathrm{URt}(A)$ from Figure 8.10, the algorithm in Figure 8.11 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 8.12.

Theorem 8.14 *Given $A \in \mathbb{C}^{m\times n}$ the cost of the algorithm in Figure 8.11 is given by*

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

Proof: Hence the proof for Theorem 8.11 can be easily modified to establish this result.

Exercise 8.15 *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 8.11.

```
Algorithm: $[A] = \textsc{FormQ}(A, t)$
Partition $A \rightarrow \left(\begin{array}{c|c} \{U\backslash R\}_{TL} & R_{TR} \\ \hline U_{BL} & A_{BR} \end{array}\right)$ and $t \rightarrow \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right)$
    where $\{U\backslash R\}_{TL}$ is $n(A) \times n(A)$ and $t_T$ has $n(A)$ elements
while $n(A_{TR}) \neq 0$ do
    Repartition
    $\left(\begin{array}{c|c} \{U\backslash R\}_{TL} & R_{TR} \\ \hline U_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} \{U\backslash R\}_{00} & r_{01} & R_{02} \\ \hline u_{10}^T & \rho_{11} & r_{12}^T \\ \hline U_{20} & u_{21} & A_{22} \end{array}\right)$ 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 $\rho_{11}$ and $\tau_1$ 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
        • $\alpha_{11} := 1 - 1/\tau_1$
        • $a_{12}^T := -(u_{21}^H A_{22})/\tau_1$
        • $A_{22} := A_{22} + u_{21} w_{12}^T$
        • $a_{21} := -u_{21}/\tau_1$

    Continue with
    $\left(\begin{array}{c|c} \{U\backslash R\}_{TL} & R_{TR} \\ \hline U_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} \{U\backslash R\}_{00} & r_{01} & R_{02} \\ \hline u_{10}^T & \alpha_{11} & a_{12}^T \\ \hline U_{20} & a_{21} & A_{22} \end{array}\right)$ 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 8.11: Algorithm for overwriting A with Q from the Householder transformations stored as Householder vectors below the diagonal of A (as produced by $[A, t] = \text{URt}(A)$).

8.4 Solving Linear Least-Squares Problems