4 QR Factorization

4.1 Reduced vs. Full QR

Consider $A \in \mathbb{C}^{m\times n}$ with $m \geq n$. The *reduced* QR factorization of A is of the form

$$A = \hat{Q}\hat{R},$$

where $\hat{Q} \in \mathbb{C}^{m\times n}$ with orthonormal columns and $\hat{R} \in \mathbb{C}^{n\times n}$ an upper triangular matrix such that $\hat{R}(j,j) \neq 0$, $j = 1, \ldots, n$.

As with the SVD $\hat{Q}$ provides an orthonormal basis for range(A), i.e., the columns of A are linear combinations of the columns of $\hat{Q}$. In fact, we have range(A) = range$(\hat{Q})$. This is true since $A\boldsymbol{x} = \hat{Q}\hat{R}\boldsymbol{x} = \hat{Q}\boldsymbol{y}$ for some $\boldsymbol{y}$ so that range$(A) \subseteq$ range$(\hat{Q})$. Moreover, range$(Q) \subseteq$ range$(\hat{A})$ since we can write $A\hat{R}^{-1} = \hat{Q}$ because $\hat{R}$ is upper triangular with nonzero diagonal elements. (Now we have $\hat{Q}\boldsymbol{x} = A\hat{R}^{-1}\boldsymbol{x} = A\boldsymbol{y}$ for some $\boldsymbol{y}$.)

Note that any partial set of columns satisfy the same property, i.e.,

$$\text{span}\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_j\} = \text{span}\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_j\}, \qquad j = 1, \ldots, n.$$

In order to obtain the *full* QR factorization we proceed as with the SVD and extend $\hat{Q}$ to a *unitary* matrix Q. Then $A = QR$ with unitary $Q \in \mathbb{C}^{m\times m}$ and upper triangular $R \in \mathbb{C}^{m\times n}$. Note that (since $m \geq n$) the last $m - n$ rows of R will be zero.

4.2 QR Factorization via Gram-Schmidt

We start by formally writing down the QR factorization $A = QR$ as

$$\begin{aligned}
\boldsymbol{a}_1 &= \boldsymbol{q}_1 r_{11} & \Longrightarrow \quad \boldsymbol{q}_1 &= \frac{\boldsymbol{a}_1}{r_{11}} & (14)\\
\boldsymbol{a}_2 &= \boldsymbol{q}_1 r_{12} + \boldsymbol{q}_2 r_{22} & \Longrightarrow \quad \boldsymbol{q}_2 &= \frac{\boldsymbol{a}_2 - r_{12}\boldsymbol{q}_1}{r_{22}} & (15)\\
&\vdots & &\vdots & (16)\\
\boldsymbol{a}_n &= \boldsymbol{q}_1 r_{1n} + \boldsymbol{q}_2 r_{2n} + \ldots + \boldsymbol{q}_n r_{nn} & \Longrightarrow \quad \boldsymbol{q}_n &= \frac{\boldsymbol{a}_n - \sum_{i=1}^{n} r_{in}\boldsymbol{q}_i}{r_{nn}} & (17)
\end{aligned}$$

Note that in these formulas the columns $\boldsymbol{a}_j$ of A are given and we want to determine the columns $\boldsymbol{q}_j$ of Q and entries r_{ij} of R such that Q is orthonormal, i.e.,

$$\boldsymbol{q}_i^* \boldsymbol{q}_j = \delta_{ij}, \tag{18}$$

R is upper triangular and $A = QR$. The latter two conditions are already reflected in the formulas above.

Using (14) in the orthogonality condition (18) we get

$$\boldsymbol{q}_1^* \boldsymbol{q}_1 = \frac{\boldsymbol{a}_1^* \boldsymbol{a}_1}{r_{11}^2} = 1$$

so that

$$r_{11} = \sqrt{\boldsymbol{a}_1^* \boldsymbol{a}_1} = \|\boldsymbol{a}_1\|_2.$$

Note that we arbitrarily chose the positive square root here (so that the factorization becomes unique).

Next, the orthogonality condition (18) gives us

$$\begin{aligned} \boldsymbol{q}_1^* \boldsymbol{q}_2 &= 0 \\ \boldsymbol{q}_2^* \boldsymbol{q}_2 &= 1. \end{aligned}$$

Now we apply (15) to the first of these two conditions. Then

$$\boldsymbol{q}_1^* \boldsymbol{q}_2 = \frac{\boldsymbol{q}_1^* \boldsymbol{a}_2 - r_{12} \boldsymbol{q}_1^* \boldsymbol{q}_1}{r_{22}} = 0.$$

Since we ensured $\boldsymbol{q}_1^* \boldsymbol{q}_1 = 1$ in the previous step, the numerator yields $r_{12} = \boldsymbol{q}_1^* \boldsymbol{a}_2$ so that

$$\boldsymbol{q}_2 = \frac{\boldsymbol{a}_2 - (\boldsymbol{q}_1^* \boldsymbol{a}_2) \boldsymbol{q}_1}{r_{22}}.$$

To find r_{22} we normalize, i.e., demand that $\boldsymbol{q}_2^* \boldsymbol{q}_2 = 1$ or equivalently $\|\boldsymbol{q}_2\|_2 = 1$. This immediately gives

$$r_{22} = \|\boldsymbol{a}_2 - (\boldsymbol{q}_1^* \boldsymbol{a}_2) \boldsymbol{q}_1\|_2.$$

To fully understand how the algorithm proceeds we add one more step (for $n = 3$). Now we have three orthogonality conditions:

$$\begin{aligned} \boldsymbol{q}_1^* \boldsymbol{q}_3 &= 0 \\ \boldsymbol{q}_2^* \boldsymbol{q}_3 &= 0 \\ \boldsymbol{q}_3^* \boldsymbol{q}_3 &= 1. \end{aligned}$$

The first of these conditions together with (17) for $n = 3$ yields

$$\boldsymbol{q}_1^* \boldsymbol{q}_3 = \frac{\boldsymbol{q}_1^* \boldsymbol{a}_3 - r_{13} \boldsymbol{q}_1^* \boldsymbol{q}_1 - r_{23} \boldsymbol{q}_1^* \boldsymbol{q}_2}{r_{33}} = 0$$

so that $r_{13} = \boldsymbol{q}_1^* \boldsymbol{a}_3$ due to the orthonormality of columns $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$.

Similarly, the second orthogonality condition together with (17) for $n = 3$ yields

$$\boldsymbol{q}_2^* \boldsymbol{q}_3 = \frac{\boldsymbol{q}_2^* \boldsymbol{a}_3 - r_{13} \boldsymbol{q}_2^* \boldsymbol{q}_1 - r_{23} \boldsymbol{q}_2^* \boldsymbol{q}_2}{r_{33}} = 0$$

so that $r_{23} = \boldsymbol{q}_2^* \boldsymbol{a}_3$.

Together this gives us

$$\boldsymbol{q}_3 = \frac{\boldsymbol{a}_3 - (\boldsymbol{q}_1^* \boldsymbol{a}_3) \boldsymbol{q}_1 - (\boldsymbol{q}_2^* \boldsymbol{a}_3) \boldsymbol{q}_2}{r_{33}}$$

and the last unknown, r_{33}, is determined by normalization, i.e.,

$$r_{33} = \|\boldsymbol{a}_3 - (\boldsymbol{q}_1^* \boldsymbol{a}_3) \boldsymbol{q}_1 - (\boldsymbol{q}_2^* \boldsymbol{a}_3) \boldsymbol{q}_2\|_2.$$

In general we can formulate the following algorithm:

$$
\begin{aligned}
r_{ij} &= \boldsymbol{q}_i^* \boldsymbol{a}_j \quad (i \neq j) \\
\boldsymbol{v}_j &= \boldsymbol{a}_j - \sum_{i=1}^{j-1} r_{ij} \boldsymbol{q}_i \\
r_{jj} &= \|\boldsymbol{v}_j\|_2 \\
\boldsymbol{q}_j &= \frac{\boldsymbol{v}_j}{r_{jj}}
\end{aligned}
$$

We can compute the reduced QR factorization with the following (somewhat more practical and almost Matlab implementation of the) *classical Gram-Schmidt algorithm.*

Algorithm (Classical Gram-Schmidt)

```
for $j = 1 : n$
    $\boldsymbol{v}_j = \boldsymbol{a}_j$
    for $i = 1 : (j-1)$
        $r_{ij} = \boldsymbol{q}_i^* \boldsymbol{a}_j$
        $\boldsymbol{v}_j = \boldsymbol{v}_j - r_{ij} \boldsymbol{q}_i$
    end
    $r_{jj} = \|\boldsymbol{v}_j\|_2$
    $\boldsymbol{q}_j = \boldsymbol{v}_j / r_{jj}$
end
```

Remark The classical Gram-Schmidt algorithm is not ideal for numerical calculations since it is known to be unstable. Note that, by construction, the Gram-Schmidt algorithm yields an existence proof for the QR factorization.

Theorem 4.1 *Let $A \in \mathbb{C}^{m \times n}$ with $m \geq n$. Then A has a QR factorization. Moreover, if A is of full rank (n), then the reduced factorization $A = \hat{Q}\hat{R}$ with $r_{jj} > 0$ is unique.*

Example We compute the QR factorization for the matrix

$$
A = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}.
$$

First $\boldsymbol{v}_1 = \boldsymbol{a}_1 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$ and $r_{11} = \|\boldsymbol{v}_1\| = \sqrt{2}$. This gives us

$$
\boldsymbol{q}_1 = \frac{\boldsymbol{v}_1}{\|\boldsymbol{v}_1\|} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}.
$$

Next,

$$\begin{aligned}\boldsymbol{v}_2 &= \boldsymbol{a}_2 - \underbrace{(\boldsymbol{q}_1^*\boldsymbol{a}_2)}_{=r_{12}}\boldsymbol{q}_1 \\ &= \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} - \frac{\sqrt{2}}{\sqrt{2}}\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}.\end{aligned}$$

This calculation required that $r_{12} = \frac{2}{\sqrt{2}} = \sqrt{2}$. Moreover, $r_{22} = \|\boldsymbol{v}_2\| = \sqrt{3}$ and

$$\boldsymbol{q}_2 = \frac{\boldsymbol{v}_2}{\|\boldsymbol{v}_2\|} = \frac{1}{\sqrt{3}}\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}.$$

In the third iteration we have

$$\boldsymbol{v}_3 = \boldsymbol{a}_3 - \underbrace{(\boldsymbol{q}_1^*\boldsymbol{a}_3)}_{=r_{13}}\boldsymbol{q}_1 - \underbrace{(\boldsymbol{q}_2^*\boldsymbol{a}_3)}_{=r_{23}}\boldsymbol{q}_2$$

from which we first compute $r_{13} = \frac{1}{\sqrt{2}}$ and $r_{23} = 0$. This gives us

$$\boldsymbol{v}_3 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} - \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} - 0 = \frac{1}{2}\begin{bmatrix} -1 \\ 2 \\ 1 \end{bmatrix}.$$

Finally, $r_{33} = \|\boldsymbol{v}_3\| = \frac{\sqrt{6}}{2}$ and

$$\boldsymbol{q}_3 = \frac{\boldsymbol{v}_3}{\|\boldsymbol{v}_3\|} = \frac{1}{\sqrt{6}}\begin{bmatrix} -1 \\ 2 \\ 1 \end{bmatrix}.$$

Collecting all of the information we end up with

$$Q = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} & \frac{-1}{\sqrt{6}} \\ 0 & \frac{1}{\sqrt{3}} & \frac{2}{\sqrt{6}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{3}} & \frac{1}{\sqrt{6}} \end{bmatrix} \quad \text{and} \quad R = \begin{bmatrix} \sqrt{2} & \sqrt{2} & \frac{1}{\sqrt{2}} \\ 0 & \sqrt{3} & 0 \\ 0 & 0 & \frac{\sqrt{6}}{2} \end{bmatrix}.$$

4.3 An Application of the QR Factorization

Consider solution of the linear system $A\boldsymbol{x} = \boldsymbol{b}$ with $A \in \mathbb{C}^{m\times m}$ nonsingular. Since

$$A\boldsymbol{x} = \boldsymbol{b} \iff QR\boldsymbol{x} = \boldsymbol{b} \iff R\boldsymbol{x} = Q^*\boldsymbol{b},$$

where the last equation holds since Q is unitary, we can proceed as follows:

1. Compute $A = QR$ (which is the same as $A = \hat{Q}\hat{R}$ in this case).
2. Compute $\boldsymbol{y} = Q^*\boldsymbol{b}$.

3. Solve the upper triangular $R\boldsymbol{x} = \boldsymbol{y}$

We will have more applications for the QR factorization later in the context of least squares problems.

Remark The QR factorization (if implemented properly) yields a very stable method for solving $A\boldsymbol{x} = \boldsymbol{b}$. However, it is about twice as costly as Gauss elimination (or $A = LU$). In fact, the QR factorization can also be applied to rectangular systems and it is the basis of Matlab's backslash matrix division operator. We will discuss Matlab examples in a later section.

4.4 Modified Gram-Schmidt

The classical Gram-Schmidt algorithm is based on projections of the form

$$\begin{aligned}\boldsymbol{v}_j &= \boldsymbol{a}_j - \sum_{i=1}^{j-1} r_{ij}\boldsymbol{q}_i \\ &= \boldsymbol{a}_j - \sum_{i=1}^{j-1} (\boldsymbol{q}_i^*\boldsymbol{a}_j)\boldsymbol{q}_i.\end{aligned}$$

Note that this means we are performing a sequence of *vector projections*. The starting point for the modified Gram-Schmidt algorithm is to rewrite one step of the classical Gram-Schmidt algorithm as a single *matrix projection*, i.e.,

$$\begin{aligned}\boldsymbol{v}_j &= \boldsymbol{a}_j - \sum_{i=1}^{j-1} (\boldsymbol{q}_i^*\boldsymbol{a}_j)\boldsymbol{q}_i \\ &= \boldsymbol{a}_j - \sum_{i=1}^{j-1} (\boldsymbol{q}_i\boldsymbol{q}_i^*)\boldsymbol{a}_j \\ &= \boldsymbol{a}_j - \hat{Q}_{j-1}\hat{Q}_{j-1}^*\boldsymbol{a}_j \\ &= \underbrace{\left(I - \hat{Q}_{j-1}\hat{Q}_{j-1}^*\right)}_{=P_j}\boldsymbol{a}_j,\end{aligned}$$

where $\hat{Q}_{j-1} = [\boldsymbol{q}_1\boldsymbol{q}_2\ldots\boldsymbol{q}_{j-1}]$ is the matrix formed by the column vectors $\boldsymbol{q}_i$, $i = 1,\ldots,j-1$.

In order to obtain the modified Gram-Schmidt algorithm we require the following observation that the single projection P_j can also be viewed as a series of complementary projections onto the individual columns $\boldsymbol{q}_i$, i.e.,

Lemma 4.2 *If* $P_j = I - \hat{Q}_{j-1}\hat{Q}_{j-1}^*$ *with* $\hat{Q}_{j-1} = [\boldsymbol{q}_1\boldsymbol{q}_2\ldots\boldsymbol{q}_{j-1}]$ *a matrix with orthonormal columns, then*

$$P_j = \prod_{i=1}^{j-1} P_{\perp\boldsymbol{q}_i}.$$

Proof First we remember that

$$P_j = I - \hat{Q}_{j-1}\hat{Q}_{j-1}^* = I - \sum_{i=1}^{j-1} \boldsymbol{q}_i \boldsymbol{q}_i^*$$

and that the complementary projector is defined as

$$P_{\perp \boldsymbol{q}_i} = I - \boldsymbol{q}_i \boldsymbol{q}_i^*.$$

Therefore, we need to show that

$$I - \sum_{i=1}^{j-1} \boldsymbol{q}_i \boldsymbol{q}_i^* = \prod_{i=1}^{j-1} \left(I - \boldsymbol{q}_i \boldsymbol{q}_i^*\right).$$

This is done by induction. For $j = 1$ the sum and the product are empty and the statement holds by the convention that an empty sum is zero and an empty product is the identity, i.e., $P_1 = I$.

Now we step from $j-1$ to j. First

$$\begin{aligned}\prod_{i=1}^{j} \left(I - \boldsymbol{q}_i \boldsymbol{q}_i^*\right) &= \prod_{i=1}^{j-1} \left(I - \boldsymbol{q}_i \boldsymbol{q}_i^*\right)\left(I - \boldsymbol{q}_j \boldsymbol{q}_j^*\right) \\ &= \left(I - \sum_{i=1}^{j-1} \boldsymbol{q}_i \boldsymbol{q}_i^*\right)\left(I - \boldsymbol{q}_j \boldsymbol{q}_j^*\right)\end{aligned}$$

by the induction hypothesis. Expanding the right-hand side yields

$$I - \sum_{i=1}^{j-1} \boldsymbol{q}_i \boldsymbol{q}_i^* - \boldsymbol{q}_j \boldsymbol{q}_j^* + \sum_{i=1}^{j-1} \boldsymbol{q}_i \underbrace{\boldsymbol{q}_i^* \boldsymbol{q}_j}_{=0} \boldsymbol{q}_j^*$$

so that the claim is proved. ■

Summarizing the discussion thus far, a single step in the Gram-Schmidt algorithm can be written as

$$\boldsymbol{v}_j = P_{\perp \boldsymbol{q}_{j-1}} P_{\perp \boldsymbol{q}_{j-2}} \dots P_{\perp \boldsymbol{q}_1} \boldsymbol{a}_j,$$

or – more algorithmically:

```
$\boldsymbol{v}_j = \boldsymbol{a}_j$
for $i = 1 : (j-1)$
    $\boldsymbol{v}_j = \boldsymbol{v}_j - \boldsymbol{q}_i \boldsymbol{q}_i^* \boldsymbol{v}_j$
end
```

For the final *modified Gram-Schmidt algorithm* the projections are arranged differently, i.e., $P_{\perp \boldsymbol{q}_i}$ is applied to all $\boldsymbol{v}_j$ with $j > i$. This leads to

Algorithm (Modified Gram-Schmidt)

```
for $i = 1 : n$
        $\boldsymbol{v}_i = \boldsymbol{a}_i$
end
for $i = 1 : n$
        $r_{ii} = \|\boldsymbol{v}_i\|_2$
        $\boldsymbol{q}_i = \frac{\boldsymbol{v}_i}{r_{ii}}$
        for $j = (i+1) : n$
            $r_{ij} = \boldsymbol{q}_i^* \boldsymbol{v}_j$
            $\boldsymbol{v}_j = \boldsymbol{v}_j - r_{ij}\boldsymbol{q}_i$
        end
end
```

We can compare the *operations count*, i.e., the number of basic arithmetic operations ('+','-','*','/'), of the two algorithms. We give only a rough estimate (exact counts will be part of the homework). Assuming vectors of length m, for the classical Gram-Schmidt roughly $4m$ operations are performed inside the innermost loop (actually m multiplications and $m-1$ additions for the inner product, and m multiplications and m subtractions for the formula in the second line). Thus, the operations count is roughly

$$\sum_{j=1}^{n}\sum_{i=1}^{j-1} 4m = \sum_{j=1}^{n}(j-1)4m \approx 4m\sum_{j=1}^{n} j = 4m\frac{n(n+1)}{2} \approx 2mn^2.$$

The innermost loop of the modified Gram-Schmidt algorithm consists formally of exactly the same operations, i.e., requires also roughly $4m$ operations. Thus its operation count is

$$\sum_{i=1}^{n}\sum_{j=i+1}^{n} 4m = \sum_{i=1}^{n}(n-i)4m = 4m\left(n^2 - \sum_{i=1}^{n} i\right) = 4m\left(n^2 - \frac{n(n+1)}{2}\right) \approx 2mn^2.$$

Thus, the operations count for the two algorithms is the same. In fact, mathematically, the two algorithms can be shown to be identical. However, we will learn later that the modified Gram-Schmidt algorithm is to be preferred due to its better numerical stability (see Section 4.6).

4.5 Gram-Schmidt as Triangular Orthogonalization

One can view the modified Gram-Schmidt algorithm (applied to the entire matrix A) as

$$AR_1R_2\ldots R_n = \hat{Q}, \tag{19}$$

where $R_1, \ldots, R_n$ are upper triangular matrices. For example,

$$\begin{aligned} R_1 &= \begin{bmatrix} 1/r_{11} & -r_{12}/r_{11} & -r_{13}/r_{11} & \cdots & -r_{1m}/r_{11} \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & \cdots & & 0 & 1 \end{bmatrix}, \\ R_2 &= \begin{bmatrix} 1 & 0 & & \cdots & 0 \\ 0 & 1/r_{22} & -r_{23}/r_{22} & \cdots & -r_{2m}/r_{22} \\ 0 & 0 & 1 & & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & \cdots & & 0 & 1 \end{bmatrix} \end{aligned}$$

and so on.

Thus we are applying triangular transformation matrices to A to obtain a matrix $\hat{Q}$ with orthonormal columns. We refer to this approach as *triangular orthogonalization.*

Since the inverse of an upper triangular matrix is again an upper triangular matrix, and the product of two upper triangular matrices is also upper triangular, we can think of the product $R_1 R_2 \ldots R_n$ in (19) in terms of a matrix $\hat{R}^{-1}$. Thus, the (modified) Gram-Schmidt algorithm yields a reduced QR factorization

$$A = \hat{Q}\hat{R}$$

of A.

4.6 Stability of CGS vs. MGS in Matlab

The following discussion is taken from [Trefethen/Bau] and illustrated by the Matlab code `GramSchmidt.m` (whose supporting routines `clgs.m` and `mgs.m` are part of a computer assignment).

We create a random matrix $A \in \mathbb{R}^{80\times 80}$ by selecting singular values $\frac{1}{2}, \frac{1}{4}, \ldots, \frac{1}{2^{80}}$ and generating $A = U\Sigma V^*$ with the help of (orthonormal) matrices U and V whose entries are normally distributed random numbers (using the Matlab command `randn`). Then we compute the QR factorization $A = QR$ using both the classical and modified Gram-Schmidt algorithms. The program then plots the diagonal elements of R together with the singular values.

First we note that

$$A = \sum_{i=1}^{80} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$$

so that

$$\boldsymbol{a}_j = A(:, j) = \sum_{i=1}^{80} \sigma_i \boldsymbol{u}_i v_{ji}.$$

Next, V is a normally distributed random unitary matrix, and therefore the entries in one of its columns satisfy

$$|v_{ji}| \approx \frac{1}{\sqrt{80}} \approx 0.1.$$

Now from the (classical) Gram-Schmidt algorithm we know that

$$r_{11} = \|\boldsymbol{a}_1\|_2 = \|\sum_{i=1}^{80} \sigma_1 v_{1i} \boldsymbol{u}_i\|_2.$$

Since the singular values were chosen to decrease exponentially only the first one really matters, i.e.,

$$r_{11} \approx \|\sigma_1 v_{11} \boldsymbol{u}_1\|_2 = \sigma_1 v_{11} \approx \frac{1}{2}\frac{1}{\sqrt{80}}$$

(since $\|\boldsymbol{u}_1\|_2 = 1$).

Similar arguments result in the general relationship

$$r_{jj} \approx \frac{1}{\sqrt{80}}\sigma_j$$

(the latter of which we know). The plot produced by `GramSchmidt.m` shows how accurately the diagonal elements of R are computed. We can observe that the classical Gram-Schmidt algorithm is stable up to $\sigma_j \approx \sqrt{\texttt{eps}}$ (where `eps` is the machine epsilon), whereas the modified Gram-Schmidt method is stable all the way up to $\sigma_j \approx \texttt{eps}$.

Remark In spite of the superior stability of the modified Gram-Schmidt algorithm it still may not produce "good" orthogonality. *Househoulder reflections* – studied in the next chapter – work better (see an example in [Trefethen/Bau]).

4.7 Householder Triangularization

Recall that we interpreted the Gram-Schmidt algorithm as triangular orthogonalization

$$AR_1R_2\ldots R_n = \hat{Q}$$

leading to the reduced QR factorization of an $m \times n$ matrix A. Now we will consider an alternative approach to computing the (full) QR factorization corresponding to *orthogonal triangularization*:

$$Q_nQ_{n-1}\ldots Q_2Q_1A = R,$$

where the matrices Q_j are unitary.

The idea here is to design matrices $Q_1, \ldots, Q_n$ such that A is successively transformed to upper triangular form, i.e.,

$$A = \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} \quad \longrightarrow \quad Q_1A = \begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & x & x \\ 0 & x & x \end{bmatrix}$$

$$\longrightarrow \quad Q_2Q_1A = \begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \\ 0 & 0 & x \end{bmatrix} \quad \longrightarrow \quad Q_3Q_2Q_1A = \begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \\ 0 & 0 & 0 \end{bmatrix},$$

where x stands for a generally nonzero entry. From this we note that Q_k needs to operate on rows $k : m$ and not change the first $k-1$ rows and columns. Therefore it will be of the form

$$Q_k = \left[\begin{array}{cc} I_{k-1} & O \\ O & F \end{array} \right],$$

where I_{k-1} is a $(k-1) \times (k-1)$ identity matrix and F has the effect that

$$F\boldsymbol{x} = \|\boldsymbol{x}\|\boldsymbol{e}_1$$

in order to introduce zeros in the lower part of column k. We will call F a *Householder reflector.*

Graphically, we can use either a rotation (*Givens rotation*) or a reflection about the bisector of $\boldsymbol{x}$ and $\boldsymbol{e}_1$ to transform $\boldsymbol{x}$ to $\|\boldsymbol{x}\|\boldsymbol{e}_1$.

Recall from an earlier homework assignment that given a projector P, then $(I-2P)$ is also a projector. In fact, $(I-2P)$ is a reflector. Therefore, if we choose $\boldsymbol{v} = \|\boldsymbol{x}\|\boldsymbol{e}_1 - \boldsymbol{x}$ and define $P = \frac{\boldsymbol{v}\boldsymbol{v}^*}{\boldsymbol{v}^*\boldsymbol{v}}$, then

$$F = I - 2P = I - 2\frac{\boldsymbol{v}\boldsymbol{v}^*}{\boldsymbol{v}^*\boldsymbol{v}}$$

is our desired Householder reflector. Since it is easy to see that F is Hermitian, so is Q_k. Note that $F\boldsymbol{x}$ can be computed as

$$F\boldsymbol{x} = \left(I - 2\frac{\boldsymbol{v}\boldsymbol{v}^*}{\boldsymbol{v}^*\boldsymbol{v}} \right) \boldsymbol{x} = \boldsymbol{x} - 2 \underbrace{\frac{\boldsymbol{v}\boldsymbol{v}^*}{\boldsymbol{v}^*\boldsymbol{v}}}_{\text{matrix}} \boldsymbol{x} = \boldsymbol{x} - 2\boldsymbol{v} \underbrace{\frac{\boldsymbol{v}^*\boldsymbol{x}}{\boldsymbol{v}^*\boldsymbol{v}}}_{\text{scalar}} .$$

In fact, we have two choices for the reflection $F\boldsymbol{x}$: $\boldsymbol{v}_+ = -\boldsymbol{x} + \operatorname{sign}(\boldsymbol{x}(1))\|\boldsymbol{x}\|\boldsymbol{e}_1$ and $\boldsymbol{v}_- = -\boldsymbol{x} - \operatorname{sign}(\boldsymbol{x}(1))\|\boldsymbol{x}\|\boldsymbol{e}_1$. Here $\boldsymbol{x}(1)$ denotes the first component of the vector $\boldsymbol{x}$. These choices are illustrated in Figure 4. A numerically more stable algorithm

Figure 4: Graphical interpretation of Householder reflections.

(that will avoid cancellation of significant digits) will be guaranteed by choosing that reflection which moves $\boldsymbol{x}$ further. Therefore we pick

$$\boldsymbol{v} = \boldsymbol{x} + \operatorname{sign}(\boldsymbol{x}(1))\|\boldsymbol{x}\|\boldsymbol{e}_1,$$

which is the same (except for orientation) as $\boldsymbol{v}_-$.

The resulting algorithm is

Algorithm (Householder QR)

```
for $k = 1 : n$        (sum over columns)
    $\boldsymbol{x} = A(k : m, k)$
    $\boldsymbol{v}_k = \boldsymbol{x} + \mathrm{sign}(\boldsymbol{x}(1))\|\boldsymbol{x}\|_2\boldsymbol{e}_1$
    $\boldsymbol{v}_k = \boldsymbol{v}_k/\|\boldsymbol{v}_k\|_2$
    $A(k : m, k : n) = A(k : m, k : n) - 2\boldsymbol{v}_k\left(\boldsymbol{v}_k^* A(k : m, k : n)\right)$
end
```

Note that the statement in the last line of the algorithm performs the reflection simultaneously for all remaining columns of the matrix A. On completion of this algorithm the matrix A contains the matrix R of the QR factorization, and the vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ are the reflection vectors. They will be used to calculate matrix-vector products of the form $Q\boldsymbol{x}$ and $Q^*\boldsymbol{b}$ later on. The matrix Q itself is not output. It can be constructed by computing special matrix-vector products $Q\boldsymbol{x}$ with $\boldsymbol{x} = \boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$.

Example We apply Householder reflection to $\boldsymbol{x} = [2, 1, 2]^T$.

First we compute

$$\begin{aligned}\boldsymbol{v} &= \boldsymbol{x} + \mathrm{sign}(\boldsymbol{x}(1))\|\boldsymbol{x}\|_2\boldsymbol{e}_1 \\ &= \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} + 3\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 5 \\ 1 \\ 2 \end{bmatrix}.\end{aligned}$$

Next we form $F\boldsymbol{x} = \boldsymbol{x} - 2\boldsymbol{v}\frac{\boldsymbol{v}^*\boldsymbol{x}}{\boldsymbol{v}^*\boldsymbol{v}}$. To this end we note that $\boldsymbol{v}^*\boldsymbol{x} = 15$ and $\boldsymbol{v}^*\boldsymbol{v} = 30$. Thus

$$F\boldsymbol{x} = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} 23 \begin{bmatrix} 5 \\ 1 \\ 2 \end{bmatrix} \frac{15}{30} = \begin{bmatrix} -3 \\ 0 \\ 0 \end{bmatrix}.$$

This vector contains the desired zero.

For many applications only products of the form $Q^*\boldsymbol{b}$ or $Q\boldsymbol{x}$ are needed. For example, if we want to solve the linear system $A\boldsymbol{x} = \boldsymbol{b}$ then we can do this with the QR factorization by first computing $\boldsymbol{y} = Q^*\boldsymbol{b}$ and then solving $R\boldsymbol{x} = \boldsymbol{y}$. Therefore, we list the respective algorithms for these two types of matrix vector products.

For the first algorithm we need to remember that

$$\underbrace{Q_n \ldots Q_2 Q_1}_{=Q^*} A = R,$$

so that we can apply exactly the same steps that were applied to the matrix A in the Householder QR algorithm:

Algorithm (Compute $Q^*\boldsymbol{b}$)

```
for $k = 1 : n$
```

```
        $\boldsymbol{b}(k:m) = \boldsymbol{b}(k:m) - 2\boldsymbol{v}_k\left(\boldsymbol{v}_k^* \boldsymbol{b}(k:m)\right)$
    end
```

For the second algorithm we use $Q = Q_1 Q_2 \dots Q_n$ (since $Q_i^* = Q_i$), so that the following algorithm simply performs the reflection operations in reverse order:

Algorithm (Compute $Q\boldsymbol{x}$)

```
    for $k = n : -1 : 1$
        $\boldsymbol{x}(k:m) = \boldsymbol{x}(k:m) - 2\boldsymbol{v}_k\left(\boldsymbol{v}_k^* \boldsymbol{x}(k:m)\right)$
    end
```

The operations counts for the three algorithms listed above are

Householder QR: $\mathcal{O}\left(2mn^2 - \frac{2}{3}n^3\right)$
Q^*b, $Q\boldsymbol{x}$: $\mathcal{O}(mn)$