are

$$\begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}^T x = -3, \qquad \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}^T x = \frac{3}{\sqrt{2}}, \qquad \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}^T x = \frac{-1}{\sqrt{2}}.$$

It can be verified that the expansion of x in this basis is

$$x = (-3) \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} + \frac{3}{\sqrt{2}} \left(\frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \right) + \frac{-1}{\sqrt{2}} \left(\frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} \right).$$

5.4 Gram–Schmidt algorithm

In this section we describe an algorithm that can be used to determine if a list of n-vectors $a_1, \ldots, a_k$ is linearly independent. In later chapters we will see that it has many other uses as well. The algorithm is named after the mathematicians Jørgen Pedersen Gram and Erhard Schmidt, although it was already known before their work.

If the vectors are linearly independent, the Gram–Schmidt algorithm produces an orthonormal collection of vectors $q_1, \ldots, q_k$ with the following properties: For each $i = 1, \ldots, k$, a_i is a linear combination of $q_1, \ldots, q_i$, and q_i is a linear combination of $a_1, \ldots, a_i$. If the vectors $a_1, \ldots, a_{j-1}$ are linearly independent, but $a_1, \ldots, a_j$ are linearly dependent, the algorithm detects this and terminates. In other words, the Gram–Schmidt algorithm finds the first vector a_j that is a linear combination of previous vectors $a_1, \ldots, a_{j-1}$.

Algorithm 5.1 GRAM–SCHMIDT ALGORITHM

given n-vectors $a_1, \ldots, a_k$

for $i = 1, \ldots, k$,

1. *Orthogonalization.* $\tilde{q}_i = a_i - (q_1^T a_i)q_1 - \cdots - (q_{i-1}^T a_i)q_{i-1}$
2. *Test for linear dependence.* if $\tilde{q}_i = 0$, quit.
3. *Normalization.* $q_i = \tilde{q}_i / \|\tilde{q}_i\|$

The orthogonalization step, with $i = 1$, reduces to $\tilde{q}_1 = a_1$. If the algorithm does not quit (in step 2), *i.e.*, $\tilde{q}_1, \ldots, \tilde{q}_k$ are all nonzero, we can conclude that the original collection of vectors is linearly independent; if the algorithm does quit early, say, with $\tilde{q}_j = 0$, we can conclude that the original collection of vectors is linearly dependent (and indeed, that a_j is a linear combination of $a_1, \ldots, a_{j-1}$).

Figure 5.3 illustrates the Gram–Schmidt algorithm for two 2-vectors. The top row shows the original vectors; the middle and bottom rows show the first and second iterations of the loop in the Gram–Schmidt algorithm, with the left-hand side showing the orthogonalization step, and the right-hand side showing the normalization step.

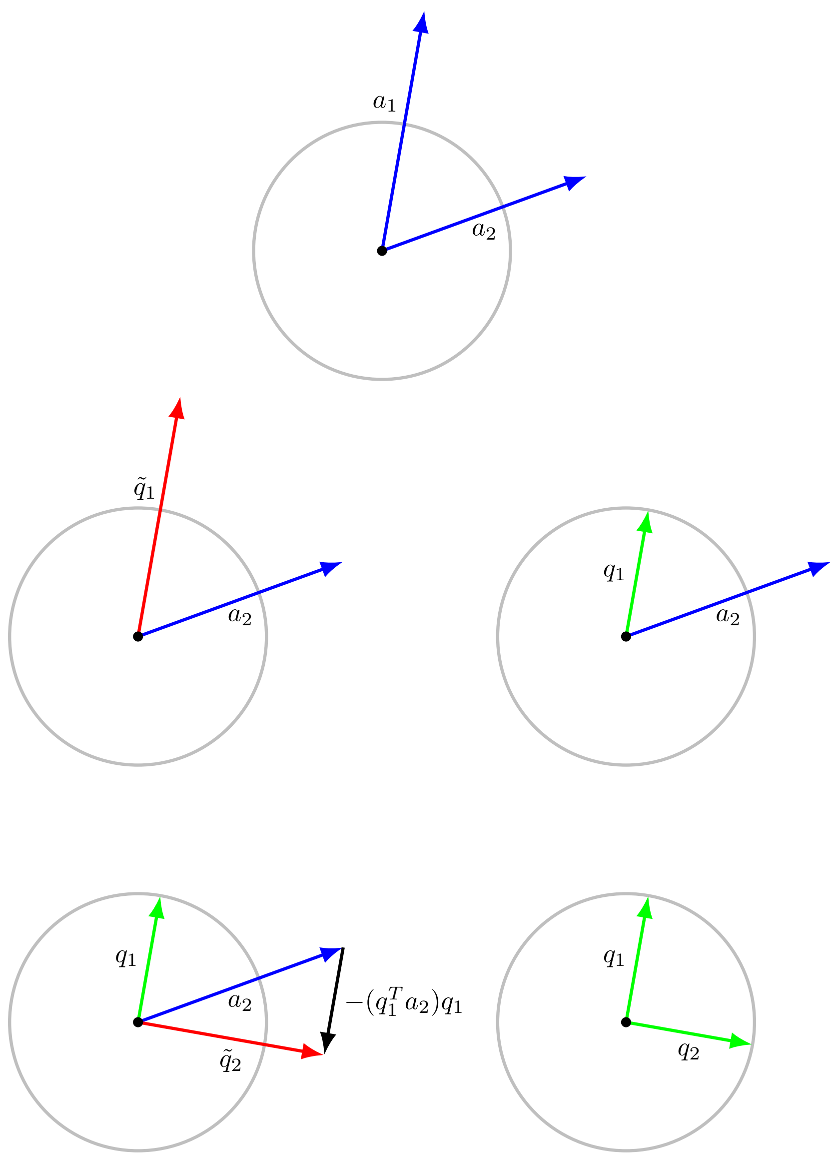

Figure 5.3 Gram–Schmidt algorithm applied to two 2-vectors a_1, a_2. *Top.* The original vectors a_1 and a_2. The gray circle shows the points with norm one. *Middle left.* The orthogonalization step in the first iteration yields $\tilde{q}_1 = a_1$. *Middle right.* The normalization step in the first iteration scales $\tilde{q}_1$ to have norm one, which yields q_1. *Bottom left.* The orthogonalization step in the second iteration subtracts a multiple of q_1 to yield the vector $\tilde{q}_2$, which is orthogonal to q_1. *Bottom right.* The normalization step in the second iteration scales $\tilde{q}_2$ to have norm one, which yields q_2.

Analysis of Gram–Schmidt algorithm. Let us show that the following hold, for $i = 1, \ldots, k$, assuming $a_1, \ldots, a_k$ are linearly independent.

1. $\tilde{q}_i \neq 0$, so the linear dependence test in step 2 is not satisfied, and we do not have a divide-by-zero error in step 3.

2. $q_1, \ldots, q_i$ are orthonormal.

3. a_i is a linear combination of $q_1, \ldots, q_i$.

4. q_i is a linear combination of $a_1, \ldots, a_i$.

We show this by induction. For $i = 1$, we have $\tilde{q}_1 = a_1$. Since $a_1, \ldots, a_k$ are linearly independent, we must have $a_1 \neq 0$, and therefore $\tilde{q}_1 \neq 0$, so assertion 1 holds. The single vector q_1 (considered as a list with one element) is evidently orthonormal, since $\|q_1\| = 1$, so assertion 2 holds. We have $a_1 = \|\tilde{q}_1\| q_1$, and $q_1 = (1/\|\tilde{q}_1\|) a_1$, so assertions 3 and 4 hold.

Suppose our assertion holds for some $i-1$, with $i < k$; we will show it holds for i. If $\tilde{q}_i = 0$, then a_i is a linear combination of $q_1, \ldots, q_{i-1}$ (from the first step in the algorithm); but each of these is (by the induction hypothesis) a linear combination of $a_1, \ldots, a_{i-1}$, so it follows that a_i is a linear combination of $a_1, \ldots, a_{i-1}$, which contradicts our assumption that $a_1, \ldots, a_k$ are linearly independent. So assertion 1 holds for i.

Step 3 of the algorithm ensures that $q_1, \ldots, q_i$ are normalized; to show they are orthogonal we will show that $q_i \perp q_j$ for $j = 1, \ldots, i-1$. (Our induction hypothesis tells us that $q_r \perp q_s$ for $r, s < i$.) For any $j = 1, \ldots, i-1$, we have (using step 1)

$$\begin{aligned} q_j^T \tilde{q}_i &= q_j^T a_i - (q_1^T a_i)(q_j^T q_1) - \cdots - (q_{i-1}^T a_i)(q_j^T q_{i-1}) \\ &= q_j^T a_i - q_j^T a_i = 0, \end{aligned}$$

using $q_j^T q_k = 0$ for $j \neq k$ and $q_j^T q_j = 1$. (This explains why step 1 is called the orthogonalization step: We subtract from a_i a linear combination of $q_1, \ldots, q_{i-1}$ that ensures $q_i \perp \tilde{q}_j$ for $j < i$.) Since $q_i = (1/\|\tilde{q}_i\|)\tilde{q}_i$, we have $q_i^T q_j = 0$ for $j = 1, \ldots, i-1$. So assertion 2 holds for i.

It is immediate that a_i is a linear combination of $q_1, \ldots, q_i$:

$$\begin{aligned} a_i &= \tilde{q}_i + (q_1^T a_i) q_1 + \cdots + (q_{i-1}^T a_i) q_{i-1} \\ &= (q_1^T a_i) q_1 + \cdots + (q_{i-1}^T a_i) q_{i-1} + \|\tilde{q}_i\| q_i. \end{aligned}$$

From step 1 of the algorithm, we see that $\tilde{q}_i$ is a linear combination of the vectors $a_1, q_1, \ldots, q_{i-1}$. By the induction hypothesis, each of $q_1, \ldots, q_{i-1}$ is a linear combination of $a_1, \ldots, a_{i-1}$, so $\tilde{q}_i$ (and therefore also q_i) is a linear combination of $a_1, \ldots, a_i$. Thus assertions 3 and 4 hold.

Gram–Schmidt completion implies linear independence. From the properties 1–4 above, we can argue that the original collection of vectors $a_1, \ldots, a_k$ is linearly independent. To see this, suppose that

$$\beta_1 a_1 + \cdots + \beta_k a_k = 0 \tag{5.6}$$

holds for some $\beta_1, \ldots, \beta_k$. We will show that $\beta_1 = \cdots = \beta_k = 0$.

We first note that any linear combination of $q_1, \ldots, q_{k-1}$ is orthogonal to any multiple of q_k, since $q_1^T q_k = \cdots = q_{k-1}^T q_k = 0$ (by definition). But each of $a_1, \ldots, a_{k-1}$ is a linear combination of $q_1, \ldots, q_{k-1}$, so we have $q_k^T a_1 = \cdots = q_k^T a_{k-1} = 0$. Taking the inner product of q_k with the left- and right-hand sides of (5.6) we obtain

$$\begin{aligned} 0 &= q_k^T(\beta_1 a_1 + \cdots + \beta_k a_k) \\ &= \beta_1 q_k^T a_1 + \cdots + \beta_{k-1} q_k^T a_{k-1} + \beta_k q_k^T a_k \\ &= \beta_k \|\tilde{q}_k\|, \end{aligned}$$

where we use $q_k^T a_k = \|\tilde{q}_k\|$ in the last line. We conclude that $\beta_k = 0$.

From (5.6) and $\beta_k = 0$ we have

$$\beta_1 a_1 + \cdots + \beta_{k-1} a_{k-1} = 0.$$

We now repeat the argument above to conclude that $\beta_{k-1} = 0$. Repeating it k times we conclude that all β_i are zero.

Early termination. Suppose that the Gram–Schmidt algorithm terminates prematurely, in iteration j, because $\tilde{q}_j = 0$. The conclusions 1–4 above hold for $i = 1, \ldots, j-1$, since in those steps $\tilde{q}_i$ is nonzero. Since $\tilde{q}_j = 0$, we have

$$a_j = (q_1^T a_j) q_1 + \cdots + (q_{j-1}^T a_j) q_{j-1},$$

which shows that a_j is a linear combination of $q_1, \ldots, q_{j-1}$. But each of these vectors is in turn a linear combination of $a_1, \ldots, a_{j-1}$, by conclusion 3 above. Then a_j is a linear combination of $a_1, \ldots, a_{j-1}$, since it is a linear combination of linear combinations of them (see exercise 1.18). This means that $a_1, \ldots, a_j$ are linearly dependent, which implies that the larger set $a_1, \ldots, a_k$ is linearly dependent.

In summary, the Gram–Schmidt algorithm gives us an explicit method for determining if a list of vectors is linearly dependent or independent.

Example. We define three vectors

$$a_1 = (-1, 1, -1, 1), \qquad a_2 = (-1, 3, -1, 3), \qquad a_3 = (1, 3, 5, 7).$$

Applying the Gram–Schmidt algorithm gives the following results.

- $i = 1$. We have $\|\tilde{q}_1\| = 2$, so
$$q_1 = \frac{1}{\|\tilde{q}_1\|} \tilde{q}_1 = (-1/2, 1/2, -1/2, 1/2),$$
which is simply a_1 normalized.
- $i = 2$. We have $q_1^T a_2 = 4$, so
$$\tilde{q}_2 = a_2 - (q_1^T a_2) q_1 = \begin{bmatrix} -1 \\ 3 \\ -1 \\ 3 \end{bmatrix} - 4 \begin{bmatrix} -1/2 \\ 1/2 \\ -1/2 \\ 1/2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$

which is indeed orthogonal to q_1 (and a_1). It has norm $\|\tilde{q}_2\| = 2$; normalizing it gives

$$q_2 = \frac{1}{\|\tilde{q}_2\|}\tilde{q}_2 = (1/2, 1/2, 1/2, 1/2).$$

- $i = 3$. We have $q_1^T a_3 = 2$ and $q_2^T a_3 = 8$, so

$$\begin{aligned} \tilde{q}_3 &= a_3 - (q_1^T a_3)q_1 - (q_2^T a_3)q_2 \\ &= \begin{bmatrix} 1 \\ 3 \\ 5 \\ 7 \end{bmatrix} - 2 \begin{bmatrix} -1/2 \\ 1/2 \\ -1/2 \\ 1/2 \end{bmatrix} - 8 \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix} \\ &= \begin{bmatrix} -2 \\ -2 \\ 2 \\ 2 \end{bmatrix}, \end{aligned}$$

which is orthogonal to q_1 and q_2 (and a_1 and a_2). We have $\|\tilde{q}_3\| = 4$, so the normalized vector is

$$q_3 = \frac{1}{\|\tilde{q}_3\|}\tilde{q}_3 = (-1/2, -1/2, 1/2, 1/2).$$

Completion of the Gram–Schmidt algorithm without early termination tells us that the vectors a_1, a_2, a_3 are linearly independent.

Determining if a vector is a linear combination of linearly independent vectors. Suppose the vectors $a_1, \ldots, a_k$ are linearly independent, and we wish to determine if another vector b is a linear combination of them. (We have already noted on page 91 that if it is a linear combination of them, the coefficients are unique.) The Gram–Schmidt algorithm provides an explicit way to do this. We apply the Gram–Schmidt algorithm to the list of $k+1$ vectors

$$a_1, \ldots, a_k, b.$$

These vectors are linearly dependent if b is a linear combination of $a_1, \ldots, a_k$; they are linearly independent if b is not a linear combination of $a_1, \ldots, a_k$. The Gram–Schmidt algorithm will determine which of these two cases holds. It cannot terminate in the first k steps, since we assume that $a_1, \ldots, a_k$ are linearly independent. It will terminate in the $(k+1)$st step with $\tilde{q}_{k+1} = 0$ if b is a linear combination of $a_1, \ldots, a_k$. It will not terminate in the $(k+1)$st step (*i.e.*, $\tilde{q}_{k+1} \neq 0$), otherwise.

Checking if a collection of vectors is a basis. To check if the n-vectors $a_1, \ldots, a_n$ are a basis, we run the Gram–Schmidt algorithm on them. If Gram–Schmidt terminates early, they are not a basis; if it runs to completion, we know they are a basis.

Complexity of the Gram–Schmidt algorithm. We now derive an operation count for the Gram–Schmidt algorithm. In the first step of iteration i of the algorithm, $i-1$ inner products

$$q_1^T a_i, \quad \ldots, \quad q_{i-1}^T a_i$$

between vectors of length n are computed. This takes $(i-1)(2n-1)$ flops. We then use these inner products as the coefficients in $i-1$ scalar multiplications with the vectors $q_1, \ldots, q_{i-1}$. This requires $n(i-1)$ flops. We then subtract the $i-1$ resulting vectors from a_i, which requires another $n(i-1)$ flops. The total flop count for step 1 is

$$(i-1)(2n-1) + n(i-1) + n(i-1) = (4n-1)(i-1)$$

flops. In step 3 we compute the norm of $\tilde{q}_i$, which takes approximately $2n$ flops. We then divide $\tilde{q}_i$ by its norm, which requires n scalar divisions. So the total flop count for the ith iteration is $(4n-1)(i-1)+3n$ flops.

The total flop count for all k iterations of the algorithm is obtained by summing our counts for $i = 1, \ldots, k$:

$$\sum_{i=1}^{k} ((4n-1)(i-1)+3n) = (4n-1)\frac{k(k-1)}{2} + 3nk \approx 2nk^2,$$

where we use the fact that

$$\sum_{i=1}^{k} (i-1) = 1 + 2 + \cdots + (k-2) + (k-1) = \frac{k(k-1)}{2}, \tag{5.7}$$

which we justify below. The complexity of the Gram–Schmidt algorithm is $2nk^2$; its order is nk^2. We can guess that its running time grows linearly with the lengths of the vectors n, and quadratically with the number of vectors k.

In the special case of $k = n$, the complexity of the Gram–Schmidt method is $2n^3$. For example, if the Gram–Schmidt algorithm is used to determine whether a collection of 1000 1000-vectors is linearly independent (and therefore a basis), the computational cost is around 2×10^9 flops. On a modern computer, can we can expect this to take on the order of one second.

A famous anecdote alleges that the formula (5.7) was discovered by the mathematician Carl Friedrich Gauss when he was a child, although it was known before that time. Here is his argument, for the case when k is odd. Lump the first entry in the sum together with the last entry, the second entry together with the second-to-last entry, and so on. Each of these pairs of numbers adds up to k; since there are $(k-1)/2$ such pairs, the total is $k(k-1)/2$. A similar argument works when k is even.

Modified Gram–Schmidt algorithm. When the Gram–Schmidt algorithm is implemented, a variation on it called the *modified Gram–Schmidt* algorithm is typically used. This algorithm produces the same results as the Gram–Schmidt algorithm (5.1), but is less sensitive to the small round-off errors that occur when arithmetic calculations are done using floating point numbers. (We do not consider round-off error in floating-point computations in this book.)