Mathematically, (8.6) and (8.4) are equivalent. However, the sequences of arithmetic operations implied by these formulas are different. The modified algorithm calculates v_j by evaluating the following formulas in order:

$$\begin{aligned}
v_j^{(1)} &= a_j, \\
v_j^{(2)} &= P_{\perp q_1} v_j^{(1)} &&= v_j^{(1)} - q_1 q_1^* v_j^{(1)}, \\
v_j^{(3)} &= P_{\perp q_2} v_j^{(2)} &&= v_j^{(2)} - q_2 q_2^* v_j^{(2)}, \\
&\;\;\vdots && \;\;\vdots \\
v_j = v_j^{(j)} &= P_{\perp q_{j-1}} v_j^{(j-1)} &&= v_j^{(j-1)} - q_{j-1} q_{j-1}^* v_j^{(j-1)}.
\end{aligned} \tag{8.7}$$

In finite precision computer arithmetic, we shall see that (8.7) introduces smaller errors than (8.4).

When the algorithm is implemented, the projector $P_{\perp q_i}$ can be conveniently applied to $v_j^{(i)}$ for each $j > i$ immediately after q_i is known. This is done in the description below.

Algorithm 8.1. Modified Gram–Schmidt

```
for $i = 1$ to $n$
    $v_i = a_i$
for $i = 1$ to $n$
    $r_{ii} = \|v_i\|$
    $q_i = v_i / r_{ii}$
    for $j = i + 1$ to $n$
        $r_{ij} = q_i^* v_j$
        $v_j = v_j - r_{ij} q_i$
```

In practice, it is common to let v_i overwrite a_i and q_i overwrite v_i in order to save storage.

The reader should compare Algorithms 7.1 and 8.1 until he or she is confident of their equivalence.

Operation Count

The Gram–Schmidt algorithm is the first algorithm we have presented in this book, and with any algorithm, it is important to assess its cost. To do so, throughout the book we follow the classical route and count the number of floating point operations— *"flops"*—that the algorithm requires. Each addition, subtraction, multiplication, division, or square root counts as one flop.