Chapter 4
Orthogonality

Even if the Gaussian elimination procedure for solving linear systems of equations and normal equations is a standard algorithm with widespread use in numerous applications, it is not sufficient in situations when one needs to separate the most important information from less important information ("noise"). The typical linear algebra formulation of "data quality" is to quantify the concept of "good and bad basis vectors"; loosely speaking, good basis vectors are those that are "very linearly independent," i.e., close to orthogonal. In the same vein, vectors that are almost linearly dependent are bad basis vectors. In this chapter we will introduce some theory and algorithms for computations with orthogonal vectors. A more complete quantification of the "quality" of a set of vectors is given in Chapter 6.

Example 4.1. In Example 3.13 we saw that an unsuitable choice of basis vectors in the least squares problem led to ill-conditioned normal equations. Along similar lines, define the two matrices

$$A = \begin{pmatrix} 1 & 1.05 \\ 1 & 1 \\ 1 & 0.95 \end{pmatrix}, \qquad B = \begin{pmatrix} 1 & 1/\sqrt{2} \\ 1 & 0 \\ 1 & -1/\sqrt{2} \end{pmatrix},$$

whose columns are plotted in Figure 4.1. It can be shown that the column vectors of the two matrices span the same plane in $\mathbb{R}^3$. From the figure it is clear that the columns of B, which are orthogonal, determine the plane much better than the columns of A, which are quite close. ∎

From several points of view, it is advantageous to use orthogonal vectors as basis vectors in a vector space. In this chapter we will list some important properties of orthogonal sets of vectors and orthogonal matrices. We assume that the vectors are in $\mathbb{R}^m$ with $m \geq n$.

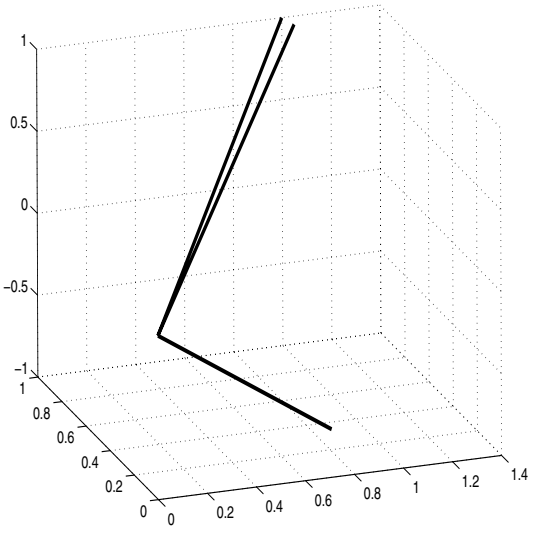

Figure 4.1. *Three vectors spanning a plane in* $\mathbb{R}^3$.

4.1 Orthogonal Vectors and Matrices

We first recall that two nonzero vectors x and y are called *orthogonal* if $x^T y = 0$ (i.e., $\cos\theta(x, y) = 0$).

Proposition 4.2. *Let* q_j, $j = 1, 2, \ldots, n$, *be orthogonal, i.e.,* $q_i^T q_j = 0$, $i \neq j$. *Then they are linearly independent.*

Proof. Assume they are linearly dependent. Then from Proposition 2.2 there exists a q_k such that

$$q_k = \sum_{j \neq k} \alpha_j q_j.$$

Multiplying this equation by q_k^T we get

$$q_k^T q_k = \sum_{j \neq k} \alpha_j q_k^T q_j = 0,$$

since the vectors are orthogonal. This is a contradiction. ∎

Let the set of orthogonal vectors q_j, $j = 1, 2, \ldots, m$, in $\mathbb{R}^m$ be normalized,

$$\|q_j\|_2 = 1.$$

Then they are called *orthonormal*, and they constitute an *orthonormal basis* in $\mathbb{R}^m$.

A square matrix

$$Q = \begin{pmatrix} q_1 & q_2 & \cdots & q_m \end{pmatrix} \in \mathbb{R}^{m \times m}$$

whose columns are orthonormal is called an *orthogonal matrix*. Orthogonal matrices satisfy a number of important properties that we list in a sequence of propositions.

Proposition 4.3. *An orthogonal matrix Q satisfies $Q^TQ = I$.*

Proof.

$$Q^TQ = \begin{pmatrix} q_1 & q_2 & \cdots & q_m \end{pmatrix}^T \begin{pmatrix} q_1 & q_2 & \cdots & q_m \end{pmatrix} = \begin{pmatrix} q_1^T \\ q_2^T \\ \cdots \\ q_m^T \end{pmatrix} \begin{pmatrix} q_1 & q_2 & \cdots & q_m \end{pmatrix}$$

$$= \begin{pmatrix} q_1^Tq_1 & q_1^Tq_2 & \cdots & q_1^Tq_m \\ q_2^Tq_1 & q_2^Tq_2 & \cdots & q_2^Tq_m \\ \vdots & \vdots & & \vdots \\ q_m^Tq_1 & q_m^Tq_2 & \cdots & q_m^Tq_m \end{pmatrix} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix},$$

due to orthonormality. ∎

The orthogonality of its columns implies that an orthogonal matrix has full rank, and it is trivial to find the inverse.

Proposition 4.4. *An orthogonal matrix $Q \in \mathbb{R}^{m\times m}$ has rank m, and, since $Q^TQ = I$, its inverse is equal to $Q^{-1} = Q^T$.*

Proposition 4.5. *The rows of an orthogonal matrix are orthogonal, i.e., $QQ^T = I$.*

Proof. Let x be an arbitrary vector. We shall show that $QQ^Tx = x$. Given x there is a uniquely determined vector y, such that $Qy = x$, since Q^{-1} exists. Then

$$QQ^Tx = QQ^TQy = Qy = x.$$

Since x is arbitrary, it follows that $QQ^T = I$. ∎

Proposition 4.6. *The product of two orthogonal matrices is orthogonal.*

Proof. Let Q and P be orthogonal, and put $X = PQ$. Then

$$X^TX = (PQ)^TPQ = Q^TP^TPQ = Q^TQ = I. \quad ∎$$

Any orthonormal basis of a subspace of $\mathbb{R}^m$ can be enlarged to an orthonormal basis of the whole space. The next proposition shows this in matrix terms.

Proposition 4.7. *Given a matrix $Q_1 \in \mathbb{R}^{m\times k}$, with orthonormal columns, there exists a matrix $Q_2 \in \mathbb{R}^{m\times(m-k)}$ such that $Q = (Q_1\, Q_2)$ is an orthogonal matrix.*

This proposition is a standard result in linear algebra. We will later demonstrate how Q can be computed.

One of the most important properties of orthogonal matrices is that they preserve the length of a vector.

Proposition 4.8. *The Euclidean length of a vector is invariant under an orthogonal transformation Q.*

Proof. $\|Qx\|_2^2 = (Qx)^T Qx = x^T Q^T Qx = x^T x = \|x\|_2^2$. ∎

Also the corresponding matrix norm and the Frobenius norm are *invariant under orthogonal transformations.*

Proposition 4.9. *Let $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ be orthogonal. Then for any $A \in \mathbb{R}^{m\times n}$,*

$$\|\,UAV\,\|_2 = \|\,A\,\|_2,$$
$$\|\,UAV\,\|_F = \|\,A\,\|_F.$$

Proof. The first equality is easily proved using Proposition 4.8. The second is proved using the alternative expression (2.8) for the Frobenius norm and the identity $\mathrm{tr}(BC) = \mathrm{tr}(CB)$. ∎

4.2 Elementary Orthogonal Matrices

We will use elementary orthogonal matrices to reduce matrices to compact form. For instance, we will transform a matrix $A \in \mathbb{R}^{m\times n}$, $m \geq n$, to triangular form.

4.2.1 Plane Rotations

A 2×2 plane rotation matrix[6]

$$G = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}, \qquad c^2 + s^2 = 1,$$

is orthogonal. Multiplication of a vector x by G rotates the vector in a clockwise direction by an angle θ, where $c = \cos\theta$. A plane rotation can be used to zero the second element of a vector x by choosing $c = x_1/\sqrt{x_1^2 + x_2^2}$ and $s = x_2/\sqrt{x_1^2 + x_2^2}$:

$$\frac{1}{\sqrt{x_1^2 + x_2^2}} \begin{pmatrix} x_1 & x_2 \\ -x_2 & x_1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \sqrt{x_1^2 + x_2^2} \\ 0 \end{pmatrix}.$$

By embedding a two-dimensional rotation in a larger unit matrix, one can manipulate vectors and matrices of arbitrary dimension.

[6] In the numerical literature, plane rotations are often called Givens rotations, after Wallace Givens, who used them for eigenvalue computations around 1960. However, they had been used long before that by Jacobi, also for eigenvalue computations.

Example 4.10. We can choose c and s in

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c & 0 & s \\ 0 & 0 & 1 & 0 \\ 0 & -s & 0 & c \end{pmatrix}$$

so that we zero element 4 in a vector $x \in \mathbb{R}^4$ *by a rotation in plane* $(2, 4)$. Execution of the MATLAB script

```
x=[1;2;3;4];
sq=sqrt(x(2)^2+x(4)^2);
c=x(2)/sq; s=x(4)/sq;
G=[1 0 0 0; 0 c 0 s; 0 0 1 0; 0 -s 0 c];
y=G*x
```

gives the result

```
y = 1.0000
    4.4721
    3.0000
         0
```

Using a sequence of plane rotations, we can now transform an arbitrary vector to a multiple of a unit vector. This can be done in several ways. We demonstrate one in the following example.

Example 4.11. Given a vector $x \in \mathbb{R}^4$, we transform it to κe_1. First, by a rotation G_3 in the plane $(3, 4)$ we zero the last element:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & c_1 & s_1 \\ 0 & 0 & -s_1 & c_1 \end{pmatrix} \begin{pmatrix} \times \\ \times \\ \times \\ \times \end{pmatrix} = \begin{pmatrix} \times \\ \times \\ * \\ 0 \end{pmatrix}.$$

Then, by a rotation G_2 in the plane $(2, 3)$ we zero the element in position 3:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c_2 & s_2 & 0 \\ 0 & -s_2 & c_2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \times \\ \times \\ \times \\ 0 \end{pmatrix} = \begin{pmatrix} \times \\ * \\ 0 \\ 0 \end{pmatrix}.$$

Finally, the second element is annihilated by a rotation G_1:

$$\begin{pmatrix} c_3 & s_3 & 0 & 0 \\ -s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \times \\ \times \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \kappa \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

According to Proposition 4.8 the Euclidean length is preserved, and therefore we know that $\kappa = \|x\|_2$.

We summarize the transformations. We have

$$\kappa e_1 = G_1(G_2(G_3 x)) = (G_1 G_2 G_3)x.$$

Since the product of orthogonal matrices is orthogonal (Proposition 4.6) the matrix $P = G_1 G_2 G_3$ is orthogonal, and the overall result is $Px = \kappa e_1$. ∎

Plane rotations are very flexible and can be used efficiently for problems with a sparsity structure, e.g., band matrices. On the other hand, for dense matrices they require more flops than Householder transformations; see Section 4.3.

Example 4.12. In the MATLAB example earlier in this section we explicitly embedded the 2×2 in a matrix of larger dimension. This is a waste of operations, since the computer execution of the code does not take into account the fact that only two rows of the matrix are changed. Instead the whole matrix multiplication is performed, which requires $2n^3$ flops in the case of matrices of dimension n. The following two MATLAB functions illustrate how the rotation should be implemented to save operations (and storage):

```
function [c,s]=rot(x,y);
  % Construct a plane rotation that zeros the second
  % component in the vector [x;y]' (x and y are scalars)
  sq=sqrt(x^2 + y^2);
  c=x/sq; s=y/sq;

function X=approt(c,s,i,j,X);
  % Apply a plane (plane) rotation in plane (i,j)
  % to a matrix X
  X([i,j],:)=[c s; -s c]*X([i,j],:);
```

The following script reduces the vector x to a multiple of the standard basis vector e_1:

```
x=[1;2;3;4];
for i=3:-1:1
  [c,s]=rot(x(i),x(i+1));
  x=approt(c,s,i,i+1,x);
end

>> x = 5.4772
            0
            0
            0
```

After the reduction the first component of x is equal to $\| x \|_2$. ∎

4.2.2 Householder Transformations

Let $v \neq 0$ be an arbitrary vector, and put

$$P = I - \frac{2}{v^T v} v v^T;$$

P is symmetric and orthogonal (verify this by a simple computation!). Such matrices are called *reflection matrices* or *Householder transformations*. Let x and y be given vectors of the same length, $\|x\|_2 = \|y\|_2$, and ask the question, "Can we determine a Householder transformation P such that $Px = y$?"

The equation $Px = y$ can be written

$$x - \frac{2v^T x}{v^T v} v = y,$$

which is of the form $\beta v = x - y$. Since v enters P in such a way that a factor β cancels, we can choose $\beta = 1$. With $v = x - y$ we get

$$v^T v = x^T x + y^T y - 2x^T y = 2(x^T x - x^T y),$$

since $x^T x = y^T y$. Further,

$$v^T x = x^T x - y^T x = \frac{1}{2} v^T v.$$

Therefore we have

$$Px = x - \frac{2v^T x}{v^T v} v = x - v = y,$$

as we wanted. In matrix computations we often want to zero elements in a vector and we now choose $y = \kappa e_1$, where $\kappa = \pm\|x\|_2$, and $e_1^T = \begin{pmatrix} 1 & 0 & \cdots & 0 \end{pmatrix}$. The vector v should be taken equal to

$$v = x - \kappa e_1.$$

In order to avoid cancellation (i.e., the subtraction of two close floating point numbers), we choose $\operatorname{sign}(\kappa) = -\operatorname{sign}(x_1)$. Now that we have computed v, we can simplify and write

$$P = I - \frac{2}{v^T v} v v^T = I - 2uu^T, \qquad u = \frac{1}{\|v\|_2} v.$$

Thus the Householder vector u has length 1. The computation of the Householder vector can be implemented in the following MATLAB code:

```
function u=househ(x)
  % Compute the Householder vector u such that
  % (I - 2 u * u')x = k*e_1, where
  % |k| is equal to the euclidean norm of x
  % and e_1 is the first unit vector
  n=length(x);      % Number of components in x
  kap=norm(x); v=zeros(n,1);
  v(1)=x(1)+sign(x(1))*kap;
  v(2:n)=x(2:n);
  u=(1/norm(v))*v;
```

In most cases one should avoid forming the Householder matrix P explicitly, since it can be represented much more compactly by the vector u. Multiplication by P should be done according to $Px = x - (2u^T x)u$, where the matrix-vector multiplication requires $4n$ flops (instead of $O(n^2)$ if P were formed explicitly). The matrix multiplication PX is done

$$PX = A - 2u(u^T X). \tag{4.1}$$

Multiplication by a Householder transformation is implemented in the following code:

```
function Y=apphouse(u,X);
  % Multiply the matrix X by a Householder matrix
  % Y = (I - 2 * u * u') * X
  Y=X-2*u*(u'*X);
```

Example 4.13. The first three elements of the vector $x = \begin{pmatrix}1 & 2 & 3 & 4\end{pmatrix}^T$ are zeroed by the following sequence of MATLAB statements:

```
>>    x=[1; 2; 3; 4];
>>    u=househ(x);
>>    y=apphouse(u,x)

    y = -5.4772
              0
              0
              0
```

As plane rotations can be embedded in unit matrices, in order to apply the transformation in a structured way, we similarly can embed Householder transformations. Assume, for instance that we have transformed the first column in a matrix to a unit vector and that we then want to zero all the elements in the second column below the main diagonal. Thus, in an example with a 5×4 matrix, we want to compute the transformation

$$P_2 A^{(1)} = P_2 \begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{pmatrix} = \begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{pmatrix} =: A^{(2)}. \tag{4.2}$$

Partition the second column of $A^{(1)}$ as follows:

$$\begin{pmatrix} a_{12}^{(1)} \\ a_{\cdot 2}^{(1)} \end{pmatrix},$$

where $a_{12}^{(1)}$ is a scalar. We know how to transform $a_{\cdot 2}^{(1)}$ to a unit vector; let $\hat{P}_2$ be a Householder transformation that does this. Then the transformation (4.2) can be implemented by embedding $\hat{P}_2$ in a unit matrix:

$$P_2 = \begin{pmatrix} 1 & 0 \\ 0 & \hat{P}_2 \end{pmatrix}. \tag{4.3}$$

It is obvious that P_2 leaves the first row of $A^{(1)}$ unchanged and computes the transformation (4.2). Also, it is easy to see that the newly created zeros in the first column are not destroyed.

Similar to the case of plane rotations, one should not explicitly embed a Householder transformation in an identity matrix of larger dimension. Instead one should apply it to the rows (in the case of multiplication from the left) that are affected in the transformation.

Example 4.14. The transformation in (4.2) is done by the following statements.

```
u=househ(A(2:m,2)); A(2:m,2:n)=apphouse(u,A(2:m,2:n));

>> A = -0.8992   -0.6708   -0.7788   -0.9400
       -0.0000    0.3299    0.7400    0.3891
       -0.0000    0.0000   -0.1422   -0.6159
       -0.0000   -0.0000    0.7576    0.1632
       -0.0000   -0.0000    0.3053    0.4680
```

∎

4.3 Number of Floating Point Operations

We shall compare the number of flops to transform the first column of an $m \times n$ matrix A to a multiple of a unit vector κe_1 using plane and Householder transformations. Consider first plane rotations. Obviously, the computation of

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} cx + sy \\ -sx + cy \end{pmatrix}$$

requires four multiplications and two additions, i.e., six flops. Applying such a transformation to an $m \times n$ matrix requires $6n$ flops. In order to zero all elements

but one in the first column of the matrix, we apply $m-1$ rotations. Thus the overall flop count is $6(m-1)n \approx 6mn$.

If the corresponding operation is performed by a Householder transformation as in (4.1), then only $4mn$ flops are needed. (Note that multiplication by 2 is not a flop, as it is implemented by the compiler as a shift, which is much faster than a flop; alternatively one can scale the vector u by $\sqrt{2}$.)

4.4 Orthogonal Transformations in Floating Point Arithmetic

Orthogonal transformations are very stable in floating point arithmetic. For instance, it can be shown [50, p. 367] that a computed Householder transformation in floating point $\hat{P}$ that approximates an exact P satisfies

$$\| P - \hat{P} \|_2 = O(\mu),$$

where μ is the unit round-off of the floating point system. We also have the backward error result

$$fl(\hat{P}A) = P(A+E), \qquad \| E \|_2 = O(\mu \| A \|_2).$$

Thus the floating point result is equal to the product of the exact orthogonal matrix and a data matrix that has been perturbed by a very small amount. Analogous results hold for plane rotations.