Solution of Linear Equations

(Com S 477/577 Notes)

Yan-Bin Jia

Sep 8, 2020

We have discussed general methods for solving arbitrary equations, and looked at the special class of polynomial equations. A subclass of the latter comprises all the systems of linear equations to which the area of linear algebra is devoted. In fact, many a problem in numerical analysis can be reduced to one of solving a system of linear equations. We already witnessed this in the use of Newton's method to solve a system of nonlinear equations. Other applications include solution of ordinary or partial differential equations (ODEs or PDEs), the eigenvalue problems of mathematical analysis, least-squares fitting of data, and polynomial approximation.

1 Elements of Linear Algebra

Recall that a *basis* for a vector space is a sequence of vectors that are linearly independent and span the space. Given a linear function $f : \mathbb{R}^n \to \mathbb{R}^m$, and a pair of bases as below:

$$
\begin{aligned}
\mathcal{B}_1 &= \{\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_n\}, && \text{for } \mathbb{R}^n, \\
\mathcal{B}_2 &= \{\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_m\}, && \text{for } \mathbb{R}^m,
\end{aligned}
$$

we can represent f by an $m \times n$ matrix A such that $f(\boldsymbol{x}) = A\boldsymbol{x}$ for any $\boldsymbol{x} \in \mathbb{R}^n$. Note that the matrix A depends on the choices of the bases $\mathcal{B}_1$ and $\mathcal{B}_2$.

The *rank* r of an $m \times n$ matrix A is the number of independent rows. The *column space* of A, denoted $\text{col}(A)$, consists of all the linear combinations of its columns. The *row space* of A, denoted $\text{row}(A)$, consists of all the linear combinations of its rows. Since each column has m components, the column space of A is a subspace of $\mathbb{R}^m$. It consists of all the points in $\mathbb{R}^m$ that are image vectors under the mapping A. Similarly, the row space is a subspace of $\mathbb{R}^n$.

The *null space* $\text{null}(A)$ of the matrix is made up of all the solutions to $A\boldsymbol{x} = \boldsymbol{0}$, where $x \in \mathbb{R}^n$.

Theorem 1 (Fundamental Theorem of Linear Allegra) *Let A be an $m \times n$ matrix. Both its row and column spaces have dimension r. Its null space has dimension $n - r$ and is the orthogonal complement of its row space (in $\mathbb{R}^n$). In other words,*

$$
\begin{aligned}
\mathbb{R}^n &= \text{row}(A) \bigoplus \text{null}(A), \\
n &= \dim(\text{row}(A)) + \dim(\text{null}(A)).
\end{aligned}
$$

Consider the system of equations

$$A\boldsymbol{x} = \boldsymbol{b}.$$

If $\boldsymbol{b}$ is not an element of $\mathrm{col}(A)$, then the system is *inconsistent* (or *overdetermined*). If $\boldsymbol{b} \in \mathrm{col}(A)$ and $\mathrm{null}(A)$ is non-trivial, then we say that the system is *underdetermined*. In this case, every solution $\boldsymbol{x}$ can be split into a row space component $\boldsymbol{x}_r$ and a null space component $\boldsymbol{x}_n$ so that

$$\begin{aligned} A\boldsymbol{x} &= A(\boldsymbol{x}_r + \boldsymbol{x}_n) \\ &= A\boldsymbol{x}_r + A\boldsymbol{x}_n \\ &= A\boldsymbol{x}_r. \end{aligned}$$

The null space goes to zero, $A\boldsymbol{x}_n = 0$, while the row space component goes to the column space, $A\boldsymbol{x}_r = A\boldsymbol{x}$.

If A is an $n \times n$ square matrix, we say that A is *singular*

iff $\det(A) = 0$
iff $\mathrm{rank}(A) < n$
iff the rows of A are not linearly independent
iff the columns of A are not linearly independent
iff the dimension of the null space of A is non-zero
iff A is not invertible.

2 LU Decomposition

An $m \times n$ matrix A, where $m \geq n$, can be written in the form

$$PA = LDU,$$

where

P is an $m \times m$ permutation matrix that specifies row interchanges,
L is an $m \times m$ square lower-triangular matrix with 1's on the diagonal,
U is an $m \times n$ upper-triangular matrix with 1's on the diagonal,
D is an $m \times m$ square diagonal matrix.

1. The entries on the diagonal of D are called "pivots" (named after the Gaussian elimination procedure).

2. When A is a square matrix, the product of the pivots is equal to $\pm\det(A)$, where the sign "$-$" is chosen if odd number of row interchanges are performed and the sign "$+$" is chosen otherwise.

3. If A is symmetric and $P = I$, the identity matrix, then $U = L^\top$.

4. If A is symmetric and positive definite, then $U = L^\top$ and the diagonal entries of D are strictly positive.

Example 3.

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix};$$

$$\begin{pmatrix} 1 & 1 & 0 \\ 2 & 1 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}.$$

Like most other decompositions and factorizations, the LU (or LDU) decomposition is used to simplify the solution of the system

$$A\boldsymbol{x} = \boldsymbol{b}.$$

Suppose A is square and non-singular, solving the above system is equivalent to solving

$$LDU\boldsymbol{x} = P\boldsymbol{b}.$$

We then first solve

$$L\boldsymbol{y} = P\boldsymbol{b}$$

for the vector $\boldsymbol{y}$ and then solve

$$U\boldsymbol{x} = D^{-1}\boldsymbol{y},$$

for the vector $\boldsymbol{x}$. Each of the above systems can be solved easily using forward or backward substitution.

2.1 Crout's Algorithm

The LU decomposition for an $n \times n$ square matrix A can be generated directly by Gaussian elimination. Nevertheless, a more efficient procedure is *Crout's algorithm.* In case no pivoting is needed, the algorithm yields two matrices $L = \{l_{ij}\}$ and $U = \{u_{ij}\}$ whose product is $A = \{a_{ij}\}$, namely,

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ l_{21} & 1 & 0 & \cdots & 0 \\ l_{31} & l_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ 0 & 0 & u_{33} & \cdots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & u_{nn} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix}.$$

The algorithm solves for L and U simultaneously and column by column.[1] At the jth outer iteration step, it generates column j of U and then column j of L.

for $j = 1, 2, \ldots, n$ **do**
  for $i = 1, 2, \ldots, j$ **do**
    $u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj}$
  for $i = j+1, j+2, \ldots, n$ **do**
    $l_{ij} = \frac{1}{u_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj} \right)$

[1] You can also do it row by row except row i of L has to be determined before row i of U.

If you work through a few iterations of the above procedure, you will see that the α's and β's that occur on the right-hand side of the two equations in the procedure are already determined by the time they are needed. And every a_{ij} is used only once and never again. Together these entries are used column by column as well. For compactness, we can store l_{ij} and u_{ij} in the location a_{ij} used to occupy, namely,

$$\begin{pmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ l_{21} & u_{22} & u_{23} & \cdots & u_{2n} \\ l_{31} & l_{32} & u_{33} & \cdots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & u_{nn} \end{pmatrix}$$

Looking at step 5 of Crout's algorithm, we should be worried about the possibility of u_{jj} becoming zero. Here is an example from [2, p. 97] showing a matrix with no LU decomposition due to this degeneracy. Suppose

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 7 \\ 3 & 5 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}.$$

We must have

$$u_{11} = 1, \quad l_{21} = 2, \quad l_{31} = 3, \quad u_{12} = 2, \quad \text{and} \quad u_{22} = 0.$$

The $(3, 2)$ entry determined from the product matrix on the right hand side is

$$l_{31}u_{12} + l_{32}u_{22} = 6.$$

It is not equal to the $(3, 2)$ entry (value 5) of the original matrix! The contradiction arised because $u_{22} = 0$. In fact, we would not even be able to continue Crout's algorithm to calculate l_{32} via a division by 0. The following theorem on the existence of the LU decomposition is given [2, p. 97]:

Theorem 2 *An $n \times n$ matrix A has an LU factorization if* $\det(A_i) \neq 0$*, where A_i is the upper left $i \times i$ submatrix, for $i = 1, \dots, n-1$. If the LU factorization exists and* $\det(A) \neq 0$*, then it is unique.*

For numerical stability, pivoting should be performed in Crout's algorithm. The key point is to notice that the first equation in the procedure for u_{ij} is exactly the same as the second equation for l_{ij} except for the division in the latter equation. This means that we can choose the largest

$$a_{ij} - \sum_{k=1}^{j-1} a_{ik}u_{kj}, \qquad i = j, \dots, n$$

as the diagonal element u_{jj} and switch corresponding rows in L and A.

Example 4. To illustrate on pivoting, let us carry out a few steps of Crout's algorithm on the matrix

$$\begin{pmatrix} 2 & -7 & 6 & 5 \\ 4 & 8 & -10 & 3 \\ 9 & -6 & -4 & 2 \\ 5 & 1 & 3 & 3 \end{pmatrix}.$$

In the first step, we need to determine u_{11}. Which of rows 1, 2, 3, 4 would result in the largest (absolute) value of u_{11}?

$$\begin{array}{ll} \text{if row 1} & 1 \cdot u_{11} = 2 \longrightarrow u_{11} = 2, \\ \text{if row 2} & 1 \cdot u_{11} = 4 \longrightarrow u_{11} = 4, \\ \text{if row 3} & 1 \cdot u_{11} = 9 \longrightarrow u_{11} = 9, \\ \text{if row 2} & 1 \cdot u_{11} = 5 \longrightarrow u_{11} = 5. \end{array}$$

Thus we set $u_{11} = 9$ and exchange rows 1 and 3 in A:

$$\begin{pmatrix} 9 & -6 & -4 & 2 \\ 4 & 8 & -10 & 3 \\ 2 & -7 & 6 & 5 \\ 5 & 1 & 3 & 3 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & -7 & 6 & 5 \\ 4 & 8 & -10 & 3 \\ 9 & -6 & -4 & 2 \\ 5 & 1 & 3 & 3 \end{pmatrix},$$

where the first matrix on the right hand side is a permutation matrix which exchanges rows 1 and 3 of the second (original) matrix via multiplication. In the second step, we use the first column to determine that

$$l_{21} = \frac{4}{9}, \qquad l_{31} = \frac{2}{9}, \qquad \text{and} \qquad l_{41} = \frac{5}{9}.$$

Next, we let $u_{12} = a_{12} = -6$ and matrices L and U take the form

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{4}{9} & & & \\ \frac{2}{9} & & & \\ \frac{5}{9} & & & \end{pmatrix} \begin{pmatrix} 9 & -6 & & \\ 0 & u_{22} & & \\ 0 & 0 & & \\ 0 & 0 & & \end{pmatrix}.$$

To determine u_{22}, we find out which of rows 2, 3, 4 would result in the largest u_{22} value:

$$\begin{array}{ll} \text{if row 2} & \frac{4}{9} \cdot (-6) + u_{22} = 8 \longrightarrow u_{22} = \frac{32}{3}, \\ \text{if row 3} & \frac{2}{9} \cdot (-6) + u_{22} = -7 \longrightarrow u_{22} = -\frac{17}{3}, \\ \text{if row 4} & \frac{5}{9} \cdot (-6) + u_{22} = 1 \longrightarrow u_{22} = \frac{13}{3}. \end{array}$$

Since row 2 yields the largest absolute value of u_{22}, we set $u_{22} = \frac{32}{3}$. Using u_{22}, we obtain the second column of L:

$$l_{32} = -\frac{17}{32} \qquad \text{and} \qquad l_{42} = \frac{13}{32}.$$

By this time, we have

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{4}{9} & 1 & 0 & 0 \\ \frac{2}{9} & -\frac{17}{32} & & \\ \frac{5}{9} & \frac{13}{32} & & \end{pmatrix} \begin{pmatrix} 9 & -6 & & \\ 0 & \frac{32}{3} & & \\ 0 & 0 & & \\ 0 & 0 & & \end{pmatrix} = \begin{pmatrix} 9 & -6 & -4 & 2 \\ 4 & 8 & -10 & 3 \\ 2 & -7 & 6 & 5 \\ 5 & 1 & 3 & 3 \end{pmatrix}.$$

3 Factorization Based on Eigenvalues

Suppose the $n \times n$ matrix A has n linearly independent eigenvectors, then

$$A = S\Lambda S^{-1},$$

where Λ is a diagonal matrix whose entries are the eigenvalues of A and S is an *eigenvector matrix* whose columns are the eigenvectors of A.

When the eigenvalues of A are all different, it is automatic that the eigenvectors are independent. Therefore A can be diagonalized.

Every $n \times n$ matrix can be decomposed into the *Jordan form*, that is,

$$A = MJM^{-1},$$

where

$$J = \begin{pmatrix} J_1 & & \\ & \ddots & \\ & & J_s \end{pmatrix}, \qquad J_i = \begin{pmatrix} \lambda_i & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{pmatrix}, \qquad 1 \le i \le s,$$

with λ_i an eigenvalue of A. Here s is the number of independent eigenvectors of A and M consists of eigenvectors and "generalized" eigenvectors.

4 QR Factorization

Suppose A is an $m \times n$ matrix with independent columns (hence $m \ge n$). We can factor A as

$$A = QR,$$

Here Q with dimensions $m \times n$ has the same column space as A but its columns are orthonormal vectors. In other words, $Q^\top Q = I$. And R with dimensions $n \times n$ is invertible and upper triangular.

The first application is the "QR algorithm" which repeatedly produces QR factorizations of matrices derived from A, building the eigenvalues of A in the process.

The second application is in the solution of an overconstrained system $A\boldsymbol{x} = \boldsymbol{b}$ in the least-squares sense. The least-squares solution $\bar{\boldsymbol{x}}$ is given by $\bar{\boldsymbol{x}} = (A^\top A)^{-1} A^\top \boldsymbol{b}$, assuming that the columns of A are independent. But $Q^\top Q = I$, so

$$\begin{aligned} \bar{\boldsymbol{x}} &= (R^\top Q^\top Q R)^{-1} R^\top Q^\top \boldsymbol{b} \\ &= (R^\top R)^{-1} R^\top Q^\top \boldsymbol{b} \\ &= R^{-1} (R^\top)^{-1} R^\top Q^\top \boldsymbol{b} \\ &= R^{-1} Q^\top \boldsymbol{b}. \end{aligned}$$

So we can obtain $\boldsymbol{x}$ by computing $Q^\top \boldsymbol{b}$ and then using backsubstitution to solve $R\bar{\boldsymbol{x}} = Q^\top \boldsymbol{b}$. This is numerically more stable than solving the system $A^\top A \bar{\boldsymbol{x}} = A^\top \boldsymbol{b}$.

The QR factorization can be be computed using the Gram-Schmidt process.

4.1 The Gram-Schmidt Procedure

Given n linearly independent vectors $\boldsymbol{v}_1, \dots, \boldsymbol{v}_n$, the Gram-Schmidt procedure constructs n orthonormal vectors $\hat{\boldsymbol{u}}_1, \dots, \hat{\boldsymbol{u}}_n$ such that these two sets of vectors span the same space. First, it constructs n orthogonal vectors $\boldsymbol{w}_1, \dots, \boldsymbol{w}_n$ below:

$$\boldsymbol{w}_1 = \boldsymbol{v}_1, \tag{1}$$

$$\boldsymbol{w}_j = \boldsymbol{v}_j - \sum_{i=1}^{j-1} \frac{\boldsymbol{v}_j^\top \boldsymbol{w}_i}{\boldsymbol{w}_i^\top \boldsymbol{w}_i} \boldsymbol{w}_i, \qquad j = 2, \dots, n. \tag{2}$$

Essentially, we subtract from every new vector $\boldsymbol{v}_j$ its projection in the directions $\boldsymbol{w}_1/\|\boldsymbol{w}_1\|$, $\ldots$, $\boldsymbol{w}_{i-1}/\|\boldsymbol{w}_{j-1}\|$ that are already set. Next, we simply perform a normalization by letting

$$\hat{\boldsymbol{u}}_j = \frac{\boldsymbol{w}_j}{\|\boldsymbol{w}_j\|}, \qquad j = 1, \ldots, n.$$

Example 5. Consider three vectors:

$$\boldsymbol{v}_1 = \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}, \qquad \boldsymbol{v}_2 = \begin{pmatrix} 2 \\ 0 \\ -2 \end{pmatrix}, \qquad \text{and} \qquad \boldsymbol{v}_3 = \begin{pmatrix} 3 \\ -3 \\ 3 \end{pmatrix}.$$

Carry out the Gram-Schmidt procedure as follows:

$$\begin{aligned}
\boldsymbol{w}_1 &= \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}, \\
\boldsymbol{w}_2 &= \boldsymbol{v}_2 - \frac{\boldsymbol{v}_2^\top \boldsymbol{w}_1}{\boldsymbol{w}_1^\top \boldsymbol{w}_1} \boldsymbol{w}_1 \\
&= \begin{pmatrix} 2 \\ 0 \\ -2 \end{pmatrix} - \frac{2}{2} \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} 1 \\ 1 \\ -2 \end{pmatrix}, \\
\boldsymbol{w}_3 &= \begin{pmatrix} 3 \\ -3 \\ 3 \end{pmatrix} - \frac{6}{2} \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix} - \frac{-6}{6} \begin{pmatrix} 1 \\ 1 \\ -2 \end{pmatrix} \\
&= \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.
\end{aligned}$$

Hence the orthonormal basis consists of vectors

$$\hat{\boldsymbol{u}}_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}, \qquad \hat{\boldsymbol{u}}_2 = \frac{1}{\sqrt{6}} \begin{pmatrix} 1 \\ 1 \\ -2 \end{pmatrix}, \qquad \text{and} \qquad \hat{\boldsymbol{u}}_3 = \frac{1}{\sqrt{3}} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

4.2 Generating the QR Factorization

To obtain the QR factorization, we first use Gram-Schmidt to orthogonalize the columns of A. The resulting orthonormal vectors constitute the columns of Q, that is, $Q = (\hat{\boldsymbol{u}}_1, \hat{\boldsymbol{u}}_2, \ldots, \hat{\boldsymbol{u}}_n)$. The matrix R is formed by keeping track of the Gram-Schmidt operations. Then, R expresses the columns of A as linear combinations of the columns of Q.

More specifically, we rewrite (1) and (2) into the following:

$$\boldsymbol{v}_j = \sum_{i=1}^{j-1} s_{ij}\boldsymbol{w}_i + \boldsymbol{w}_j, \qquad j = 1, \ldots, n,$$

where $s_{ij} = (\boldsymbol{v}_j^\top \boldsymbol{w}_i)/(\boldsymbol{w}_i^\top \boldsymbol{w}_i)$, $1 \leq i \leq j-1$, have already been calculated by the Gram-Schmidt procedure. The above equations are further rewritten as

$$\begin{aligned} \boldsymbol{v}_j &= \sum_{i=1}^{j-1} s_{ij}\|\boldsymbol{w}_i\|\hat{\boldsymbol{u}}_i + \|\boldsymbol{w}_j\|\hat{\boldsymbol{u}}_j \\ &= \sum_{i=1}^{n} r_{ij}\hat{\boldsymbol{u}}_i, \end{aligned}$$

where

$$r_{ij} = \left(\begin{array}{ll} s_{ij}\|\boldsymbol{w}_i\| & \text{if } 1 \leq i < j, \\ \|\boldsymbol{w}_j\| & \text{if } i = j, \\ 0 & \text{if } i > j. \end{array} \right.$$

From $A = QR$ we see that $R = (r_{ij})$.

Example 6. In the last example, we notice that

$$\begin{aligned} \boldsymbol{v}_1 &= \boldsymbol{w}_1 &&= \sqrt{2}\hat{\boldsymbol{u}}_1, \\ \boldsymbol{v}_2 &= \boldsymbol{w}_1 + \boldsymbol{w}_2 &&= \sqrt{2}\hat{\boldsymbol{u}}_1 + \sqrt{6}\hat{\boldsymbol{u}}_2, \\ \boldsymbol{v}_3 &= 3\boldsymbol{w}_1 - \boldsymbol{w}_2 + \boldsymbol{w}_3 &&= 3\sqrt{2}\hat{\boldsymbol{u}}_1 - \sqrt{6}\hat{\boldsymbol{u}}_2 + \sqrt{3}\hat{\boldsymbol{u}}_3. \end{aligned}$$

So the QR decomposition is given by

$$\begin{pmatrix} 1 & 2 & 3 \\ -1 & 0 & -3 \\ 0 & -2 & 3 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{3}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{3}} \\ 0 & -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{3}} \end{pmatrix} \begin{pmatrix} \sqrt{2} & \sqrt{2} & 3\sqrt{2} \\ 0 & \sqrt{6} & -\sqrt{6} \\ 0 & 0 & \sqrt{3} \end{pmatrix}.$$

References

[1] M. Erdmann. Lecture notes for *16-811 Mathematical Fundamentals for Robotics*. The Robotics Institute, Carnegie Mellon University, 1998.

[2] G. Golub and C. F. Van Loan. *Matrix Computations*, 3rd edition. The Johns Hopkins University Press, Baltimore, 1996.

[3] G. Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 1993.

[4] W. H. Press, *et al.* *Numerical Recipes in C++: The Art of Scientific Computing*. Cambridge University Press, 2nd edition, 2002.