Solution of Linear Equations
(Com S 477/577 Notes)
Yan-Bin Jia
Sep 7, 2017

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:

\[
B_1 = \{e_1, e_2, \ldots, e_n\}, \quad \text{for } \mathbb{R}^n,
\]
\[
B_2 = \{d_1, d_2, \ldots, d_m\}, \quad \text{for } \mathbb{R}^m,
\]

we can represent \( f \) by an \( m \times n \) matrix \( A \) such that \( f(x) = Ax \) for any \( x \in \mathbb{R}^n \). Note that the matrix \( A \) depends on the choices of the bases \( B_1 \) and \( B_2 \).

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 \( Ax = 0 \), where \( x \in \mathbb{R}^n \).

**Theorem 1 (Fundamental Theorem of Linear Algebra)** 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,

\[
\mathbb{R}^n = \text{row}(A) \bigoplus \text{null}(A),
\]

\[
n = \dim(\text{row}(A)) + \dim(\text{null}(A)).
\]

Consider the system of equations

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

\[
Ax = A(x_r + x_n) = Ax_r + Ax_n = Ax_r.
\]

The null space goes to zero, \( Ax_n = 0 \), while the row space component goes to the column space, \( Ax_r = Ax \).

If \( A \) is an \( n \times n \) square matrix, we say that \( A \) is singular

iff \( \det(A) = 0 \)
iff \( \text{rank}(A) < n \)
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 \( U = L^T \).

4. If \( A \) is symmetric and positive definite, then \( U = L^T \) and the diagonal entries of \( D \) are strictly positive.
Example 3.

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

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

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

\[ Ax = b. \]

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

\[ \text{LDU} \, x = \text{P} \, b. \]

We then first solve

\[ Ly = P \, b \]

for the vector \( y \) and then solve

\[ U \, x = D^{-1} \, y, \]

for the vector \( x \). 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,

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

The algorithm solves for \( L \) and \( U \) simultaneously and column by column.\(^1\) At the \( j \)th outer iteration step, it generates column \( j \) of \( U \) and then column \( j \) of \( L \).

\[
\text{for } j = 1, 2, \ldots, n \text{ do}
\]

\[
\text{for } i = 1, 2, \ldots, j \text{ do}
\]

\[
u_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj}
\]

\[
\text{for } i = j + 1, j + 2, \ldots, n \text{ do}
\]

\[
l_{ij} = \frac{1}{u_{jj}} \left( a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj} \right)
\]

\(^1\)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 $\alpha$’s and $\beta$’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,

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

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

**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 $i = 1, \ldots, n - 1$. 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 $a_{ij} - \sum_{k=1}^{j-1} a_{ik}u_{kj}$, $i = j, \ldots, n$ 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

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

4
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 \( u_{11} \)?

- If row 1: \( 1 \cdot u_{11} = 2 \rightarrow u_{11} = 2 \),
- If row 2: \( 1 \cdot u_{11} = 4 \rightarrow u_{11} = 4 \),
- If row 3: \( 1 \cdot u_{11} = 9 \rightarrow u_{11} = 9 \),
- If row 2: \( 1 \cdot u_{11} = 5 \rightarrow u_{11} = 5 \).

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 \( l_{21} = \frac{4}{9} \), \( l_{31} = \frac{2}{9} \), and \( l_{41} = \frac{5}{9} \).

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

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

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

- If row 2: \( \frac{4}{9} \cdot (-6) + u_{22} = 8 \rightarrow u_{22} = \frac{32}{3} \),
- If row 3: \( \frac{2}{9} \cdot (-6) + u_{22} = -7 \rightarrow u_{22} = -\frac{17}{3} \),
- If row 4: \( \frac{5}{9} \cdot (-6) + u_{22} = 1 \rightarrow u_{22} = \frac{13}{3} \).

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 \( L \):

\[
l_{32} = -\frac{17}{32} \quad \text{and} \quad l_{42} = \frac{13}{32}.
\]

By this time, we have

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

### 3 Factorization Based on Eigenvalues

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

\[ A = SAS^{-1}, \]
where $\Lambda$ 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

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

with $\lambda_i$ 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 $m \geq n$). 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, $Q^TQ = I$. 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 $Ax = b$ in the least-squares sense. The least-squares solution $\bar{x}$ is given by $\bar{x} = (A^TA)^{-1}A^Tb$, assuming that the columns of $A$ are independent. But $Q^TQ = I$, so

$$ \bar{x} = (R^TQ^TQR)^{-1}R^TQ^Tb $$

$$ = (R^TR)^{-1}R^TQ^Tb $$

$$ = R^{-1}(R^T)^{-1}R^TQ^Tb $$

$$ = R^{-1}Q^Tb. $$

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

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

#### 4.1 The Gram-Schmidt Procedure

Given $n$ linearly independent vectors $v_1, \ldots, v_n$, the Gram-Schmidt procedure constructs $n$ orthonormal vectors $\hat{u}_1, \ldots, \hat{u}_n$ such that these two sets of vectors span the same space. First, it constructs $n$ orthogonal vectors $w_1, \ldots, w_n$ below:

$$ w_1 = v_1, \quad j = 1, \ldots, n, $$

$$ w_j = v_j - \sum_{i=1}^{j-1} \frac{v_j^T w_i}{w_i^T w_i} w_i, \quad j = 2, \ldots, n. $$
Essentially, we subtract from every new vector \( v_j \) its projection in the directions \( w_1/\|w_1\|, \ldots, w_{i-1}/\|w_{i-1}\| \) that are already set. Next, we simply perform a normalization by letting

\[
\hat{u}_j = \frac{w_j}{\|w_j\|}, \quad j = 1, \ldots, n.
\]

**Example 5.** Consider three vectors:

\[ v_1 = \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}, \quad v_2 = \begin{pmatrix} 2 \\ 0 \\ -2 \end{pmatrix}, \quad \text{and} \quad v_3 = \begin{pmatrix} 3 \\ -3 \\ 3 \end{pmatrix}. \]

Carry out the Gram-Schmidt procedure as follows:

\[
w_1 = \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix},
\]

\[
w_2 = v_2 - \frac{v_2^T w_1}{w_1^T w_1} w_1
\]

\[ = \begin{pmatrix} 2 \\ 0 \\ -2 \end{pmatrix} - \frac{2}{2} \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ -2 \end{pmatrix},
\]

\[
w_3 = \begin{pmatrix} 3 \\ -3 \\ 3 \end{pmatrix} - \frac{6}{2} \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix} - \frac{6}{6} \begin{pmatrix} 1 \\ 1 \\ -2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.
\]

Hence the orthonormal basis consists of vectors

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

### 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, \( Q = (\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_n) \). 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:

\[ v_j = \sum_{i=1}^{j-1} s_{ij} w_i + w_j, \quad j = 1, \ldots, n, \]

where \( s_{ij} = \frac{(v_j^T w_i)}{(w_i^T w_i)}, \) \( 1 \leq i \leq j - 1, \) have already been calculated by the Gram-Schmidt procedure. The above equations are further rewritten as

\[ v_j = \sum_{i=1}^{j-1} s_{ij} \|w_i\| \hat{u}_i + \|w_j\| \hat{u}_j = \sum_{i=1}^{n} r_{ij} \hat{u}_i, \]

where

\[ r_{ij} = \begin{cases} s_{ij} \|w_i\| & \text{if } 1 \leq i < j, \\ \|w_j\| & \text{if } i = j, \\ 0 & \text{if } i > j. \end{cases} \]

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

**Example 6.** In the last example, we notice that

\[
\begin{align*}
  v_1 &= w_1 = \sqrt{2} \hat{u}_1, \\
  v_2 &= w_1 + w_2 = \sqrt{2} \hat{u}_1 + \sqrt{6} \hat{u}_2, \\
  v_3 &= 3w_1 - w_2 + w_3 = 3\sqrt{2} \hat{u}_1 - \sqrt{6} \hat{u}_2 + \sqrt{3} \hat{u}_3.
\end{align*}
\]

So the QR decomposition is given by

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

**References**


