Lecture 2: Gauss elimination and LU factorization

Michael S. Floater

August 29, 2024

These notes are based on Sections 2.1–2.3 of the book.

1 Simultaneous linear equations

We are interested in solving the linear system of n equations

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & & & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix},$$

in the n unknowns $x_1, \ldots, x_n$. We can express this in matrix and vector notation as

$$A\mathbf{x} = \mathbf{b}. \tag{1}$$

where $A \in \mathbb{R}^{n,n}$ is the matrix of coefficients, $\mathbf{b} \in \mathbb{R}^n$ is a given vector, the right-hand side, and $\mathbf{x} \in \mathbb{R}^n$ is the vector of unknowns to be determined. The equations have a unique solution if and only if A is non-singular, i.e., the inverse A^{-1} exists. The solution is then $\mathbf{x} = A^{-1}\mathbf{b}$. We do not normally calculate A^{-1} explicitly if our only goal is to compute $\mathbf{x}$. Calculating A^{-1} might lead to unnecessary loss of accuracy.

2 Forward and back substitution

Equation (1) is easy to solve if A is either lower triangular or upper triangular, i.e.,

$$\begin{bmatrix} a_{1,1} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ a_{n-1,1} & \cdots & a_{n-1,n-1} & 0 \\ a_{n,1} & \cdots & \cdots & a_{n,n} \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} a_{1,1} & 0 & \cdots & a_{1,n} \\ 0 & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a_{n,n} \end{bmatrix},$$

respectively. In both cases, its determinant is the product of its diagonal elements:

$$\det(A) = \prod_{k=1}^{n} a_{k,k}.$$

Therefore, A is non-singular if and only if all its diagonal elements are non-zero.

Suppose A is lower triangular. We then obtain the solution $\mathbf{x}$ by *forward substitution*. The first equation is

$$a_{1,1}x_1 = b_1.$$

We can solve this for x_1 to obtain

$$x_1 = \frac{b_1}{a_{1,1}},$$

which is valid since $a_{1,1} \neq 0$. The second equation is

$$a_{2,1}x_1 + a_{2,2}x_2 = b_2.$$

Using the value of x_1 already computed, we can solve this equation for x_2:

$$x_2 = \frac{b_2 - a_{2,1}x_1}{a_{2,2}},$$

which is again valid since $a_{2,2} \neq 0$. We continue in this way. To compute each x_i we use the values of $x_1, \ldots, x_{i-1}$ already computed and find that

$$x_i = \frac{b_i - \sum_{j=1}^{i-1} a_{i,j}x_j}{a_{i,i}}, \qquad i = 1, 2, \ldots, n.$$

Conversely, if A is upper triangular, we obtain the solution by *back substitution*. The last equation is

$$a_{n,n}x_n = b_n,$$

from which we obtain the value of the unknown x_n, by

$$x_n = \frac{b_n}{a_{n,n}}.$$

The second to last equation is

$$a_{n-1,n-1}x_{n-1} + a_{n-1,n}x_n = b_{n-1},$$

and using the value of x_n already computed we find x_{n-1}:

$$x_{n-1} = \frac{b_{n-1} - a_{n-1,n}x_n}{a_{n-1,n-1}}.$$

Continuing in this way, the whole algorithm is

$$x_i = \frac{b_i - \sum_{j=i+1}^{n} a_{i,j}x_j}{a_{i,i}}, \qquad i = n, n-1, \ldots, 1.$$

3 Gauss elimination and pivoting

Given a set of simultaneous equations $A\mathbf{x} = \mathbf{b}$, the solution $\mathbf{x}$ is unchanged if any of the following operations is performed:

1. Multiplication of an equation by a non-zero constant.
2. Addition of a multiple of one equation to another.
3. Interchange of two equations.

The same operations have to be performed on both sides of the equation. These operations are used to convert the system of equations to the trivial case, i.e., upper triangular form. There are different ways to go about this. The usual strategy is a *pivotal strategy* which tries to avoid the accumulation of errors and for some matrices is crucially important.

A *pivot* entry is required to be non-zero, but preferably far from zero. Finding this element is called *pivoting*. Once the pivot element is found,

rows (and possibly columns) are interchanged to bring the pivot element onto the diagonal. The swapping of rows is equivalent to multiplying A by a permutation matrix. In practice, the matrix elements are rarely moved, since this would cost too much time. Instead the algorithms keep track of the permutations. Pivoting increases the overall computational cost of the algorithm. However, sometimes pivoting is necessary for the algorithm to work at all, at other times it increases the numerical stability. We illustrate this with two examples.

3.1 Example 1

Consider the three simultaneous equations

$$\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}.$$

We want to convert the matrix into upper triangular form, and then find $\mathbf{x}$ by back substitution. The first equation cannot form the first row of the upper triangle because its first coefficient is zero. So we need to 'pivot', i.e., swap the first equation with one of the other equations. For example, we can swap it with the second equation, so that

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 4 \end{bmatrix}.$$

Now, to obtains zeros in the first column below the diagonal element we subtract row 1 from row 3, so that

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix}.$$

We now move to the second column. The diagonal element in the second column is non-zero and so we do not need to pivot. To obtain a zero beneath the diagonal we subtract row 2 from row 3, to arrive at

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}.$$

We have now reached an upper triangular matrix and we find $\mathbf{x}$ be back substitution, which gives the solution $x_1 = 5/2$, $x_2 = 3/2$, and $x_3 = -1/2$.

3.2 Example 2

This example shows that not only do zero coefficients cause problems, but small coefficients can cause numerical problems too:

$$\begin{bmatrix} 0.0002 & 1.044 \\ 0.2302 & -1.624 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1.046 \\ 0.678 \end{bmatrix}.$$

The exact solution is $x_1 = 10$, $x_2 = 1$. If we assume we use four digits of accuracy and we do not pivot then we multiply the first equation by $0.2302/0.0002 = 1151$ and subtract from the second. Then the last equation becomes

$$-1204x_2 = -1203,$$

which gives the solution $x_2 = 0.9992$. Using this number to solve for x_1 in the first equation gives the answer $x_1 = 14.18$, which is far from the true solution.

In general, suppose we solve a set of n equations by Gauss elimination. At the first step, one of the n equations is chosen as the pivot and moved to the first row. Then zeros are introduced in the first column below the first row. This leaves at the second step $n-1$ equations to be transformed. At the start of the k-th step there are $n - k + 1$ equations remaining. Suppose the current matrix has elements $\tilde{a}_{ij}$, $i, j = 1, \ldots, n$. We select a pivot element as follows.

In *partial pivoting*, the pivot is the largest element in absolute value of the k-th column between rows k and n, i.e., we find a row $i_* \in \{k, \ldots, n\}$ such that

$$|\tilde{a}_{i_*,k}| = \max_{i=k,\ldots,n} |\tilde{a}_{i,k}|.$$

We then swap rows k and i_*.

In *total pivoting*, we find a pair (i_*, j_*), where $i, j \in \{k, \ldots, n\}$, such that

$$|\tilde{a}_{i_*,j_*}| = \max_{i,j=k,\ldots,n} |\tilde{a}_{i,j}|.$$

We then swap rows k and i_* and swap columns k and j_*. When we swap columns k and j_* we must also swap the variables x_k and x_{j_*}.

Total pivoting is more expensive, but for certain systems may be required for acceptable accuracy.

4 LU Factorization

Instead of using Gauss elimination to solve (1), we can first factorize A into a lower triangular matrix L (i.e., $l_{i,j} = 0$ for $i < j$) and an upper triangular matrix U (i.e., $u_{i,j} = 0$ for $i > j$), that is, $A = LU$. Now $LU\mathbf{x} = \mathbf{b}$, which we can solve by forward and back substitution. First we find $\mathbf{y}$ such that $L\mathbf{y} = \mathbf{b}$ by forward substitution. Then we find $\mathbf{x}$ such that $U\mathbf{x} = \mathbf{y}$ by back substitution. Other applications of the LU factorization are

1. Calculation of the determinant:
$$\det A = (\det L)(\det U) = \Big(\prod_{k=1}^{n} l_{k,k}\Big)\Big(\prod_{k=1}^{n} u_{k,k}\Big).$$

2. Non-singularity testing: $A = LU$ is non-singular if and only if all the diagonal elements of L and U are non-zero.

3. Calculating the inverse of A: the inverse of a triangular matrix is easy to compute, and we have $A^{-1} = U^{-1}L^{-1}$.

To find the factorization $A = LU$, let $\mathbf{l}_1, \ldots, \mathbf{l}_n$ be the columns of L and let $\mathbf{u}_1^T, \ldots, \mathbf{u}_n^T$ be the rows of U:

$$L = \begin{bmatrix} \mathbf{l}_1 & \mathbf{l}_2 & \cdots & \mathbf{l}_n \end{bmatrix}, \qquad U = \begin{bmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \\ \vdots \\ \mathbf{u}_n^T \end{bmatrix}.$$

Then the element $a_{i,j}$ of A is

$$a_{i,j} = \sum_{k=1}^{n} l_{i,k}u_{k,j} = \sum_{k=1}^{n} (\mathbf{l}_k)_i (\mathbf{u}_k^T)_j = \sum_{k=1}^{n} (\mathbf{l}_k\mathbf{u}_k^T)_{i,j},$$

and therefore,

$$A = \sum_{k=1}^{n} \mathbf{l}_k\mathbf{u}_k^T.$$

(This is an example of block multiplication.) Assume that A is nonsingular and that the factorization exists. Hence the diagonal elements of L are non-zero. Since $\mathbf{l}_k\mathbf{u}_k^T$ stays the same if $\mathbf{l}_k$ is replaced by $\alpha\mathbf{l}_k$ and $\mathbf{u}_k$ is replaced by

$\alpha^{-1}\mathbf{u}_k$ for $\alpha \neq 0$, we can assume that all diagonal elements of L are equal to 1.

Since the first $k-1$ components of $\mathbf{l}_k$ and $\mathbf{u}_k$ are zero, the matrix $\mathbf{l}_k\mathbf{u}_k^T$ has zeros in the first $k-1$ rows and columns. It follows that $\mathbf{u}_1^T$ is the first row of A and $\mathbf{l}_1$ is the first column of A divided by $a_{1,1}$ so that $l_{1,1} = 1$.

Having found $\mathbf{l}_1$ and $\mathbf{u}_1$, we form the matrix

$$A_1 = A - \mathbf{l}_1\mathbf{u}_1^T = \sum_{k=2}^{n} \mathbf{l}_k\mathbf{u}_k^T.$$

The first row and column of A_1 are zero and it follows that $\mathbf{u}_2^T$ is the second row of A_1 and $\mathbf{l}_2$ is the second column of A_1 scaled so that $l_{2,2} = 1$.

We continue in this way. We initialize the algorithm by letting $A_0 = A$. For all $k = 1, \ldots, n$ set $\mathbf{u}_k^T$ to be the k-th row of A_{k-1} and $\mathbf{l}_k$ to be the k-th column of A_{k-1}, scaled so that $l_{k,k} = 1$. Then calculate

$$A_k = A_{k-1} - \mathbf{l}_k\mathbf{u}_k^T. \tag{2}$$

As an example, we find the LU factorization of

$$A = \begin{bmatrix} 3 & 4 \\ 5 & 6 \end{bmatrix}.$$

In the first step we use the first row and column of A:

$$\mathbf{u}_1^T = \begin{bmatrix} 3 & 4 \end{bmatrix},$$

and

$$\mathbf{l}_1 = \frac{1}{3}\begin{bmatrix} 3 \\ 5 \end{bmatrix} = \begin{bmatrix} 1 \\ 5/3 \end{bmatrix}.$$

Then we find

$$\mathbf{l}_1\mathbf{u}_1^T = \begin{bmatrix} 3 & 4 \\ 5 & 20/3 \end{bmatrix},$$

and we let

$$A_1 = A - \mathbf{l}_1\mathbf{u}_1^T = \begin{bmatrix} 0 & 0 \\ 0 & -2/3 \end{bmatrix}.$$

In the second step we use the second row and column of A_1:

$$\mathbf{u}_2^T = \begin{bmatrix} 0 & -2/3 \end{bmatrix},$$

and

$$\mathbf{l}_2 = \frac{1}{-2/3}\begin{bmatrix} 0 \\ -2/3 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

This completes the factorization:

$$A = \begin{bmatrix} \mathbf{l}_1 & \mathbf{l}_2 \end{bmatrix} \begin{bmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 5/3 & 1 \end{bmatrix} \begin{bmatrix} 3 & 4 \\ 0 & -2/3 \end{bmatrix}.$$

All elements of the first k rows and columns of A_k are zero, and so we can (if we want to) use the storage of the orginal A to accumulate the columns of L and the rows of U. The full LU factorization requires $O(n^3)$ operations.

If A is sparse we would like L and U to be sparse too, but this is not always the case: we can get 'fill-in'. However, we have

Theorem 1 *Let $A = LU$ without pivoting. Then all leading zeros in the rows of A to the left of the diagonal are inherited by L. Similarly, leading zeros in the columns of A above the diagonal are inherited by U.*

We see this in the example:

$$\begin{bmatrix} 2 & 0 & 1 & 0 & 0 \\ 0 & 3 & 2 & 0 & 0 \\ 1 & 2 & -3 & 0 & 1 \\ 0 & 0 & 0 & 3 & 1 \\ 0 & 0 & 1 & 1 & -3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{2}{3} & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{6}{29} & \frac{1}{3} & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 1 & 0 & 0 \\ 0 & 3 & 2 & 0 & 0 \\ 0 & 0 & -\frac{29}{6} & 0 & 1 \\ 0 & 0 & 0 & 3 & 1 \\ 0 & 0 & 0 & 0 & -\frac{272}{87} \end{bmatrix}.$$

So to reduce fill-in one might consider swapping rows and columns of A to get as many leading zeros as possible before performing LU.

5 Exercises

Exercise 2.1 *Implement back substitution.*

Exercise 2.3 *An LU factorization is calculated of an $n \times n$ matrix A, where L has ones on the diagonal and the moduli of all off-diagonal elements do not exceed 1. Let α be the largest moduli of the elements of A. Prove by induction that elements of U satisfy $|u_{i,j}| \leq 2^{i-1}\alpha$. Construct 2×2 and 3×3 non-zero matrices A that give $|u_{2,2}| = 2\alpha$ and $|u_{3,3}| = 4\alpha$ respectively.*

Exercise 2.5 *Let A be a real nonsingular $n \times n$ matrix that has a factorization $A = LU$, where L is lower triangular with ones on its diagonal and U is upper triangular. Show that for $k = 1, \ldots, n$, the first k rows of U span the same subspace as the first k rows of A. Similarly, show that the first k columns of L span the same subspace as the first k columns of A.*