**St. Francis Xavier University**
**Department of Computer Science**

**CSCI 554: Matrix Computation**
**Lecture 3: Gaussian Elimination and LU Decomposition**
**Winter 2022**

# 1 Gaussian Elimination

Let's continue our study of systems of linear equations that we initiated in the previous lecture by considering methods of solving these systems. In the previous lecture, we focused on special systems such as triangular systems, and developed algorithms to solve these special systems. We remarked then that it's relatively painless to convert a general matrix to a special form, but we never explicitly described that conversion process.

If we have a system of linear equations $Ax = b$, then converting the matrix $A$ to be an upper triangular matrix $U$ would allow us to use the method of backward substitution to solve the equivalent system $Ux = y$. To ensure that our two systems remain equivalent throughout this conversion process, we use three *elementary row operations*:

- Adding a multiple of one row to another row;
- Interchanging the order of two rows; and
- Multiplying a row by some nonzero constant.

Restricting ourselves to these three operations, we can perform the conversion of $A$ while ensuring the overall system is not modified or changed. In fact, the following property holds in this case.

**Proposition 1.** *Given a system of linear equations $Ax = b$, if another system $\hat{A}x = \hat{b}$ is obtainable from $Ax = b$ by one of the three elementary row operations, then the two systems have the same solutions.*

Recall also that a system of linear equations $Ax = b$ has a unique solution if and only if the matrix $A$ is invertible. Fortunately, our elementary row operations do not affect invertibility, and so the following additional property holds when we use these operations.

**Proposition 2.** *Given a matrix $A$, if another matrix $\hat{A}$ is obtainable from $A$ by one of the three elementary row operations, then $\hat{A}$ is invertible if and only if $A$ is invertible.*

Thus, starting with an invertible matrix $A$, we can use our elementary row operations to obtain an upper triangular matrix $U$ that is also invertible, thus guaranteeing that backward substitution gives us a unique solution.

## 1.1 Gaussian Elimination without Pivoting

For the purposes of our study, we will restrict ourselves to using only the first elementary row operation—that is, adding a multiple of one row to another row—to perform the conversion of $A$ to an upper triangular matrix, which is known as *Gaussian elimination without pivoting* or sometimes just *Gaussian elimination*.

In what follows, we will rely on an assumption involving the *pth leading principal submatrices* of our matrix $A$. The $p$th leading principal submatrix of $A$, denoted $A_p$, is the submatrix consisting of the first $p$ rows and columns of $A$. (You could view this as the upper-left square of $A$ for varying values of $p$.)

**Example 3.** Consider the matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}.$$

This matrix has three leading principal submatrices: $A_1 = \begin{bmatrix} a_{11} \end{bmatrix}$, $A_2 = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$, and $A_3 = A$.

Our assumption in this section is to take each leading principal matrix $A_p$ to be invertible, for all $1 \leq p \leq n$. This implies that $A_n = A$ is itself invertible, so we are guaranteed to have a unique solution to the system $Ax = b$.

For convenience, since we will be working with matrices directly instead of their corresponding linear equations, we will sometimes use an *augmented matrix* to represent both $A$ and $b$ in one matrix $\begin{bmatrix} A \mid b \end{bmatrix}$, where $b$ is "concatenated" to the right edge of $A$ and separated by a delimiting line.

**Example 4.** The system of linear equations

$$\begin{aligned} 2x_1 - 3x_2 &= 18 \\ 5x_1 + 2x_2 &= 7 \end{aligned}$$

is represented by the augmented matrix

$$\begin{bmatrix} A \mid b \end{bmatrix} = \left[\begin{array}{cc|c} 2 & -3 & 18 \\ 5 & 2 & 7 \end{array}\right].$$

In order to convert our matrix $A$ to an upper triangular matrix, we must zero out all entries of $A$ below the diagonal. We can do this in a total of $n-1$ iterations through our augmented matrix: one iteration for each row of $\begin{bmatrix} A \mid b \end{bmatrix}$. Starting with an augmented matrix

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & & a_{2n} & b_2 \\ \vdots & & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{array}\right],$$

we can zero out the entries of column 1 below $a_{11}$ by calculating $m_{i1} = a_{i1}/a_{11}$ for all $2 \leq i \leq n$.[1] We then adjust the second through $n$th rows of $\begin{bmatrix} A \mid b \end{bmatrix}$ by calculating

$$\begin{aligned} a_{ij}^{(1)} &= a_{ij} - m_{i1}a_{1j} \text{ for all } 2 \leq i \leq n \text{ and } 2 \leq j \leq n; \text{ and} \\ b_i^{(1)} &= b_i - m_{i1}b_1 \text{ for all } 2 \leq i \leq n. \end{aligned}$$

This gives us a modified augmented matrix of the form

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ 0 & a_{22}^{(1)} & & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & & \ddots & \vdots & \vdots \\ 0 & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} & b_n^{(1)} \end{array}\right].$$

We then repeat this process by zeroing out the entries of column 2 below $a_{22}^{(1)}$ and adjusting the third through $n$th rows of $\begin{bmatrix} A \mid b \end{bmatrix}$, and so on. On the $k$th iteration, we will calculate the values

$$\begin{aligned} m_{ik} &= a_{ik}^{(k-1)}/a_{kk}^{(k-1)} \text{ for all } (k+1) \leq i \leq n; \\ a_{ij}^{(k)} &= a_{ij}^{(k-1)} - m_{ik}a_{kj}^{(k-1)} \text{ for all } (k+1) \leq i \leq n \text{ and } (k+1) \leq j \leq n; \text{ and} \\ b_i^{(k)} &= b_i^{(k-1)} - m_{ik}b_k^{(k-1)} \text{ for all } (k+1) \leq i \leq n. \end{aligned}$$

[1] Note that, in order to calculate $m_{i1}$, we must have that $a_{11} \neq 0$. Fortunately, this is guaranteed by our assumption that $A_p$ is invertible for all $p$.

Eventually, after executing each iteration and adjusting each row, we arrive at an augmented matrix of the form

$$\left[\begin{array}{ccccc|c} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & b_1 \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & & a_{2n}^{(1)} & b_2^{(1)} \\ 0 & 0 & a_{33}^{(2)} & & a_{3n}^{(2)} & b_3^{(2)} \\ \vdots & & & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn}^{(n-1)} & b_n^{(n-1)} \end{array}\right]$$

where the entries $a_{ij}$ form an upper triangular matrix.

However, from a space efficiency aspect, this is not a particularly good augmented matrix: the entire lower triangular portion of the matrix contains useless zero entries! As a small enhancement, instead of leaving the zero entries in each column $j$ as they are, we will store the multipliers $m_{ij}$ in their respective row $i$. Our augmented matrix will then be of the form

$$\left[\begin{array}{ccccc|c} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & b_1 \\ m_{21} & a_{22}^{(1)} & a_{23}^{(1)} & & a_{2n}^{(1)} & b_2^{(1)} \\ m_{31} & m_{32} & a_{33}^{(2)} & & a_{3n}^{(2)} & b_3^{(2)} \\ \vdots & & & \ddots & \vdots & \vdots \\ m_{n1} & m_{n2} & m_{n3} & \cdots & a_{nn}^{(n-1)} & b_n^{(n-1)} \end{array}\right].$$

Why would we do this? We'll find that those multipliers $m_{ij}$ will come in handy later on, so it's prudent of us to hold on to them for now in that unused space.

To express our method for Gaussian elimination without pivoting in pseudocode, we will present it in two parts: one part to handle entries of $A$, and another part to handle entries of $b$ separately. We separate these two parts because the matrix $A$ and the vector $b$ are likely to be stored separately in a computer's memory, rather than as a single augmented matrix. Regardless, it's a simple matter to combine the two parts if we are indeed working with an augmented matrix.

First, we handle the matrix $A$. Remember that our method automatically stores the multipliers $m_{ij}$ in the place of each zero entry below the diagonal, so we must assign those multipliers to the appropriate indices.

**Algorithm 1:** Gaussian elimination without pivoting—matrix $A$

```
for 1 ≤ k ≤ (n − 1) do
    for (k + 1) ≤ i ≤ n do
        a_ik ← a_ik / a_kk                ▷ store the multiplier m_ik (here referenced by a)
        for (k + 1) ≤ j ≤ n do
            a_ij ← a_ij − a_ik a_kj       ▷ recall that a_ik = m_ik
return A
```

We then handle the vector $b$ in a similar manner, using the multiplier entries that we stored previously in $A$.

**Algorithm 1:** Gaussian elimination without pivoting—vector $b$

```
for 1 ≤ k ≤ (n − 1) do
    for (k + 1) ≤ i ≤ n do
        b_i ← b_i − a_ik b_k              ▷ recall that a_ik = m_ik
return b
```

In our analysis, we will consider the matrix $A$ and vector $b$ algorithms simultaneously, as if the two parts were combined into a single algorithm. Note that this is possible because the bounds of the two outer for loops in both algorithms match. Therefore, we can consolidate the instruction calculating each value $b_i$ into the body of the second for loop of the algorithm calculating the matrix $A$.

Observe that, within the two outer for loops, we perform one division (to compute $a_{ik}$), one multiplication (to compute $a_{ik}b_k$), and one addition (to add $a_{ik}b_k$ to $b_i$). We also perform additional operations within the third nested for loop: one multiplication (to compute $a_{ik}a_{kj}$) and one addition (to add $a_{ik}a_{kj}$ to $a_{ij}$).

Since the nested for loops and the varying bounds on each for loop make the general analysis a bit complex, let's begin by focusing on the number of operations performed during the first iteration; that is, when $k = 1$. On this iteration, we perform a total of $(n-1)$ divisions, $(n-1)^2+(n-1)$ multiplications, and $(n-1)^2+(n-1)$ additions. Altogether, this gives us a total of $2(n-1)^2 + 3n - 3 = (2n+1)(n-1)$ operations performed in the first iteration, which is approximately $2n^2$ when we disregard lower-order terms.

On the second iteration, we perform all the same steps, but on a matrix with one fewer row and one fewer column. Thus, the total number of operations performed on this iteration is approximately $2(n-1)^2$. The same observation applies to subsequent iterations until the algorithm halts. Thus, the overall number of operations performed by the algorithm from start to finish is approximately

$$2n^2 + 2(n-1)^2 + 2(n-2)^2 + \cdots = 2\sum_{k=1}^{n} k^2$$

and, with a little bit of work, we can approximate this sum as $2\sum_{k=1}^{n} k^2 \approx \frac{2}{3}n^3$. Therefore, using Gaussian elimination without pivoting requires $O(n^3)$ time, which outpaces the $O(n^2)$ backward substitution algorithm we would subsequently need to use to solve the system of linear equations.

## 1.2 (Re-)Using the Multipliers $m_{ij}$

Recall that, when we were performing Gaussian elimination on a matrix $A$, we observed that it would be wise to store the multipliers $m_{ij}$ that we computed at each step in the previously-useless lower triangular region of zeros. While it doesn't seem immediately obvious why we would hold on to these multipliers, doing so actually saves us a great deal of work if we need to solve more than one system of linear equations involving the matrix $A$.

Suppose that we have performed Gaussian elimination followed by backward substitution to solve the system $Ax = b$. Now, suppose we have a second system that we wish to solve, and this system is of the form $Ax = \hat{b}$; notice that the only change occurs on the right-hand side of the system. While we could simply perform Gaussian elimination followed by backward substitution once again to solve this new system, that would be rather naïve of us; the matrix $A$ is exactly the same, and so the process of performing Gaussian elimination on $A$ would give us the exact same result (including the multipliers).

Thus, instead of performing redundant computations, we reuse the multipliers $m_{ij}$ that we obtained from our previous computation and apply them directly to the vector $\hat{b}$. When we modified the vector $b$ from our original system, we calculated the values

$$b_i^{(k)} = b_i^{(k-1)} - m_{ik}b_k^{(k-1)} \text{ for all } (k+1) \leq i \leq n.$$

In doing so, we computed a vector

$$\begin{bmatrix} b_1 \\ b_2^{(1)} \\ b_3^{(2)} \\ \vdots \\ b_n^{(n-1)} \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = y,$$

and doing the same calculations for our new vector $\hat{b}$ would produce a vector $\hat{y}$ of the same form. This means that if we wanted to solve the system $Ax = \hat{b}$, the only real work we would need to do is in computing the vector $\hat{y}$: we can then use our upper triangular coefficient matrix $U$ that we obtained from our matrix $A$ to solve the new system $Ux = \hat{y}$.

In total, computing each of the entries $\hat{b}_i^{(k)}$ to obtain the vector $\hat{y}$ requires us to perform approximately $2n^2$ operations. Comparing this to the $O(n^3)$ operations for a complete Gaussian elimination without pivoting, it's clear that we enjoy quite a nice benefit by reusing our multipliers!

# 2 LU Decomposition

Let's revisit for a moment the expression we had for modifying the vector $b$ to $y$ and, specifically, for calculating individual entries $b_i^{(k)}$. Recall that this expression was

$$b_i^{(k)} = b_i^{(k-1)} - m_{ik} b_k^{(k-1)} \text{ for all } (k+1) \leq i \leq n.$$

Observe that the $k$th iteration of $b_i$ involves the term $b_k^{(k-1)}$, which is equal to the previously-calculated term $y_k$ in our vector $y$. Thus, we can rewrite the expression as

$$b_i^{(k)} = b_i^{(k-1)} - m_{ik} y_k \text{ for all } (k+1) \leq i \leq n.$$

Let's rearrange expressions of this form to obtain an expression for terms of $y$ directly. Observe that $y_1 = b_1$, so we move on to considering $y_2$, which we can calculate with the expression $y_2 = b_2^{(1)} = b_2 - m_{21} y_1$. Following this, we can calculate $y_3 = b_3^{(2)} = b_3^{(1)} - m_{32} y_2 = (b_3 - m_{31} y_1) - m_{32} y_2$. Generalizing this pattern for any $i$, we see that we're able to calculate all terms $y_i$ using the expression

$$y_i = b_i - \sum_{j=1}^{i-1} m_{ij} y_j \text{ for all } 1 \leq i \leq n.$$

But then, rearranging once again, we obtain the expression

$$\sum_{j=1}^{i-1} m_{ij} y_j + y_i = b_i \text{ for all } 1 \leq i \leq n,$$

which corresponds exactly to a system of linear equations modelling the matrix equation

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ m_{21} & 1 & 0 & & 0 \\ m_{31} & m_{32} & 1 & & 0 \\ \vdots & & & \ddots & \vdots \\ m_{n1} & m_{n2} & m_{n3} & \cdots & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}.$$

This leads to a very interesting observation: the vector $y$, which we obtained from $b$ via Gaussian elimination, is itself the solution to another system $Ly = b$, where $L$ is a lower triangular matrix containing our multipliers. Specifically, $L$ is said to be a *unit lower triangular matrix*, since the entries along the diagonal of $L$ are all ones.

Where does this leave us now? Combining all that we've learned in this lecture so far, we see that:

- we can reduce the system $Ax = b$ using Gaussian elimination to obtain the equivalent system $Ux = y$, where $U$ is an upper triangular matrix; and
- the vector $y$ is the solution to another system $Ly = b$, where $L$ is a unit lower triangular matrix.

Substituting one expression into the other, we get that $LUx = b = Ax$, so $A = LU$ for any vector $b$. We can therefore conclude that there exists a decomposition of our original matrix $A$ into two matrices $L$ and $U$, each of which has a special form. Moreover, we can obtain both of these matrices using our Gaussian elimination method; $U$ is taken directly from the upper triangular matrix obtained via Gaussian elimination, while $L$ is formed by adding our multipliers $m_{ij}$ to an identity matrix. Appropriately enough, we call this decomposition of $A$ into two matrices the *$LU$ decomposition* of $A$.

After performing the $LU$ decomposition to get a system of the form $LUx = b$, we can easily solve the system by first solving the system $Ly = b$ via forward substitution to obtain $y$, and then solving the system $Ux = y$ via backward substitution to obtain $x$.

While we have just demonstrated the existence of the matrices $L$ and $U$, we can go one step further by proving that these matrices are, in fact, unique for a given matrix $A$. This property is summarized in the following theorem.

**Theorem 5** (LU decomposition theorem)**.** *Let $A$ be an $n \times n$ matrix where all leading principal submatrices of $A$ are invertible. Then $A$ can be decomposed into a product $A = LU$ in exactly one way, where $L$ is a unit lower triangular matrix and $U$ is an upper triangular matrix.*

*Proof.* Let $A$ be a matrix as specified in the statement of the theorem. We have already proved the existence of the matrices $L$ and $U$. Thus, all that remains is for us to prove that $L$ and $U$ are unique.

Consider the matrix equation corresponding to the system $A = LU$:

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

We will show that each row and column of $L$ and $U$ are uniquely determined by an induction-style argument.

Observe that the first row of $L$ has only one nonzero entry: the 1 in the first column. Therefore, the first row of $L$ is uniquely determined. Multiplying the first row of $L$ by the $j$th column of $U$, we find that $a_{1j} = u_{1j}$, so the first row of $U$ is also uniquely determined. From this, we see also that the first column of $U$ is uniquely determined, since it contains only one nonzero entry: $u_{11}$. Finally, multiplying the $i$th row of $L$ by the first column of $U$, we find that $a_{i1} = l_{i1}u_{11}$ or, equivalently, $l_{i1} = a_{i1}/u_{11}$. Thus, the first column of $L$ is uniquely determined.[2]

Now, assume that we have uniquely determined the first $(k-1)$ columns of $L$ and the first $(k-1)$ rows of $U$. The $k$th row of $L$ is of the form

$$\begin{bmatrix} l_{k1} & l_{k2} & l_{k3} & \cdots & l_{k(k-1)} & 1 & 0 & \cdots & 0 \end{bmatrix},$$

and since each of the entries $l_{k1}$ through $l_{k(k-1)}$ occur in the first $(k-1)$ columns of $L$, they are each uniquely determined and so too is the $k$th row of $L$.

Multiplying the $k$th row of $L$ by the $j$th column of $U$, where $j \geq k$, we find that $a_{kj} = \sum_{m=1}^{k-1} l_{km}u_{mj} + u_{kj}$. All of the terms of $U$ in this expression, besides $u_{kj}$, occur in the first $(k-1)$ rows of $U$ and are therefore uniquely determined. Thus, $u_{kj} = a_{kj} - \sum_{m=1}^{k-1} l_{km}u_{mj}$ is uniquely determined for all $k \leq j \leq n$, and so too is the $k$th row of $U$.

Since we can use the previous expression to calculate $u_{kk}$, the $k$th column of $U$ is uniquely determined.

Finally, multiplying the $i$th row of $L$ by the $k$th column of $U$, we find that $a_{ik} = \sum_{m=1}^{k-1} l_{im}u_{mk} + l_{ik}u_{kk}$. All of the terms of $L$ in this expression, besides $l_{ik}$, occur in the first $(k-1)$ columns of $L$ and are therefore uniquely determined. Moreover, we can obtain $u_{kk}$ from the previous step, and we know that $u_{kk} \neq 0$. Thus, $l_{ik} = (1/u_{kk})\left(a_{ik} - \sum_{m=1}^{k-1} l_{im}u_{mk}\right)$ is uniquely determined for all $(k+1) \leq i \leq n$, and so too is the $k$th column of $L$.

By the principle of mathematical induction, we have therefore shown that both $L$ and $U$ are unique. □

As you may have observed, LU decomposition and Gaussian elimination without pivoting are equivalent, since the same matrices $L$ and $U$ are produced using either method. Therefore, we have already seen an algorithm to obtain the LU decomposition under the guise of performing Gaussian elimination! However, the proof of Theorem 5 gives us an alternative method of computing the LU decomposition of $A$, which is sometimes referred to as *Doolittle's method*.

[2] We can safely perform the division by $u_{11}$ as a consequence of our assumption that all leading principal submatrices of $A$ are invertible.