Gaussian Elimination and the LU Decomposition

Varun Shankar

March 17, 2019

1 Introduction

We will now discuss how to solve linear systems of the form $Ax = b$. For convenience, we will adopt the notation of capital letters for matrices and small letters for vectors. The goal will be to learn how to write solvers in Matlab (or conceptually in any language). We will use polynomial interpolation as our motivating example. In addition, we will keep track of costs.

2 Gaussian Elimination

This is a technique for solving large linear systems involving square matrices of dimension $N \times N$. Henceforth, we will also occasionally append a subscript to a matrix indicating its dimensions: $A_{N\times N}$.

The best way to recall how we go through Gaussian elimination is to do an example.

2.1 Example 1

Consider the following linear system $Ax = b$ taken off of Wikipedia, where we have

$$\underbrace{\begin{bmatrix} 1 & 3 & 1 \\ 1 & 1 & -1 \\ 3 & 11 & 5 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 9 \\ 1 \\ 35 \end{bmatrix}}_{b}. \tag{1}$$

The typical approach of Gaussian elimination is to go along the diagonal, and attempt to zero out every element below the diagonal by using *row transformations*. Any transformation we apply to the matrix must also be applied to the right hand side, ensuring that we aren't really changing the system of

equations. Thus, it is common to form the *augmented matrix* and perform row transformations on this matrix:

$$\begin{bmatrix} 1 & 3 & 1 & 9 \\ 1 & 1 & -1 & 1 \\ 3 & 11 & 5 & 35 \end{bmatrix}, \tag{2}$$

where we simply took the rhs b and appended it to A. Let's work through this example. The goal is to bring A to the *row-echelon form*: we want all non-zero rows to be above any rows of all zeros, and we want the leading coefficient of a non-zero row to be to the right of the leading coefficient of the row above it.

1. The first step is to transform the matrix so that the first column has only zeros below the diagonal. This means we need to transform both rows 2 and 3.
   So, first, we apply the transformation $R_2 \leftarrow R_2 - R_1$ to give us:

   $$\begin{bmatrix} 1 & 3 & 1 & 9 \\ 0 & -2 & -2 & -8 \\ 3 & 11 & 5 & 35 \end{bmatrix}. \tag{3}$$

   Then, we apply the transformation $R_3 \leftarrow R_3 - 3R_1$ to give us:

   $$\begin{bmatrix} 1 & 3 & 1 & 9 \\ 0 & -2 & -2 & -8 \\ 0 & 2 & 2 & 8 \end{bmatrix}. \tag{4}$$

   Now column 1 has only 1 non-zero element in it.
2. We now want to induce a zero below element $(2, 2)$ of the matrix (row 2, column 2, indices starting at 1). This is easily done by the transformation $R_3 \leftarrow R_3 + R_2$, giving us:

   $$\begin{bmatrix} 1 & 3 & 1 & 9 \\ 0 & -2 & -2 & -8 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{5}$$

   This matrix is now in row-echelon form. Of course, in this example, we ended up with the last row being full of zeros, implying that the matrix is *singular*. More specifically, we have more equations than unknowns.

2.2 Example 2

Let's do another example, again off of Wikipedia. Consider the system:

$$\underbrace{\begin{bmatrix} 2 & 1 & -1 \\ -3 & -1 & 2 \\ -2 & 1 & 2 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 8 \\ -11 \\ -3 \end{bmatrix}}_{b}. \tag{6}$$

Remember, we want to zero elements below the diagonal. Doing so transforms A into an *upper-triangular* matrix, one with zeros below its main diagonal. A linear system with an upper triangular matrix is easily solved by back-substitution, which we will demonstrate now.

First, let's proceed with Gaussian elimination on the augmented system:

$$\begin{bmatrix} 2 & 1 & -1 & 8 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{bmatrix}. \tag{7}$$

1. Looking at the first column, we can induce zeros below the diagonal using the two transformations. First, we use $R_2 \leftarrow R_2 + \frac{3}{2}R_1$. This gives:

$$\begin{bmatrix} 2 & 1 & -1 & 8 \\ 0 & \frac{1}{2} & \frac{1}{2} & 1 \\ -2 & 1 & 2 & -3 \end{bmatrix}. \tag{8}$$

   Then, we use $R_3 \leftarrow R_3 + R_1$. This gives:

$$\begin{bmatrix} 2 & 1 & -1 & 8 \\ 0 & \frac{1}{2} & \frac{1}{2} & 1 \\ 0 & 2 & 1 & 5 \end{bmatrix}. \tag{9}$$

2. Next, we need to zero out the element below the main diagonal on the second column. This is easily done by the row transformation $R_3 \leftarrow R_3 - 4R_1$, giving:

$$\begin{bmatrix} 2 & 1 & -1 & 8 \\ 0 & \frac{1}{2} & \frac{1}{2} & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix}. \tag{10}$$

The linear system $Ax = b$ was transformed into a different linear system $Ux = y$, where U is upper-triangular. What was the point of this? Well, we can now easily solve for x_1, x_2 and x_3. To see this, write the above augmented system back as a set of equations:

$$2x_1 + x_2 - x_3 = 8 \tag{11}$$

$$0x_1 + \frac{1}{2}x_2 + \frac{1}{2}x_3 = 1 \tag{12}$$

$$0x_1 + 0x_2 - x_3 = 1. \tag{13}$$

Looking at the above equations, we can see that we can solve for the x values in *reverse*.

1. First, solve for x_3. $x_3 = -1$.
2. Then, solve $\frac{1}{2}x_2 + \frac{1}{2}x_3 = 1$ for x_2.
3. Then, solve $2x_1 + x_2 - x_3 = 8$ for x_1.

This is called **back-substitution**.

In general, to solve a linear system $Ax = b$, we can use Gaussian elimination to transform this into a system of the form $Ux = y$, which we can then solve by back-substitution. The problem with this approach is if we have multiple right hand sides. Then, we need to redo the Gaussian elimination process for each right hand side.

Unless, of course, we decide to get more clever!

3 LU Decomposition

The LU decomposition is simply a way to make Gaussian Elimination more efficient for multiple right hand sides. The idea is to *decompose* A into two matrices so that $A = LU$. Then,

$$LUx = b, \tag{14}$$

$$\implies Ly = b, \tag{15}$$

$$Ux = y, \tag{16}$$

where U and y are the same matrices we get from Gaussian Elimination! So, what is L?

L is a *unit lower-triangular matrix*. This is a matrix with 1 on the diagonal, and zeros above the diagonal. But how do we compute L? Simple: we record the row operations we performed on the matrix A in L; L is a **diary of row transformations, but with signs flipped**.

3.1 Example: Method 1

Let's build an LU decomposition for the second example above. First, let's start with an L that looks like this:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ * & 1 & 0 \\ * & * & 1 \end{bmatrix}. \tag{17}$$

Recall that A looks like this:

$$A = \begin{bmatrix} 2 & 1 & -1 \\ -3 & -1 & 2 \\ -2 & 1 & 2 \end{bmatrix}. \tag{18}$$

We will transform A into U, and since L is our (flipped) diary, record row transformations there. More importantly, we don't need to use the right hand side in any way!

Our first transformation was $R_2 \leftarrow R_2 + \frac{3}{2}R_1$. This gives us:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{3}{2} & 1 & 0 \\ * & * & 1 \end{bmatrix}, \ \tilde{A} = \begin{bmatrix} 2 & 1 & -1 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ -2 & 1 & 2 \end{bmatrix}, \tag{19}$$

where $\tilde{A}$ is the A you get after the first row transformation. Notice that L has a $-\frac{3}{2}$ in it.

Our next transformation was $R_3 \leftarrow R_3 + R_1$. This gives:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{3}{2} & 1 & 0 \\ -1 & * & 1 \end{bmatrix}, \ \tilde{A} = \begin{bmatrix} 2 & 1 & -1 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 2 & 1 \end{bmatrix}. \tag{20}$$

Since our multiplier was 1, we stored a -1 in L.

Our final transformation was $R_3 \leftarrow R_3 - 4R_1$, which gives us:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{3}{2} & 1 & 0 \\ -1 & 4 & 1 \end{bmatrix}, \ \tilde{A} = U = \begin{bmatrix} 2 & 1 & -1 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & -1 \end{bmatrix}. \tag{21}$$

Since our multiplier was -4, we stored a 4 in L. It's easy to verify that $A = LU$.

3.2 Example: Method 2

If the above approach to construct L seems a bit ad-hoc, consider the following way of constructing L, equivalent to the technique above. Let's again go through the zeroing out process on the matrix A, and keep track of the transformations carefully.

First, we have A, which looks like:

$$A = \begin{bmatrix} 2 & 1 & -1 \\ -3 & -1 & 2 \\ -2 & 1 & 2 \end{bmatrix}. \tag{22}$$

The first column transformation is $R_2 \leftarrow R_2 + \frac{3}{2}R_1$. We can write this as a matrix-matrix product of a transformation matrix L_1 with A. To do so, we need a matrix L_1 that leaves everything except R_2 untouched, and sets the first elements of R_2 to zero. This matrix-matrix product can be written as:

$$L_1 A = \begin{bmatrix} 1 & 0 & 0 \\ \frac{3}{2} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & -1 \\ -3 & -1 & 2 \\ -2 & 1 & 2 \end{bmatrix} = \begin{bmatrix} 2 & 1 & -1 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ -2 & 1 & 2 \end{bmatrix}. \tag{23}$$

Next, we want to transform L_1A to zero out the last element in the first column. Recall that this transformation was $R_3 \leftarrow R_3 + R_1$. We can write this transformation as another matrix product $L_2(L_1A)$:

$$L_2(L_1A) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & -1 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ -2 & 1 & 2 \end{bmatrix} = \begin{bmatrix} 2 & 1 & -1 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 2 & 1 \end{bmatrix} \tag{24}$$

We have one more transformation to zero out the 2 at the bottom of the second column. This transformation was $R_3 \leftarrow R_3 - 4R_1$, which can expressed as the product $L_3(L_2L_1A)$:

$$L_3(L_2L_1A) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -4 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & -1 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 & -1 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & -1 \end{bmatrix} = U. \tag{25}$$

So we have

$$L_3L_2L_1A = U, \tag{26}$$

$$\implies A = (L_3L_2L_1)^{-1}U, \tag{27}$$

$$\implies A = L_1^{-1}L_2^{-1}L_3^{-1}U. \tag{28}$$

But, since $A = LU$, we have

$$L = L_1^{-1}L_2^{-1}L_3^{-1}. \tag{29}$$

It's pretty easy to invert the L_1, L_2 and L_3 matrices. We only need to flip the signs on the individual off-diagonal elements. This is because each of the L matrices is an "atomic triangular matrix", a matrix that has off-diagonal entries zero except in a single column. Thus, we have:

$$L_1^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{3}{2} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{30}$$

$$L_2^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}, \tag{31}$$

$$L_3^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 4 & 1 \end{bmatrix}. \tag{32}$$

Multiplying to get L, we obtain the same matrix as in method 1:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{3}{2} & 1 & 0 \\ -1 & 4 & 1 \end{bmatrix}. \tag{33}$$

This ended up working out very neatly because the matrix L is unit lower triangular. Method 1 is hence the most straightforward way to form L, but method 2 explains why method 1 works.

3.3 Algorithm

From the above descriptions, we can write out a simple procedure in Matlab-style notation for the LU decomposition/factorization. This is shown in Algorithm 1. This costs $O(\frac{2}{3}N^3)$ operations. Further, in practice, it is easy to store

Algorithm 1 LU Decomposition

Given: A, an $N \times N$ matrix.
Generate: L, a unit lower triangular matrix, and U, an upper triangular matrix so that $A = LU$.

Initialize $U = A$, $L = I$.
for i = 1,N-1 **do**
    for j = i+1,N **do**
        $L(j,i) = U(j,i)/U(i,i)$.
        $U(j, i:N) = U(j, i:N) - L(j,i)U(i, i:N)$.
    end for
end for

L and U by overwriting A, since we know the diagonal of L only contains ones.

There's a problem with this algorithm though: notice the division by $U(i,i)$. What if $U(i,i) \approx 0$? The algorithm then breaks down. The element $U(i,i)$ is called a **pivot**.

3.4 Pivoting

Algorithm 2 LU Decomposition with Partial Pivoting

Given: A, an $N \times N$ matrix.
Generate: L, a unit lower triangular matrix, U, an upper triangular matrix, and P, a pivoting matrix, so that $PA = LU$.

Initialize $U = A$, $L = I$, $P = I$.
for i = 1,N-1 **do**
    Find $m \geq i$ that results in maximum $|U(m,i)|$ (index of largest column element in absolute value).
    Swap rows $U(i, i:N)$ and $U(m, i:N)$.
    Swap rows $L(i, 1:i-1)$ and $L(m, 1:i-1)$.
    Swap row m of P and row i of P.
    for j = i+1,N **do**
        $L(j,i) = U(j,i)/U(i,i)$.
        $U(j, i:N) = U(j, i:N) - L(j,i)U(i, i:N)$.
    end for
end for

To overcome the issue of a very small or zero pivot, we will utilize a strategy called **partial row pivoting**. The idea is very simple, and something you've

probably employed when using Gaussian Elimination: swap rows so that the column with non-zero entry is at the "top" of the matrix. If a column is full of zeros at any point in Gaussian Elimination, we know that the matrix is singular and can stop.

However, this approach presents its own problem. How do we keep track of row swaps so that we can also apply them to any arbitrary right hand side that may be given to us? The answer, much like in the LU decomposition, is to keep a **diary of row swaps in a permutation/pivot matrix** P. Let's see an example of how to build a pivot matrix P, then include pivoting strategies in our algorithm.

Consider the matrix

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 5 \\ 4 & 6 & 8 \end{bmatrix}. \tag{34}$$

After zeroing out the below-diagonal elements in the first column, we'd have:

$$L_1 A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 3 \\ 0 & 2 & 4 \end{bmatrix}. \tag{35}$$

Now, if we try to apply Algorithm 1 to zero out the 2 in the second column, the algorithm breaks down due to an attempt to divide by zero. A simple solution is to *swap* rows and keep track of the row swaps. Let P_1 be a pivot matrix. We will select P_1 so that it swaps rows 2 and 3, and leaves row 1 untouched:

$$P_1 L_1 A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} L_1 A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 2 & 4 \\ 0 & 0 & 3 \end{bmatrix}. \tag{36}$$

Now, we can proceeed with the LU algorithm as stated in Algorithm 1. Of course, since we already have a zero below the diagonal, we are done, and have obtained the matrix U. In general, we are looking for one final pivoting matrix of the form $PA = LU$. If we have a few pivoting matrices $P_1, \ldots, P_k$, the pivoting matrix P is defined as

$$P = P_k P_{k-1} \ldots P_1, \tag{37}$$

the product in reverse order. We now need to incorporate this pivoting strategy for each "zeroing out" step (row transformation) into Algorithm 1. Each pivoting step needs to swap the appropriate rows in L and U. This is given in Algorithm 2.

Other pivoting strategies are possible. Instead of looking for the largest column entry and swapping the corresponding rows, one could look for the largest row entry and swap the columns. This is called column pivoting, and is quite rare. In addition, one could swap both rows and columns by picking the largest along one or the other; this is more stable (in a sense yet to be discussed), but is much more expensive.

4 Triangular Solves

Once we finish our (P)LU decomposition, we need to solve the linear system $Ax = b$, where b may be either a vector or a matrix of right hand sides. Thus, we have:

$$Ax = b, \tag{38}$$
$$\implies PAx = Pb, \tag{39}$$
$$\implies LUx = Pb, \tag{40}$$
$$\implies Ly = Pb,\ Ux = y. \tag{41}$$

We need to solve two *triangular* systems in succession: first, we need to solve $Ly = b$, then we need to solve $Ux = y$. Let's consider the $3x3$ case. For example, if we have

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = Pb = \begin{bmatrix} 1 \\ 1 \\ 3 \end{bmatrix}, \tag{42}$$

we can solve for y by **forward substitution**: first solve $1y_1 = 1$, then solve $2y_1 + y_2 = 1$, then solve $3y_1 + 2y_2 + y_3 = 3$. This is summarized in Algorithm 3. The system $Ux = y$ can be solved by the opposite approach: start from the

Algorithm 3 Forward Substitution

Given: L, an $N \times N$ unit lower-triangular matrix.
Given: $d = Pb$, a right hand side vector.
Generate: y, the solution to $Ly = d$.
1: Initialize $y = d$.
2: **for** i = 2,N **do**
3: $\quad y(i) = d(i) - L(i, 1 : i-1) * y(1 : i-1)$.
4: **end for**

bottom and work your way up. This is called **backward substitution**, and is summarized in Algorithm 4. Forward and backward substitution each cost

Algorithm 4 Back Substitution

Given: U, an $N \times N$ upper-triangular matrix.
Given: y, a right hand side vector.
Generate: x, the solution to $Ux = y$.
1: Initialize x to a vector of N zeros.
2: Set $x(N) = y(N)/U(N, N)$.
3: **for** i = N-1,1 **do**
4: $\quad x(i) = (y(i) - U(i, i+1 : N) * x(i+1 : N))/U(i, i)$.
5: **end for**

$O(N^2)$ operations. This means that if you have multiple right hand sides, it

is cheap to compute the PLU decomposition for $O(N^3)$ operations, and then simply use repeatedly for $O(N^2)$ cost. If we instead did Gaussian elimination (or Gauss-Jordan elimination) and all the right hand sides were not known ahead of time, we would incur an extra $O(N^3)$ cost for each right hand side (rather than an $O(N^2)$ cost).

5 Looking ahead

We will discuss the stability properties of numerical algorithms, and see whether Gaussian Elimination (in the form of LU decomposition) is stable.