Systems of Equations - LU Factorization

We now shift to the next major topic: solving a linear system of equations $A\mathbf{x} = \mathbf{b}$.

Linear systems appear in models for traffic flow and other networks, Google search algorithms, computer graphics, quantum computing, Markov chains (probability)... In particular, it is often the case that we solve $A\mathbf{x} = \mathbf{b}_1$, followed by $A\mathbf{x} = \mathbf{b}_2$, then $A\mathbf{x} = \mathbf{b}_3$, and so on. That is, A is a "structure" matrix - doesn't really change - and b_i is a "loading vector" - forces coming in that change. For instance, this might be the case when testing designs. Our factorizations are largely designed to make later solves more efficient - avoid repeating work on A.

Formally, we call A the coefficient matrix, size $m \times n$, the variable vector is $\mathbf{x}$, size $n \times 1$, and $\mathbf{b}$ is the right-hand-side vector, size $m \times 1$.

Example 1. Write the following system of equations as an augmented matrix and solve.

$$\begin{aligned} 2x + 3y &= 0 \\ 4x + 2y &= 8 \end{aligned}$$

Example 2. Solve the following system.

$$\left[\begin{array}{rrr:r} 6 & 4 & 2 & 18 \\ -3 & -12 & 1 & 5 \\ 0 & 5 & 1 & -3 \end{array}\right]$$

The row operations used during the process in this example can be stored for re-use in the LU factorization. The matrices L and U are so named because they are *lower triangular* and *upper triangular* matrices, meaning all entries above and below (respectively) the main diagonal are zeros. The matrix L should be *unit* lower triangular, meaning the diagonal entries are 1's. Then at each row elimination step, write the operation in the form $R_b - cR_p$ where R_p is the row containing the *pivot*, the number being used to introduce zeros. The pivot should be the leading nonzero term in its row. Then store the constant c in the corresponding entry in L.

Example 3. For $A = \begin{bmatrix} 6 & 4 & 2 & 4 \\ -3 & -12 & 1 & 1 \\ 0 & 5 & 1 & 1 \end{bmatrix}$, the factorization goes like this.

1. Begin with $L = \begin{bmatrix} 1 & 0 & 0 \\ & 1 & 0 \\ & & 1 \end{bmatrix}$ and $U = \begin{bmatrix} & & & \\ 0 & & & \\ 0 & 0 & & \end{bmatrix}$.

2. The first pivot must be the first element of the matrix, which is 6.

3. This pivot must be used to introduce a 0 to the first element of the second row. This is row operation $R_2 - (-\frac{1}{2})R_1$. The scalar in front of the pivot row is the element that goes into L at that location.

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ & & 1 \end{bmatrix} \qquad U = \begin{bmatrix} & & & \\ 0 & & & \\ 0 & 0 & & \end{bmatrix}$$

4. Repeat for all rows below the pivot row. Since we do not need to introduce a 0 to the last row, we are effectively performing $R_3 - 0R_1$. We now have

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ 0 & & 1 \end{bmatrix} \qquad U = \begin{bmatrix} & & & \\ 0 & & & \\ 0 & 0 & & \end{bmatrix}$$

5. The row that remains unchanged is copied as-is into the same row of U.

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ 0 & & 1 \end{bmatrix} \qquad U = \begin{bmatrix} 6 & 4 & 2 & 4 \\ 0 & & & \\ 0 & 0 & & \end{bmatrix}$$

6. A is now row equivalent to $\begin{bmatrix} 6 & 4 & 2 & 4 \\ 0 & -10 & 2 & 3 \\ 0 & 5 & 1 & 1 \end{bmatrix}$. The process begins again; -10 must be the pivot.

7. The next row operation is $R_3 - (-\frac{1}{2})R_2$.

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ 0 & -\frac{1}{2} & 1 \end{bmatrix} \qquad U = \begin{bmatrix} 6 & 4 & 2 & 4 \\ 0 & & & \\ 0 & 0 & & \end{bmatrix}$$

8. The second row of the row reduced A is remaining unchanged, so copy it into U.

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ 0 & -\frac{1}{2} & 1 \end{bmatrix} \qquad U = \begin{bmatrix} 6 & 4 & 2 & 4 \\ 0 & -10 & 2 & 3 \\ 0 & 0 & & \end{bmatrix}$$

9. Update A. $A = \begin{bmatrix} 6 & 4 & 2 & 4 \\ 0 & -10 & 2 & 3 \\ 0 & 0 & 2 & 2.5 \end{bmatrix}$

10. Copy the last unchanged row into U, completing the factorization.

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ 0 & -\frac{1}{2} & 1 \end{bmatrix} \qquad U = \begin{bmatrix} 6 & 4 & 2 & 4 \\ 0 & -10 & 2 & 3 \\ 0 & 0 & 2 & 2.5 \end{bmatrix}$$

$\square$

Once you have an LU factorization, solving the system can be done through two steps of *back substitution*. First, augment L with b and solve for an intermediate result we'll call $\mathbf{y}$. Then augment U with y and solve to get the final answer.

Example 4. Let $A = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ -3 & 2 & 1 \end{bmatrix} \begin{bmatrix} 4 & 6 & 2 \\ 0 & 6 & 3 \\ 0 & 0 & 1 \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} 8 \\ 7 \\ -17 \end{bmatrix}$. Solve $A\mathbf{x} = \mathbf{b}$.

On each repeated solve, we only do the back substitution, and don't have to recalculate L and U. Next goal: find out how much work this saves.

When comparing algorithms, we primarily look at *operation counts*. That's the total number of operations (including $+, -, \times, \div, \surd$, etc.) the algorithm has to perform. Operation counts are a function of the size of the input. We'll assume from here on that A is square, that is, $n \times n$.

Now it is helpful to look at the algorithm in pseudocode form.

Algorithm 1 Computing an LU factorization - Classical Gaussian Elimination

---

```
for j = 1, ..., n-1 do
    for i = j+1, ..., n do
        c = a_ij / a_jj
        for k = j+1, ..., n do
            a_ik = a_ik - c(a_jk)
```

There is a specific notation called "big-O" notation $\mathcal{O}$ that is also convenient for this where the coefficients are also omitted. Saying that Classical Gaussian Elimination is an $\mathcal{O}(n^3)$ algorithm means that as n gets large, terms below cubic become negligible to the total running time of the algorithm, and the number of operations will grow like n^3.

Algorithm 2 Back Substitution

```
for i = n, ..., 1 do
    for j=i+1,..., n do
        b_i = b_i - a_ij(x_j)
    x_i = b_i/a_ii
```

What is the operation count for back-substitution?

Thus back-substitutions are a $\mathcal{O}(n^2)$ algorithm.

Example 5. Estimate the time required to carry out 1000 back-substitutions on a 5000×5000 system where one elimination takes 10 seconds.