LU Decomposition

LU decomposition is a better way to implement Gauss elimination, especially for repeated solving a number of equations with the same left-hand side. That is, for solving the equation $Ax = b$ with different values of b for the same A.

Note that in Gauss elimination the left-hand side (A) and the right-hand side (b) are modi£ed within the same loop and there is no way to save the steps taken during the elimination process. If the equation has to be solved for different values of b, the elimination step has do to done all over again.

Let's take an example where the solutions are needed for different values of b (e.g., determining the position of a moving object from different sets of radar equations). Since elimination takes more than 90% of the computational load, it would be better modify A only, save the results and use them repeatedly with different values of b.

This provides the motivation for LU decomposition where a matrix A is written as a product of a lower triangular matrix L and an upper triangular matrix U. That is, A is decomposed as $A = LU$.

The original equation is to solve

$$Ax - b = 0ﬀ$$

At the end of the Gauss elimination, the resulting equations were

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
a'_{22}x_2 + a'_{23}x_3 + \cdots + a'_{2n}x_n &= b'_2 \\
a''_{33}x_3 + a''_{34}x_4 + \cdots a''_{3n} &= b''_3 \\
&\vdots \\
a_{nn}^{(n-1)}x_n &= b_n^{(n-1)}
\end{aligned}$$

which can be written as

$$Ux - d = 0 \tag{1}$$

Let's premultiply (1) by another matrix L, which results in

$$L(Ux - d) = 0$$

That is,

$$LUx - Ld = 0 \tag{2}$$

Comparing (1) and (2), it is clear that

$$LU = A$$
$$Ld = b \tag{3}$$

To reduce computational load, L is taken an a lower triangular matrix with 1's along the diagonal.

Now, we have two equations to solve

$$\begin{aligned} Ld &= b \qquad (4) \\ Ux = d & \qquad (5) \end{aligned}$$

in order to solve for x. The advantage is that L captures the transformation (using Gauss elimination) from the original matrix A to the upper diagonal matrix U. That is, if L and U are stored, the steps in the Gauss elimination are also stored. Then, if we have to solve the equation for different values of b, we could use the stored values of L and U, instead of doing the elimination once again.

Solving $Ax = b$ using LU decomposition

Decomposition Factor A into $A = LU$. The upper diagonal matrix U is given by the result of the elimination step in Gauss elimination.

$$U = \begin{bmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1n}x_n \\ 0 & a'_{22} & a'_{23} & \cdots & a'_{2n} \\ 0 & 0 & a''_{33} & \cdots & a''_{3n} \\ \vdots & & & & \\ 0 & 0 & \cdots & 0 & a_{nn}^{(n-1)} \end{bmatrix}$$

The lower diagonal matrix L is given by

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ \frac{a_{21}}{a_{11}} & 1 & 0 & 0 & \dots & 0 \\ \frac{a_{31}}{a_{11}} & \frac{a'_{32}}{a'_{22}} & 1 & 0 & \dots & 0 \\ \vdots & & & & & \end{bmatrix} \qquad (6)$$

Substitution This involves two steps

1. Forward substitution: Solve $Ld = b$ to £nd d. The values of d_i are given by

$$\begin{aligned} d_1 &= b_1 \\ d_i &= b_i - \sum_{j=1}^{i-1} l_{ij} b_j \quad i = 2, 3, \dots, n \end{aligned}$$

2. Back substitution: Solve $Ux = d$ to £nd x. The values of x_i are given by

$$\begin{aligned} x_n &= \frac{d_n}{u_{nn}} \\ x_i &= \frac{d_i - \sum_{j=i+1}^{n} u_{ij} x_j}{u_{ii}} \quad i = n-1, n-2, \dots, 1 \end{aligned} \qquad (7)$$

The pseudocode for solving $Ax = b$ via LU decomposition is given below.

```
% Diagonalization

for k = 1 : n-1
```

```
    for i = k+1 : n
        a(i,k) = a(i,k) / a(k,k)
        for j = k+1:n
            a(i,j) = a(i,j) - a(i,k) .  a(k,j)
        end
    end
end

% Forward substitution to solve Ld=b

x(1)=b(1)
for i = 2 : n
    s=0
    for j = 1 : i-1
        s = s + a(i,j) * x(j)
    end
    x(i) = b(i) - s
end

% back substitution to solve Ux=d

x(n) = x(n) / a(n, n)
for i = n-1 : -1 : 1
    s=0
    for j = i+1 to n
        s = s + a( i, j) * x(j)
    end
    x(i) = (x(i) - s)/a(i,i)
end
```

LU decomposition requires $\frac{n^3}{3} + O(n^2)$ operations, which is the same as in the case of Gauss elimination. But the advantage is that once the matrix A is decomposed into $A = LU$, the substitution step can be carried out ef£ciently for different values of b. Note that the elimination step in Gauss elimination takes $\frac{n^3}{3} + O(n)$ operation as opposed to n^2 operations for substitution. The steps of solving $Ax = b$ using LU decomposition are shown in Figure 1.

Finding the inverse of a matrix using LU decomposition

Consider a 3×3 matrix A. Finding the inverse of A involves three sets of linear equations

$$Ax = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$Ay = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

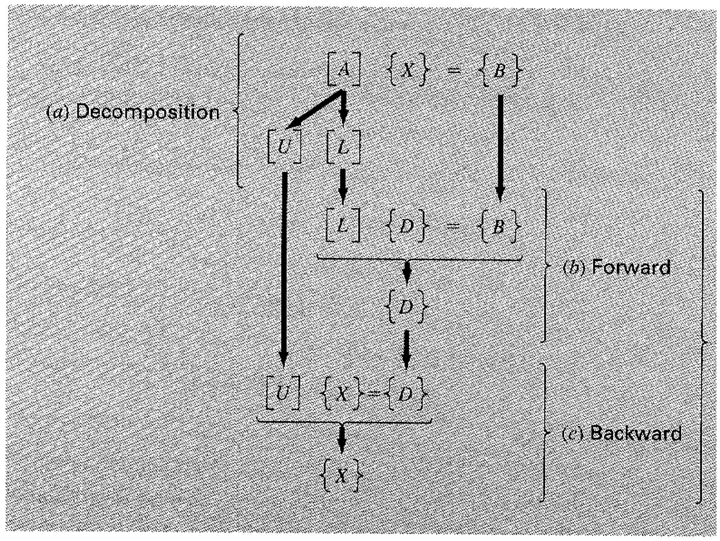

Figure 1: Steps of solving $Ax = b$ using LU decomposition

$$Az = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{8}$$

The the inverse A^{-1} is given by

$$A^{-1} = [x \;\; y \;\; z] \tag{9}$$

where x, y and z are the solutions (column vectors) of the three sets of linear equations given earlier.

The solutions x, y and z can be found using LU decomposition. First decompose A into $A = LU$, save L and U and then carry out the substitution step three times to £nd x, y and z. This is an ef£cient was of £nd the inverse of a matrix.

The pseudocode for £nding the inverse of a matrix is given below:

```
%BEGIN DECOMPOSITION

% Diagonalization

for k = 1 : n-1
    for i = k+1 : n
        a(i,k) = a(i,k) / a(k,k)
        for j = k+1:n
            a(i,j) = a(i,j) - a(i,k) .  a(k,j)
        end
    end
end
```

```
%END DECOMPOSITION

for i = 1 : n
    for j = 1 : n
        if i = j
           b(j) = 1
        else
           b(j) = 0
        end
    end

    x(1)=b(1)
    for k = 2 : n
        s=0
        for j = 1 : k-1
            s = s + a(k,j) * x(j)
        end
        x(k) = b(k) - s
    end

    % back substitution to solve Ux=d

    x(n) = x(n) / a(n, n)
    for k = n-1 : -1 : 1
        s=0
        for j = k+1 to n
            s = s + a( k, j) * x(j)
        end
        x(k) = (x(k) - s)/a(k,k)
    end

    for j = 1: n
        a(j,i) = x(j)
    end
end
```

Resources:

1. Steven Chapra and Raymond Canale, Numerical Methods for Engineers, Fourth Edition, McGraw-Hill, 2002 (see Sections 10.1-10.2).

2. http://csep10.phys.utk.edu/guidry/phys594/lectures/linear_algebra/lanotes/node3.html