LU Factorization

Berlin Chen
Department of Computer Science & Information Engineering
National Taiwan Normal University

Reference:

1. *Applied Numerical Methods with MATLAB for Engineers*, Chapter 10 & Teaching material

Chapter Objectives (1/2)

- Understanding that *LU* factorization involves decomposing the coefficient matrix into two triangular matrices that can then be used to efficiently evaluate different right-hand-side vector
- Knowing how to express Gauss elimination as an *LU* factorization
- Given an *LU* factorization, knowing how to evaluate multiple right-hand-side vectors

Chapter Objectives (2/2)

- Recognizing that Cholesky's method provides an efficient way to decompose a symmetric matrix and that **the resulting triangular matrix and its transpose** can be used to evaluate right-hand-side vectors efficiently
- Understanding in general terms what happens when MATLAB's backslash operator is used to solve linear systems

LU Factorization (1/2)

- Recall that the forward-elimination step of **Gauss elimination** comprises the bulk of the computational effort

Forward Elimination	$\frac{2n^3}{3} + O(n^2)$
Back Substitution	$n^2 + O(n)$
Total	$\frac{2n^3}{3} + O(n^2)$

- ***LU* factorization** methods separate the time-consuming elimination of the matrix [*A*] from the manipulations of the right-hand-side [*b*]

- Once [*A*] has been factored (or decomposed), multiple right-hand-side vectors can be evaluated in an efficient manner

LU Factorization (2/2)

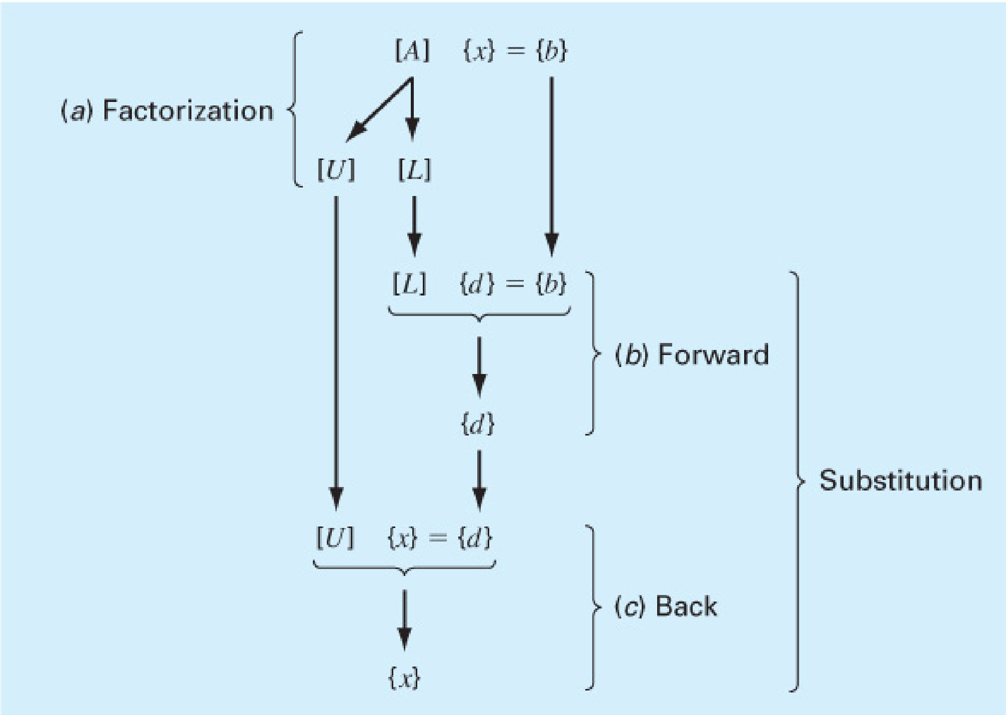

FIGURE 10.1
The steps in *LU* factorization.

- *LU* factorization involves two steps:
  - Factorization to decompose the [*A*] matrix into a product of a lower triangular matrix [*L*] and an upper triangular matrix [*U*]. [*L*] has 1 for each entry on the diagonal
  - Substitution to solve for {*x*}
- Gauss elimination can be implemented using *LU* factorization

Gauss Elimination as *LU* Factorization (1/5)

- [*A*]{*x*}={*b*} can be rewritten as [*L*][*U*]{*x*}={*b*} using *LU* factorization
- The *LU* factorization algorithm requires the same total flops as for Gauss elimination
- The main advantage is once [*A*] is decomposed, the same [*L*] and [*U*] can be used for multiple {*b*} vectors

- MATLAB's `lu` function can be used to generate the [*L*] and [*U*] matrices:

```
[L, U] = lu(A)
```

Gauss Elimination as *LU* Factorization (2/5)

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \end{Bmatrix}$$

The first step in Gauss elimination is to multiply row 1 by the factor [recall Eq. (9.9)]

$$f_{21} = \frac{a_{21}}{a_{11}}$$

and subtract the result from the second row to eliminate a_{21}. Similarly, row 1 is multiplied by

$$f_{31} = \frac{a_{31}}{a_{11}}$$

and the result subtracted from the third row to eliminate a_{31}. The final step is to multiply the modified second row by

$$f_{32} = \frac{a'_{32}}{a'_{22}}$$

and subtract the result from the third row to eliminate a'_{32}.

This matrix, in fact, represents an efficient storage of the *LU* factorization of $[A]$,

$$[A] \rightarrow [L][U] \tag{10.11}$$

where

$$[U] = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a'_{22} & a'_{23} \\ 0 & 0 & a''_{33} \end{bmatrix} \tag{10.12}$$

and

$$[L] = \begin{bmatrix} 1 & 0 & 0 \\ f_{21} & 1 & 0 \\ f_{31} & f_{32} & 1 \end{bmatrix} \tag{10.13}$$

The following example confirms that $[A] = [L][U]$.

Gauss Elimination as *LU* Factorization (3/5)

$$[A] = \begin{bmatrix} 3 & -0.1 & -0.2 \\ 0.1 & 7 & -0.3 \\ 0.3 & -0.2 & 10 \end{bmatrix}$$

After forward elimination, the following upper triangular matrix was obtained:

$$[U] = \begin{bmatrix} 3 & -0.1 & -0.2 \\ 0 & 7.00333 & -0.293333 \\ 0 & 0 & 10.0120 \end{bmatrix}$$

The factors employed to obtain the upper triangular matrix can be assembled into a lower triangular matrix. The elements a_{21} and a_{31} were eliminated by using the factors

$$f_{21} = \frac{0.1}{3} = 0.0333333 \qquad f_{31} = \frac{0.3}{3} = 0.1000000$$

and the element a_{32} was eliminated by using the factor

$$f_{32} = \frac{-0.19}{7.00333} = -0.0271300$$

Example 10.1

Thus, the lower triangular matrix is

$$[L] = \begin{bmatrix} 1 & 0 & 0 \\ 0.0333333 & 1 & 0 \\ 0.100000 & -0.0271300 & 1 \end{bmatrix}$$

Consequently, the *LU* factorization is

$$[A] = [L][U] = \begin{bmatrix} 1 & 0 & 0 \\ 0.0333333 & 1 & 0 \\ 0.100000 & -0.0271300 & 1 \end{bmatrix} \begin{bmatrix} 3 & -0.1 & -0.2 \\ 0 & 7.00333 & -0.293333 \\ 0 & 0 & 10.0120 \end{bmatrix}$$

This result can be verified by performing the multiplication of $[L][U]$ to give

$$[L][U] = \begin{bmatrix} 3 & -0.1 & -0.2 \\ 0.0999999 & 7 & -0.3 \\ 0.3 & -0.2 & 9.99996 \end{bmatrix}$$

where the minor discrepancies are due to roundoff.

Gauss Elimination as *LU* Factorization (4/5)

- To solve [*A*]{*x*}={*b*}, first decompose [*A*] to get [*L*][*U*]{*x*}={*b*}
- Set up and solve [*L*]{*d*}={*b*}, where {*d*} can be found using *forward* substitution
- Set up and solve [*U*]{*x*}={*d*}, where {*x*} can be found using *backward* substitution
- In MATLAB:

```
[L, U] = lu(A)
d = L\b
x = U\d
```

Gauss Elimination as *LU* Factorization (5/5)

$$\begin{bmatrix} 3 & -0.1 & -0.2 \\ 0.1 & 7 & -0.3 \\ 0.3 & -0.2 & 10 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 7.85 \\ -19.3 \\ 71.4 \end{Bmatrix}$$

and that the forward-elimination phase of conventional Gauss elimination resulted in

$$\begin{bmatrix} 3 & -0.1 & -0.2 \\ 0 & 7.00333 & -0.293333 \\ 0 & 0 & 10.0120 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 7.85 \\ -19.5617 \\ 70.0843 \end{Bmatrix}$$

①

The forward-substitution phase is implemented by applying Eq. (10.8):

$$\begin{bmatrix} 1 & 0 & 0 \\ 0.0333333 & 1 & 0 \\ 0.100000 & -0.0271300 & 1 \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix} = \begin{Bmatrix} 7.85 \\ -19.3 \\ 71.4 \end{Bmatrix}$$

or multiplying out the left-hand side:

$$\begin{aligned} d_1 \qquad\qquad\qquad &= 7.85 \\ 0.0333333 d_1 + \qquad d_2 \qquad &= -19.3 \\ 0.100000 d_1 - 0.0271300 d_2 + d_3 &= 71.4 \end{aligned}$$

We can solve the first equation for $d_1 = 7.85$, which can be substituted into the second equation to solve for

$$d_2 = -19.3 - 0.0333333(7.85) = -19.5617$$

②

Both d_1 and d_2 can be substituted into the third equation to give

$$d_3 = 71.4 - 0.1(7.85) + 0.02713(-19.5617) = 70.0843$$

Thus,

$$\{d\} = \begin{Bmatrix} 7.85 \\ -19.5617 \\ 70.0843 \end{Bmatrix}$$

This result can then be substituted into Eq. (10.3), $[U]\{x\} = \{d\}$:

$$\begin{bmatrix} 3 & -0.1 & -0.2 \\ 0 & 7.00333 & -0.293333 \\ 0 & 0 & 10.0120 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 7.85 \\ -19.5617 \\ 70.0843 \end{Bmatrix}$$

which can be solved by back substitution (see Example 9.3 for details) for the final solution:

$$\{x\} = \begin{Bmatrix} 3 \\ -2.5 \\ 7.00003 \end{Bmatrix}$$

Cholesky Factorization

- Symmetric systems occur commonly in both mathematical and engineering/science problem contexts, and there are special solution techniques available for such systems
- The *Cholesky factorization* is one of the most popular of these techniques, and is based on the fact that a symmetric matrix can be decomposed as $[A]=[U]^T[U]$, where T stands for transpose

$$u_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} u_{ki}^2}$$

$$u_{ij} = \frac{a_{ij} - \sum_{k=1}^{i-1} u_{ki} u_{kj}}{u_{ii}} \qquad \text{for } j = i+1, \ldots, n$$

- The rest of the process is similar to *LU* decomposition and Gauss elimination, except only one matrix, $[U]$, needs to be stored