2.3. Factorizations

- Reading: Trefethen and Bau (1997), Lecture 20
- Solve the \( n \times n \) linear system

\[
Ax = b
\]  \( (1) \)

by Gaussian elimination

- Gaussian elimination is a direct method
  * The solution is found after a finite number of operations
- Iterative methods find a solution as the number of operations tend to infinity
  * Iteration is superior for large sparse systems

- Gaussian elimination is a factorization of \( A \) as

\[
A = LU
\]  \( (2a) \)

- \( L \) is unit lower triangular and \( U \) is upper triangular
LU Factorization

\( A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \) \hfill (2b)

\( L = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \cdots & 0 \\ l_{31} & l_{32} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & 1 \end{bmatrix} \), \quad U = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ 0 & 0 & \cdots & u_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{bmatrix} \) \hfill (2c)

- Once \( L \) and \( U \) have been determined

\[ A \mathbf{x} = L U \mathbf{x} = \mathbf{b} \]

- Let

\[ U \mathbf{x} = \mathbf{y} \] \quad (3a)

- Then

\[ L \mathbf{y} = \mathbf{b} \] \quad (3b)

- Solve (3b) for \( \mathbf{y} \) by forward substitution and (3a) for \( \mathbf{x} \) by backward substitution
LU Factorization

• Classical Gaussian elimination
  – Use row saxpys to reduce $A$ to upper triangular form $U$
  – Show that these row operations are the product of $A$ and lower triangular matrices $L_k$, $k = 1 : n - 1$, i.e.,
    \[
    L_{n-1} \cdots L_2 L_1 A = U
    \]  
    (4a)
  – Thus
    \[
    L = L_1^{-1} L_2^{-1} \cdots L_{n-1}^{-1}
    \]  
    (4b)
  – cf. Trefethen and Bau (1997), Lecture 20

• We use a direct factorization
  – Multiply the first row of $L$ by $U$ to obtain the first row of $A$
    
    \[
    a_{1j} = (1) u_{1j}, \quad j = 1 : n
    \]
    
    * Thus
    
    \[
    u_{1j} = a_{1j}, \quad j = 1 : n
    \]
  – Multiply $L$ by the first column of $U$ to obtain the first column of $A$
    \[
    l_{i1} u_{11} = a_{i1}, \quad i = 2 : n
    \]
    or
    \[
    l_{i1} = a_{i1} / u_{11}, \quad i = 2 : n
    \]
    * Redundant information with $i = 1$ is not written
    * The algorithm fails if $u_{11} = a_{11} = 0
LU Factorization

• Continuing
  – Multiply the second row of $L$ by $U$ to obtain the second row of $A$

  \[ l_{21} u_{1j} + u_{2j} = a_{2j}, \quad j = 2 : n \]

  * Thus,

  \[ u_{2j} = a_{2j} - l_{21} u_{1j}, \quad j = 2 : n \]

  – The general procedure is

  \[ u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj}, \quad j = i : n, \quad (5a) \]

  \[ l_{ji} = \frac{1}{u_{ii}} (a_{ji} - \sum_{k=1}^{i-1} l_{jk} u_{ki}), \quad j = i + 1 : n, \]

  \[ i = 1 : n - 1 \quad (5b) \]

  * The sums are zero when the lower limit exceeds the upper one

  * The procedure fails if $u_{ii} = 0$ for some $i$

  * The matrices $L$ and $U$ can be stored in the locations used for the same elements of $A$
LU Factorization

function A = lufactor(A)
% lufactor: Compute the LU decomposition of an n-by-n
% matrix A by direct factorization. On return, L is
% stored in the lower triangular part of A and U is
% stored in the upper triangular part.

[n n] = size(A);
% Loop over the rows
for i = 1: n - 1
% Calculate the i th column of L
for j = i + 1: n
    for k = 1: i - 1
        A(j,i) = A(j,i) - A(j,k)*A(k,i);
    end
    A(j,i) = A(j,i)/A(i,i);
end
% Calculate the (i + 1) th row of U
for j = i + 1: n
    for k = 1: i
        A(i+1,j) = A(i+1,j) - A(i+1,k)*A(k,j);
    end
end
end
LU Factorization

• Observe:

  i. The first row of $U$ needs no explicit calculation
  ii. The 1s on the diagonal of $L$ are not stored
  iii. MATLAB Loops do not execute when the lower index exceeds the upper one
  iv. Classical Gaussian elimination corresponds to a different loop ordering

  – cf. Trefethen and Bau (1997), Lecture 20

• Example 1. Factor

  $A = \begin{bmatrix}
  1 & 2 & 0 & -4 \\
  -1 & 0 & 6 & 2 \\
  3 & -2 & -25 & 0 \\
  -2 & -3 & 4 & 4
  \end{bmatrix}$

  – $i = 1$: Calculate the first column of $L$ and the second row of $U$

  $A \Rightarrow \begin{bmatrix}
  1 & 2 & 0 & -4 \\
  -1 & 2 & 6 & -2 \\
  3 & -2 & -25 & 0 \\
  -2 & -3 & 4 & 4
  \end{bmatrix}$
LU Factorization

- $i = 2$: Calculate the second column of $L$ and the third row of $U$

$$A \Rightarrow \begin{bmatrix}
1 & 2 & 0 & -4 \\
-1 & 2 & 6 & -2 \\
3 & -4 & -1 & 4 \\
-2 & 1/2 & 4 & 4
\end{bmatrix}$$

- $i = 3$: Calculate the third column of $L$ and the fourth row of $U$

$$A \Rightarrow \begin{bmatrix}
1 & 2 & 0 & -4 \\
-1 & 2 & 6 & -2 \\
3 & -4 & -1 & 4 \\
-2 & 1/2 & -1 & 1
\end{bmatrix}$$

- The $L$ and $U$ factors

$$L = \begin{bmatrix}
1 & 0 & 0 & 0 \\
-1 & 1 & 0 & 0 \\
3 & -4 & 1 & 0 \\
-2 & 1/2 & -1 & 1
\end{bmatrix}, \quad U = \begin{bmatrix}
1 & 2 & 0 & -4 \\
0 & 2 & 6 & -2 \\
0 & 0 & -1 & 4 \\
0 & 0 & 0 & 1
\end{bmatrix}$$
L Exposed

- Elementary transformations

\[
\begin{align*}
\mathbf{L}_1 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ -3 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \end{bmatrix}, & \mathbf{L}_2 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 4 & 1 & 0 \\ 0 & -1/2 & 0 & 1 \end{bmatrix}, \\
\mathbf{L}_3 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}
\end{align*}
\]

- \( \mathbf{L}_k \) is the identity matrix with its \( k \)th column replaced by the \( k \)th column of \( \mathbf{L} \) with the negative of the subdiagonal elements

\[
\mathbf{L}^{-1} = \mathbf{L}_3\mathbf{L}_2\mathbf{L}_1 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 2.5 & 3.5 & 1 & 1 \end{bmatrix}
\]
Forward and Backward Substitution

- Solution of (3b) by forward substitution

\[ y_i = b_i - \sum_{j=1}^{i-1} l_{ij} y_j \]  \hspace{1cm} (6a)

function \( y = \text{forward}(L, b) \)
\% forward: Solution of a n-by-n lower triangular system
\% \( Ly = b \) by forward substitution.

\[
[n \ n] = \text{size}(L);
y(1) = b(1);
\text{for } i = 2:n
y(i) = b(i) - \text{dot}(L(i,1:i-1)', y(1:i-1));
\text{end}
\]

- Solve (3a) by backward substitution

\[ x_i = \frac{1}{u_{ii}} [y_i - \sum_{j=i+1}^{n} u_{ij} x_j] \]  \hspace{1cm} (6b)

function \( x = \text{backward}(U, y) \)
\% backward: Solution of a n-by-n upper triangular system
\% \( Ux = y \) by backward substitution.

\[
[n \ n] = \text{size}(U);
x(n) = y(n)/U(n,n);
\text{for } i = n - 1:-1:1
x(i) = (y(i) - \text{dot}(U(i,i+1:n)', x(i+1:n)))/U(i,i);
\text{end}
Forward and Backward Substitution

- *Example 2.* Solve the linear system $A\mathbf{x} = \mathbf{b}$ for $\mathbf{x}$ when $A$ is as given in Example 1 and

$$\mathbf{b} = [-1, 7, -24, 3]^T$$

- Using the forward substitution algorithm

$$\mathbf{y} = [-1, 6, 3, 1]^T$$

- Using backward substitution

$$\mathbf{x} = [1, 1, 1, 1]^T$$
Operation Count

- *Factorization:* (cf. p. 5)
  - Inner column loop:
    * $i - 1$ multiplications and subtractions
    * 1 division
  - Inner row loop: $i$ multiplications and subtractions
  - The summations on $j$ give
    * $(2i - 1)(n - i)$ multiplications and subtractions
    * $n - i$ divisions
  - The summations on $i$ give
    * Multiplications and subtractions:
      \[
      \sum_{i=1}^{n-1} (2i - 1)(n - i) = \frac{n(n - 1)(2n - 1)}{6}
      \]
    * Divisions:
      \[
      \sum_{i=1}^{n-1} (n - 1) = \frac{n(n - 1)}{2}
      \]
    * Formulas (1.2.1, 2) were used to obtain the result
  - A floating point operation (FLOP) is any $+, -, *, /, \sqrt{\cdot}, \cdots$
    * For large $n$ the factorization has about $2n^3/3$ FLOPs
Operation Count

- **Forward Substitution:** (cf. p. 9)
  - \(i - 1\) multiplications and additions/subtractions in the dot product
  * Total multiplications and additions/subtractions:
    \[
    \sum_{i=2}^{n} (i - 1) = \frac{n(n - 1)}{2}
    \]

- **Backward Substitution:**
  - \(n(n - 1)/2\) multiplications and additions/subtractions and \(n\) divisions

- **Forward and Backward Substitution:**
  - For large \(n\), there are about \(2n^2\) FLOPs

- Factorization dominates forward and backward substitution for large \(n\)