MOTIVATION: if some pivot (in Gaussian elimination without pivoting) is exactly equal to 0, then the elimination fails (divide by 0), and if some pivot is very small in magnitude relative to other numbers in the matrix $A$, then the computation may be numerically unstable.

After $k-1$ steps of the forward elimination, we have computed the reduced matrix

$$
A^{(k-1)} = \begin{bmatrix}
a_{11} & a_{12} & \cdots & \cdots & a_{1n} \\
a^{(i)}_{22} & \cdots & \cdots & \cdots & a^{(i)}_{2n} \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
0 & a^{(k-2)}_{k-1,k-1} & a^{(k-2)}_{k-1,n} & \cdots & a^{(k-2)}_{k-1,n} \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
a^{(k-1)}_{nk} & a^{(k-1)}_{nk} & \cdots & a^{(k-1)}_{nk} & a^{(k-1)}_{nn}
\end{bmatrix}.
$$

Two common pivoting strategies:

(i) PARTIAL PIVOTING -- choose $a_{mk}^{(k-1)}$ as the pivot for step $k$, where

$$
\left|a_{mk}^{(k-1)}\right| = \max_{k \leq i \leq n} \left|a_{ik}^{(k-1)}\right|.
$$

If $m \neq k$, then interchange rows $m$ and $k$.

Matrix formulation of partial pivoting:

$$
A^{(1)} = M_{1}P_{1}A
$$

where $P_{1}$ is a permutation matrix that does the appropriate row interchange at step 1. Then

$$
A^{(2)} = M_{2}P_{2}A^{(1)} = M_{2}P_{2}M_{1}P_{1}A
$$

Thus, after $n-1$ steps, we have

$$
U \equiv A^{(n-1)} = M_{n-1}P_{n-1}M_{n-2}P_{n-2} \cdots M_{1}P_{1}A.
$$

Note: the product $M_{n-1}P_{n-1}M_{n-2}P_{n-2} \cdots M_{1}P_{1}$ is not lower triangular, so this is not an $LU$ factorization of $A$. 

56
(ii) COMPLETE PIVOTING -- choose \(a_{mp}^{(k-1)}\) as the pivot for step \(k\), where

\[
\left|a_{mp}^{(k-1)}\right| = \max_{k \leq j \leq n} \max_{k \leq i \leq n} \left|a_{ij}^{(k-1)}\right|.
\]

If \(m \neq k\) or \(p \neq k\), then interchange rows \(m\) and \(k\) and columns \(p\) and \(k\).

Matrix formulation of complete pivoting:

\[
A^{(1)} = M_1 P_1 AQ_1,
\]

where \(P_1\) and \(Q_1\) are permutation matrices. Then

\[
A^{(2)} = M_2 P_2 A^{(1)} Q_2 = M_2 P_2 (M_1 P_1 AQ_1) Q_2,
\]

and so on. After \(n-1\) steps,

\[
A^{(n-1)} = M_{n-1} P_{n-1} M_{n-2} P_{n-2} \cdots M_1 P_1 A Q_1 \cdots Q_{n-2} Q_{n-1} = U
\]

is upper triangular (but note that \(M_{n-1} P_{n-1} M_{n-2} P_{n-2} \cdots M_1 P_1\) is not lower triangular).

In the second paragraph on page 95 of the textbook, it is argued that the effect of row interchanges (in the partial pivoting strategy) is the same as if the rows of \(A\) had been appropriately interchanged initially, and then Gaussian elimination without pivoting applied.

That is, if we were to make the appropriate row interchanges in \(A\) to form a new matrix \(\hat{A}\), and then apply Gaussian elimination without row interchanges to \(\hat{A}\), we would get exactly the same upper triangular matrix \(U\) that is computed by Gaussian elimination with partial pivoting applied to \(A\).

The following is a precise statement of this.

**Theorem**

Let

\[
A^{(1)} = M_1 P_1 A
\]

\[
A^{(2)} = M_2 P_2 A^{(1)}
\]

\[
A^{(3)} = M_3 P_2 A^{(2)}
\]

\[
\vdots
\]

\[
A^{(n-1)} = M_{n-1} P_{n-1} A^{(n-2)} = U
\]
represent the reduction of $A$ to upper triangular form using Gaussian elimination with partial pivoting. Let

$$ \hat{A} = P_{n-1}P_{n-2} \cdots P_2P_1A. $$

If Gaussian elimination without pivoting is applied to $\hat{A}$ giving an upper triangular matrix $\hat{U}$, then $\hat{U} = U$.

Proof. See Theorem 2.9, page 125 of *Introduction to Matrix Computations* by G.W. Stewart.

EXAMPLE

Let

$$ A = \begin{bmatrix} 2 & 4 & 3 & 2 \\ 3 & 6 & 5 & 2 \\ 2 & 5 & 2 & -3 \\ 4 & 5 & 14 & 14 \end{bmatrix}. $$

Apply Gaussian elimination with partial pivoting to $A$:

With

$$ P_1 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \quad P_1A = \begin{bmatrix} 4 & 5 & 14 & 14 \\ 3 & 6 & 5 & 2 \\ 2 & 5 & 2 & -3 \\ 2 & 4 & 3 & 2 \end{bmatrix}. $$

With

$$ M_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3/4 & 1 & 0 & 0 \\ -1/2 & 0 & 1 & 0 \\ -1/2 & 0 & 0 & 1 \end{bmatrix}, \quad M_1P_1A = \begin{bmatrix} 4 & 5 & 14 & 14 \\ 0 & 9/4 & -11/2 & -17/2 \\ 0 & 5/2 & -5 & -10 \\ 0 & 3/2 & -4 & -5 \end{bmatrix}. $$

With

$$ P_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad P_2M_1P_1A = \begin{bmatrix} 4 & 5 & 14 & 14 \\ 0 & 5/2 & -5 & -10 \\ 0 & 9/4 & -11/2 & -17/2 \\ 0 & 3/2 & -4 & -5 \end{bmatrix}. $$
With 

\[
M_2 = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & -9/10 & 1 & 0 \\
0 & -3/5 & 0 & 1
\end{bmatrix}, \quad M_2 P_2 M_1 P_1 A = \begin{bmatrix}
4 & 5 & 14 & 14 \\
0 & 5/2 & -5 & -10 \\
0 & 0 & -1 & 1/2 \\
0 & 0 & -1 & 1
\end{bmatrix}.
\]

With 

\[
P_3 = I \quad \text{and} \quad M_3 = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & -1 & 1
\end{bmatrix}, \quad M_3 P_3 M_2 P_2 M_1 P_1 A = \begin{bmatrix}
4 & 5 & 14 & 14 \\
0 & 5/2 & -5 & -10 \\
0 & 0 & -1 & 1/2 \\
0 & 0 & 0 & 1/2
\end{bmatrix} = U.
\]

Now define 

\[
\hat{A} = P_3 P_2 P_1 A = \begin{bmatrix}
4 & 5 & 14 & 14 \\
2 & 5 & 2 & -3 \\
3 & 6 & 5 & 2 \\
2 & 4 & 3 & 2
\end{bmatrix}
\]

and let 

\[
P = P_3 P_2 P_1 = \begin{bmatrix}
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0
\end{bmatrix}.
\]

Apply Gaussian elimination without pivoting to \( \hat{A} \):

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
-1/2 & 1 & 0 & 0 \\
-3/4 & 0 & 1 & 0 \\
-1/2 & 0 & 0 & 1
\end{bmatrix} \hat{A} = \begin{bmatrix}
4 & 5 & 14 & 14 \\
0 & 5/2 & -5 & -10 \\
0 & 9/4 & -11/2 & -17/2 \\
0 & 3/2 & -4 & -5
\end{bmatrix} = A^{(1)}
\]

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & -9/10 & 1 & 0 \\
0 & -3/5 & 0 & 1
\end{bmatrix} A^{(1)} = \begin{bmatrix}
4 & 5 & 14 & 14 \\
0 & 5/2 & -5 & -10 \\
0 & 0 & -1 & 1/2 \\
0 & 0 & -1 & 1
\end{bmatrix} = A^{(2)}
\]
\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & -1 & 1
\end{bmatrix}
\begin{bmatrix}
2/1000 \\
2/1100 \\
1052/50 \\
141454
\end{bmatrix}
= 
\begin{bmatrix}
4 & 5 & 14 & 14 \\
0 & 5/2 & -5 & -10 \\
0 & 0 & -1 & 1/2 \\
0 & 0 & 0 & 1/2
\end{bmatrix} 
\equiv \hat{U},
\]

which is identical to the upper triangular matrix \( U \) determined by Gaussian elimination with partial pivoting. This example illustrates the above Theorem.

****************************************

The above Theorem also proves the following result.

**Theorem 1.8.8** (page 98)

For any \( n \times n \) nonsingular matrix \( A \), there exists a permutation matrix \( P \) such that \( PA \) has an \( LU \) factorization. That is, such that

\[
PA = LU \quad \text{or} \quad A = P^T LU .
\]

The above example also illustrates Theorem 1.8.8: although there does not exist an \( LU \) factorization of \( A \) (why?), there is a permutation matrix \( P \) such that \( PA \) has an \( LU \) factorization, namely

\[
PA = \hat{L}\hat{U} .
\]

What is \( \hat{L} \)?

**NOTE**

The MATLAB function \( lu \) uses Gaussian elimination with partial pivoting. Execution of

\[
[L, U, P] = \text{lu}(A)
\]
determines matrices \( L, U \) and \( P \) such that

\[
PA = LU .
\]

**Significance of the above Theorem.**

For purposes of analyzing roundoff error, one can assume that all interchanges are done beforehand (as interchanges produce no roundoff error). So one needs to analyze only Gaussian elimination without pivoting.

Notes on implementing Gaussian elimination:
1. Unless $A$ has special properties (e.g., $A$ is positive definite or diagonally dominant), pivoting must be done to insure stability.

2. In practice, the additional expense of complete pivoting is not worthwhile -- partial pivoting is usually sufficient. More on this later.

3. A program that reduces $A$ to upper triangular form must keep track of any interchanges made in order to solve $Ax = b$. For example, one can use a vector to store the pivotal rows (with partial pivoting) as in the algorithm on page 99. The vector $intch$ is such that

$$
intch(k) \begin{cases} 
p, & \text{if row } p \text{ is the pivotal row at step } k \\
0, & \text{if the pivot is 0 at step } k
\end{cases}
$$

Note that if the pivot is 0 at step $k$, then $a_{kk} = a_{k+1,k} = \cdots = a_{nk} = 0$, which implies that $A$ is singular and no elimination is done at this step, although the algorithm continues with the $LU$ factorization.

The vector $intch$ is used to interchange entries in the vector $b$ when $Ax = b$ is solved -- see the algorithm on page 100.

4. From Section 1.7, recall that if $A = LU$ then

$$
L = \begin{bmatrix}
1 \\
m_{21} & 1 \\
m_{31} & m_{32} & 1 \\
& \ddots & \ddots \\
& & \ddots & 1 \\
m_{n1} & m_{n2} & m_{n3} & \cdots & m_{n,n-1} & 1
\end{bmatrix}
$$

where the entries $m_{ij}$ are the multipliers. In the Gaussian elimination algorithm (with or without pivoting), these multipliers should be saved in memory as this costs very little and will save a lot if you have more than one linear system to solve with the same coefficient matrix $A$. See pages 76-78.

As shown at the middle of page 79 (and in Example 1.7.25 on page 82), the matrices $L$ and $U$ can be stored in one $n \times n$ array (as the 1’s on the diagonal of $L$ do not have to be stored):
In the following algorithms, this $n \times n$ array of data for $L$ and $U$ overwrites the array $A$.

5. Basic form of the forward elimination (without pivoting):

For $k = 1, 2, \ldots, n - 1$

For $i = k + 1, k + 2, \ldots, n$

\[ a_{ik} \leftarrow -\frac{a_{ik}}{a_{kk}} \]

For $j = k + 1, k + 2, \ldots, n$

\[ a_{ij} \leftarrow a_{ij} + a_{ik}a_{kj} \]

This algorithm accesses entries of $A$ "by rows" (that is, it is a row-oriented algorithm): for example, when $i = k + 1$, the multiplier is stored in $a_{k+1,k}$ and the entries in row $k + 1$ are accessed in the order

\[ a_{k+1,k+1}, a_{k+1,k+2}, \ldots, a_{k+1,n}, \]

then $i$ is set to $k + 2$ and the entries of row $k + 2$ are accessed from left to right, and so on.

Instead, the entries of $A$ could be accessed "by columns" as in the following algorithm:

For $k = 1, 2, \ldots, n - 1$

For $i = k + 1, k + 2, \ldots, n$

\[ a_{ik} \leftarrow -\frac{a_{ik}}{a_{kk}} \]

For $j = k + 1, k + 2, \ldots, n$

For $i = k + 1, k + 2, \ldots, n$

\[ a_{ij} \leftarrow a_{ij} + a_{ik}a_{kj} \]

Note here that all of the multipliers for step $k$ of the algorithm are computed first and stored in
Then the entries in column $k + 1$ of $A$ are accessed and updated, then those in column $k + 2$, then those in column $k + 3$, and so on.

The reason for choosing between row and column oriented algorithms has do with how 2-dimensional arrays are stored in your computer -- by rows or by columns. Usually the choice of row versus column order for storage of arrays is dependent upon the programming language used.