Section 1.8: To summarize the discussion started at the end of the last lecture, it makes sense from a stability point of view to consider row swaps that would make the \( A(i, i) \) entry be the largest in its column. The result of such swaps can then be recorded in a permutation matrix \( P \), which can start out as the identity matrix, and whose rows get swapped in the same fashion as the rows of \( A \).

Remark 1. We define a permutation matrix \( P \) of size \( n \times n \) as a \( 0-1 \) matrix which has precisely one 1 in each row and column, and this means that such a matrix has an associated function \( i \to p(i) \) with \( (i, p(i)) \) being the entry which is 1 in row \( i \). Such a function is a bijection from \( \{1, 2, \ldots, n\} \) to itself, aka a permutation, and this is what gives the matrix its name. Also, for an \( n \times n \) matrix \( A \), \( PA \) is the matrix obtained from \( A \) by rearranging the rows of \( A \) in the order given by the permutation. See example below.

Example. 

\[
P = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad p(1) = 3, \quad p(2) = 1, \quad p(3) = 2;
\]

and if

\[
A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \quad \text{then } PA = \begin{bmatrix} 7 & 8 & 9 \\ 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}.
\]

To come back to the discussion of a stable Gaussian elimination or variant of the \( LU \) factorization, the output of this code would then be three-fold: \( [L, U, P] \), where \( P \) would be the permutation matrix encoding the row swaps that have been done, and so \( PA = LU \). (Alternately, this can be achieved by returning a vector that represents the permutation itself.)

To write this code, you will have to take the following ingredients. For each outer iteration \( i \),

- look down the column and find the maximum entry (in absolute value) and its index. This can be achieved by doing, at step \( i \),

\[
[m, index] = \max(\text{abs}(A(i : n, i))); \quad index = index + i - 1;
\]

as \( index = 2 \), for example, means the largest entry is in row \( i + 1 \).

- if \( i \neq \text{index} \), swap rows \( A(i,:) \) and \( A(\text{index}, :) \), but also \( L(i,1:(i-1)) \) and \( L(\text{index},1:(i-1)) \) (provided \( i \geq 2 \))

- record the swapping either in \( P \) or in the vector \( p \)

- continue with the rest of the code.
The amount of work thus added is \( O(n^3) \) assignments, so that does not change the \( 2/3n^3+O(n^2) \) flop count.

The above is called partial pivoting; even more accuracy can be obtained by doing total pivoting, where one searches for the pivot not only among rows but also among columns.

**A Special Case: Cholesky Decomposition.** We say that a matrix \( A \) is positive definite if it is symmetric and, for any \( x \) an \( n \times 1 \) real vector with \( x \neq 0 \), \( x^T A x > 0 \).

If \( A \) is positive definite, \( A \) is not only guaranteed to have an \( \text{LDV} \) factorization, but in addition it will also have \( V = L^T \) and \( D \) is a matrix with all diagonal entries positive.

To see this, note the following.

**Lemma 1.** If \( A \) is positive definite, \( A(1:k, 1:k) \) is positive definite for any \( k \).

**Proof.** For any \( y = (x_1, \ldots, x_k)^T \) with \( y \neq 0 \), take an \( n \times n \) vector \( x = (y^T, 0, \ldots, 0)^T \) to see that \( y^T A(1:k, 1:k) y = x^T A x > 0 \).

This means both that the \( \text{LU} \), and hence \( \text{LDV} \), factorizations exist, since all principal submatrices of \( A \) are non-singular, and also, since \( A = A^T \), that \( L = V^T \). With a little bit more work one may also show that all entries on the diagonal of \( D \) are positive. As such, one may choose \( \tilde{D} = \sqrt{D} \), the diagonal matrix having on the diagonal the square roots of the entries on the diagonal of \( D \), and write

\[
A = \text{LDV}^T = L\tilde{D} \tilde{D}^T = (L\tilde{D})(L\tilde{D})^T, 
\]

and take advantage of the fact that \( \tilde{D} = \tilde{D}^T \) as \( \tilde{D} \) is diagonal to define \( R = (L\tilde{D})^T \) and conclude that

**Theorem 1.** If \( A \) is positive definite, there exists an upper triangular matrix \( R \) such that

\[
A = R^T R. 
\]

This decomposition is unique, and it is called the **Cholesky Decomposition.**

While one could obtain \( R \) from \( A \) via the \( \text{LU} \) factorization, it is more advantageous to use the symmetry of the problem, and solve directly. We do this by noting that if we write

\[
R^T R = \begin{bmatrix} r_{11} & 0 & \ldots & 0 \\ r_{12} & r_{22} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ r_{1n} & r_{2n} & \ldots & r_{nn} \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & \ldots & r_{1n} \\ 0 & r_{22} & \ldots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & 0 & r_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{12} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \ldots & a_{nn} \end{bmatrix} = A, 
\]

then multiplying \( R^T R \) we can obtain

\[
r_{11}^2 = a_{11}, \\
r_{11}r_{1j} = a_{1j}, \quad \text{for all} \ j \geq 1, 
\]

and this gives formulas for all the entries in the first row of \( r \).

Going further, we note that for any \( i \),

\[
r_{1i}^2 + r_{2i}^2 + \ldots + r_{ii}^2 = a_{ii}, \\
r_{1i}r_{1j} + r_{2i}r_{2j} + \ldots + r_{(i-1)i}r_{(i-1)j} + r_{ii}r_{ij} = a_{ij}, \quad \text{for all} \ j \geq i + 1, 
\]
we see that we can solve recurrently for all entries of $R$, via the following code.

Note the usage of $\text{triu}(A,0)$; this MATLAB function extracts the upper triangular part of $A$.

```matlab
function R = cholesky_factor(A);
    n = size(A,1);
    R = triu(A,0);  % we initialize $R$ as the upper triangular part of $A$
    for i = 1 : n
        for k = 1 : (i - 1)
            R(i,i) = R(i,i) - R(k,i)^2;
        end
        if R(i,i) ≤ 0, error('A is not positive definite'), end
        R(i,i) = sqrt(R(i,i));
    end
    for j = (i + 1) : n
        for k = 1 : (i - 1)
            R(i,j) = R(i,j) - R(k,i) * R(k,j);
        end
        R(i,j) = R(i,j) / R(i,i);
    end
end
```

If we think of $\text{sqrt}$ as one operation, a quick calculation shows a flop count of

$$
\sum_{i=1}^{n} ((2i - 2) + 1 + (n - i) \cdot (2(i - 1) + 1)) = n^3 - 2n^3/3 + O(n^2) = n^3/3 + O(n^2) ;
$$

you will notice this is half the cost of $LU$ factorization.