Chapter 1.7: Gaussian Elimination and the LU factorization

We now depart a bit from the textbook’s progression, as it seems more natural to introduce Gaussian elimination before the Cholesky decomposition.

Chapter 1.7: Gaussian Elimination and the LU decomposition. We continue our review of methods for solving systems of linear equations with the first method you have encountered in Math 18 or thereabouts: Gaussian elimination.

Gaussian elimination for a linear system (also known as row-reduction to echelon form) is based on three types of elementary operations:

a) add a multiple of an equation (row) to another
b) swap two equations (rows)
c) multiply an equation (row) by a non-zero constant

The principle of Gaussian elimination is simple. Faced with an $n \times n$ system $Ax = b$, perform elementary operations to transform this system into an upper triangular system $Ux = b'$, which we can then solve via backward substitution.

Note: here and throughout the following two lectures, we will assume that the solution to the system exists and is unique, i.e., $A$ is invertible.

Proposition 1. Performing elementary operations does not change the set of solutions to the system.

Sketch of proof. Each operation is reversible, and so any solution of the final system is a solution of the initial one, and vice-versa. Of note: if $A$ is nonsingular, the upper triangular matrix obtained at the end of the algorithm is non-singular; elementary row operations of type a) have no effect on the determinant, type b) changes its sign, and type c) multiplies it by a non-zero constant. In either case the determinant stays non-zero.

To emphasize how Gaussian Elimination works, we work on the augmented matrix $\tilde{A} = [A|b]$ (which “mimics” the equations) as follows. We may zero out the second entry in the first column by subtracting the first row times $a_{21}/a_{11}$ from the second row:
We can continue this process, subtracting \( \frac{a_{j1}}{a_{11}} \) times row 1 from row \( j \) for \( j = 3, \ldots, n \), until we have “zeroed out” the 2nd through \( n \)th entries in the first column and \( A \) has been brought to the form \( U_1 \),

\[
U_1 = \begin{bmatrix}
a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\
0 & \hat{a}_{22} & \hat{a}_{23} & \cdots & \hat{a}_{2n} \\
\vdots & & & & \\
0 & \hat{\alpha}_{n2} & \hat{\alpha}_{n3} & \cdots & \hat{\alpha}_{nn}
\end{bmatrix},
\]

while we performed the same operations on \( b \).

Then we continue to zero out the lower part of the matrix, by now focusing on the second column and the entries \((3, 2)\) through \((n, 2)\), and using multiples of the second row, until the matrix becomes

\[
U_2 = \begin{bmatrix}
a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\
0 & \hat{a}_{22} & \hat{a}_{23} & \cdots & \hat{a}_{2n} \\
\vdots & & & & \\
0 & 0 & \hat{\alpha}_{n3} & \cdots & \hat{\alpha}_{nn}
\end{bmatrix},
\]

and on and on until the \((n-1)\)st column gets zeroed out under the diagonal.

Below is MATLAB code for this procedure.

```matlab
function x = ge_solve(A, b);
    n = size(A,1);
    for i = 1:n
        if A(i,i) == 0, error('cannot perform GE straightforwardly on this matrix'), end
        for j = (i+1):n
            l = A(j,i)/A(i,i);
            A(j,i) = 0;
            for k = (i+1):n
                A(j,k) = A(j,k) - l * A(i,k);
            end
            b(j) = b(j) - l * b(i);
        end
    end
    x = upptriangsolve(A, b);
end
```

Note that in the above, we call an upper triangular solve for the upper triangular system \( Ax = b \). This code you will have to write for HW 1, based on the lower triangular solve we wrote in class.

**LU factorization.** Suppose now that we rewrite the code we wrote before, to adapt it a little bit; in particular, suppose that we wanted to save the multipliers, rather than overwrite them each time; then \( l \) would have to become \( L(j,i) \) in the code above, and let us also initialize the new matrix \( L \) not with 0s, but as the identity matrix. Also, suppose we call \( U \) the final form of the matrix \( A \) (which, for convenience, we have simply rewritten before).

Then we will have actually found, in the course of the computation, a factorization of \( A \) in the form \( LU = A \).
Let us rewrite the code to compute this factorization (we will see later why this factorization is important). Note that this factorization does not depend on \( b \), which is why it does not need it as an input.

```matlab
function x = lu_factor(A);
n = size(A,1); L = eye(n);
for i = 1 : n
    if A(i,i) = 0, error('simple GE will not work on this matrix'), end
    for j = (i + 1) : n
        L(j,i) = A(j,i)/A(i,i);
        A(j,i) = 0;
        for k = (i + 1) : n
            A(j,k) = A(j,k) - L(j,i)*A(i,k);
        end
    end
end
U = A;
```

How might we know that \( LU = A \)? Let us examine the matrix \( L \):

\[
L = \begin{bmatrix}
1 & 0 & 0 & \ldots & 0 \\
0 & 1 & 0 & \ldots & 0 \\
\vdots & & & & \\
0 & 0 & \ldots & 1 & \ldots & 0 \\
0 & 0 & \ldots & l_{(i+1)i} & \ldots & 0 \\
\vdots & & & & & \\
0 & 0 & \ldots & l_{ni} & \ldots & 0
\end{bmatrix}
\]

Let now \( L_i \) be the lower triangular matrix with ones on the main diagonal, and the only other nonzero entries in the column \( i \), under the diagonal:

\[
L_i = \begin{bmatrix}
1 & 0 & \ldots & 0 & \ldots & 0 \\
0 & 1 & \ldots & 0 & \ldots & 0 \\
\vdots & & & & & \\
0 & 0 & \ldots & 1 & \ldots & 0 \\
0 & 0 & \ldots & l_{(i+1)i} & \ldots & 0 \\
\vdots & & & & & \\
0 & 0 & \ldots & l_{ni} & \ldots & 0
\end{bmatrix}
\]

for \( 1 \leq i \leq (n - 1) \).

If we denote by \( U_1 \) the matrix after the first iteration of the code has been completed (and zeros have been introduced in the first column, below the diagonal), \( U_i \) the matrix after the \( i \)th iteration has been completed (and now there are zeros under the diagonal in the 1st through \( i \)th columns), it is not too hard to see that

\[
A = L_1U_1 = L_1L_2U_2 = \ldots = L_1L_2\ldots L_{n-1}U_{n-1}.
\]

We will only show the first of the equalities above.

**Lemma 1.** \( A = L_1U_1 \).
Proof. Consider what multiplication by row 2 ≤ j ≤ n of L₁ does to U₁: it multiplies the first row of U₁ by l₁j and adds it to row j. Given that the first row of U₁ is the first row of A, this reverses the process through which we obtained U₁ from A! So A = L₁U₁.

We can employ the same kind of reasoning to show the rest of the equalities. What is less obvious, but not too hard to show, is that L₁L₂...Lₙ₋₁ = L. You will have to do this in HW 2 for n = 4.

Theorem 1. If A is non-singular (that is, invertible) and if all of its principal submatrices are non-singular, one may factor A = LU with L a “unit” lower triangular matrix (that is, it has 1s on the diagonal) and U an upper triangular matrix. Moreover, this factorization is unique.

Remark 1. The condition that all of the principal submatrices of A are non-singular is equivalent to saying that there will be no error message when the code ge_solve is called with A as an input (none of the A(i, i) will be 0).

Remark 2. As part of HW 2, you will have to show this uniqueness; some results you will need are the fact that the inverse of an upper triangular matrix is upper triangular, similarly, the inverse of a lower triangular matrix is lower triangular (which you can get by transposition).

Flop ("floating point operation") count. The third for loop performs 2 operations each time it is called, for a total 2(n − i) operations. To this we add another 1 inside the second for loop (the calculation of L(j, i)). The rest is an assignment that is not counted. We run the second for loop (n − i) times, for a total of (n − i) (2(n − i) + 1) operations, and we sum this over i with i from 1 through n:

\[ \sum_{i=1}^{n} (n - i)(2(n - i) + 1) = \sum_{i=1}^{n} 2(n - i)^2 + (n - i), \]

then we switch indexing to j = n − i to obtain

\[ \sum_{j=0}^{n-1} 2j^2 + j = 2 \frac{(n - 1)n(2n - 1)}{6} + \frac{(n - 1)n}{2} = O \left( \frac{2}{3} n^3 \right) + O(n^2). \]

The LDV factorization. If a matrix has a unique LU factorization, it has a unique LDV factorization, where D is a diagonal matrix, L is a unit lower triangular matrix, and V is a unit upper triangular matrix. This can be seen immediately since in this case DV = U and so D is the matrix whose diagonal consists of U(1, 1) through U(n, n), and V = D⁻¹U.