LECTURE 11

Gaussian Elimination

The LU factorization technique discussed in the preceding lecture seems quite different from the Gaussian elimination technique for solving systems of linear equations that is taught in linear algebra courses. The aim of this lecture is to show the connection between the two techniques and then develop a computer algorithm for carrying out Gaussian eliminiation.

So let me first remind you how Gaussian elimination works. Consider the following system of equations

$$\begin{aligned} x_1 + x_2 + x_3 &= 6 \\ 2x_1 - 2x_2 + x_3 &= 1 \\ x_1 + x_2 - x_3 &= 0 \end{aligned}$$

The matrix representation of this system is

$$\mathbf{Ax = b} \quad \Rightarrow \quad \begin{pmatrix} 1 & 1 & 1 \\ 2 & -2 & 1 \\ 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 6 \\ 1 \\ 0 \end{pmatrix}$$

Now there are several operations that one can perform on a system of equations, without changing its solution.

1. We can replace any equation by a non-zero constant times the original equation.
2. We can replace any equation by its sum with another equation.
3. We can carry out the above operations any number of times.

In the matrix language these operations equations translate to operations on the augmented matrix

$$[\mathbf{A}]\,[\mathbf{b}] = \begin{bmatrix} 1 & 1 & 1 \\ 2 & -2 & 1 \\ 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 6 \\ 1 \\ 0 \end{bmatrix}$$

Thus, for example we can add -2 times the first row to the second row and -1 times the first row to the third row to obtain the following equivalent system

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & -4 & -1 \\ 0 & 0 & -2 \end{bmatrix} \begin{bmatrix} 6 \\ -11 \\ -6 \end{bmatrix}$$

which, being an upper triangular system, is readily solved.

Example 11.1. Construct a Maple program for solving

$$\begin{pmatrix} 1 & 1 & 1 \\ 2 & -2 & 1 \\ 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 6 \\ 1 \\ 0 \end{pmatrix}$$

using Gaussian elimination.

The following code will do the trick.

```
n := 3:
n1 := n+1:
A := array(1..n,1..n,[[1,1,1], [2,-2,1],[1,1,-1]]):
b := array(1..n,[6,1,0]):

print(`A is`, A);
print(`b is`, b);

x := array(1..n):
U := array(1..n,1..n):
c := array(1..n):

# define augmented matrix
AugA := array(1..n,1..n1):

for i from 1 to n do
 for j from 1 to n do
 AugA[i,j] := A[i,j]:
 od:
 AugA[i,n1] := b[i]:
od:


# carry out Gaussian eliminiation
for k from 1 to n-1 do
 for j from k+1 to n do
 m := AugA[j,k]/AugA[k,k]:
 for i from k to n1 do
 AugA[j,i] := AugA[j,i] - m*AugA[k,i]:
 od:
 od:
od:

print(`AugA is`, AugA):

# we interprete AugA as representing an upper triangular
# system Ux=c and solve for x using back substitution

# identify the upper triangular matrix
 for i from 1 to n do
 for j from 1 to n do
 U[i,j] := AugA[i,j]:
 od:
 od:
# identify the column vector c
 for i from 1 to n do
 c[i] := AugA[i,n1]:
 od:

# carry out the back substitution
for i from 0 to n-1 do
 x[n-i] := (c[n-i] - add(U[n-i,n-j]*x[n-j],j=0..i-1))/U[n-i,n-i]:
od:
```

```
print('x is', x);
```

Note: In this example, I introduced the array variables `AugA`, `U`, and `c` only for the purpose of conceptual clarity. A more efficient program would only require using the variables `A`, `b`, and `x`.

1. Connection with LU Factorization

To make the connection with *LU* factorization, let's carry out another example of Gaussian elmination. Consider the following matrix. (Note: the operations below are applied only to the *unaugmented* matrix $\mathbf{A}$ of a linear system $\mathbf{Ax} = \mathbf{b}$.)

$$\mathbf{A} = \begin{bmatrix} 6 & -2 & 2 & 4 \\ 12 & -3 & 6 & 11 \\ -12 & 5 & 1 & -3 \\ 6 & 1 & 20 & 23 \end{bmatrix}$$

In the first stage of Gaussian elimination we leave the first row unchanged, muliply the first row by 2 and add it to the second row, multiply the first row by -2 and add it to the third row, and multiply the first row by 1 and add it to the fourth row; producing

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & 1 & 2 & 3 \\ 0 & 1 & 5 & 5 \\ 0 & 3 & 18 & 19 \end{bmatrix}$$

Let's associate with this stage a column vector

$$\mathbf{c}_1 = \begin{bmatrix} 1 \\ -2 \\ 2 \\ -1 \end{bmatrix}$$

Here the first non-zero component 1 indicates that we left the first row unchanged, the second component indicates that we multiplied the first row by -2 before subtracting it from the second row, the third component indicates that we multiplied the first row by 2 and subtracting it from the third row, and the fourth component indicates that we multiplied the first row by 1 and subtracted it from the fourth row.

In the next stage we ignore row 1, leave row 2 unchanged, and add multiples of row 2 to rows 3 and 4. This stage can thus be characterized by the following column vector

$$\mathbf{c}_2 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 3 \end{bmatrix}$$

and the subsequent matrix will be

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 2 & 3 \\ 0 & 0 & 12 & 10 \end{bmatrix}$$

In the next stage we ignore the first two rows, leave row 3 unchanged, and multiply row 3 by $\frac{2}{9}$ and add it to the last row. The corresponding column vector will be

$$\mathbf{c}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 4 \end{bmatrix}$$

and the resulting matrix will be

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

Let us write down one last column vector to indicate that there's nothing left to do

$$\mathbf{c}_4 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Now set $\mathbf{U}$ denote the upper triangular matrix that represents the result of the final stage of Gaussian elimination

$$\mathbf{U} = \begin{pmatrix} 6 & -2 & 2 & 4 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & 0 & 2 \end{pmatrix}$$

and let $\mathbf{L}$ the lower triangular matrix that's formed by adjoining the column vectors $\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3$ and $\mathbf{c}_4$:

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ -2 & 1 & 1 & 0 \\ 1 & 3 & 4 & 1 \end{pmatrix}$$

Then, surprise, surprise, these two matrices provide an LU factorization of the original matrix $\mathbf{A}$.

1.1. New and Improved Notation. In the preceding example, we saw that we could simultaneously identify an LU factorization of a matrix $\mathbf{A}$ if we conscientiously kept track of the multipliers of the pivot rows. We did this by associating a column vector to each stage of the Gaussian elimination procedure. What I now want to describe is a notational hat-trick that allows us to work only with the matrix $\mathbf{A}$. The idea is this we keep track of the multipliers by placing the multiplier for a given row, say the i^{th} row for the k^{th} stage of Gaussian elimination in the ki slot of the matrix. Of course, these new entries are not really components of the matrix $\mathbf{A}$, we can put them there though without losing information because after the k^{th} stage of Gaussian elimination, the matrix $\mathbf{A}$ will always have a zero in the ki slot. To make the special interpretation of these entries manifest we'll underline them. The example above would then work out as follows

$$\begin{pmatrix} 6 & -2 & 2 & 4 \\ 12 & -3 & 6 & 11 \\ -12 & 5 & 1 & -3 \\ 6 & 1 & 20 & 23 \end{pmatrix}$$

$$\begin{pmatrix} 6 & -2 & 2 & 4 \\ \underline{2} & 1 & 2 & 3 \\ \underline{-2} & 1 & 5 & 5 \\ \underline{1} & 3 & 18 & 19 \end{pmatrix}$$

$$\begin{pmatrix} 6 & -2 & 2 & 4 \\ \underline{2} & 1 & 2 & 3 \\ \underline{-2} & \underline{1} & 3 & 2 \\ \underline{1} & \underline{3} & 12 & 10 \end{pmatrix}$$

$$\begin{pmatrix} 6 & -2 & 2 & 4 \\ \underline{2} & 1 & 2 & 3 \\ \underline{-2} & \underline{1} & 3 & 2 \\ \underline{1} & \underline{3} & \underline{4} & 2 \end{pmatrix}$$

Now the LU factorization of $\mathbf{A}$ can be obtained by pulling off the underlined entries into a unit lower triangular matrix $\mathbf{L}$ and interpreting the entries that remain as the components of a upper triangular matrix $\mathbf{U}$.

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \underline{2} & 1 & 0 & 0 \\ \underline{-2} & \underline{1} & 1 & 0 \\ \underline{1} & \underline{3} & \underline{4} & 1 \end{pmatrix} \quad , \quad \mathbf{U} = \begin{pmatrix} 6 & -2 & 2 & 4 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & 0 & 2 \end{pmatrix}$$

Thus, another method of solution for a linear system of the form $\mathbf{Ax} = \mathbf{b}$ would be to carry out the technique above to find an LU factorization of the (unaugmented) matrix $\mathbf{A}$ and then solve $\mathbf{Lz} = \mathbf{b}$ for $\mathbf{z}$ and finally $\mathbf{Ux} = \mathbf{z}$.

EXAMPLE 11.2. Write down the Maple code that automates the preceding example; i.e., write down the Maple code that carrys out Gaussian elimination and an LU factorization simultaneously.

Here's the code.

```
n := 4;
A := array(1..n,1..n,[[6,-2,2,4],[12,-3,6,11], [-12,5,1,-3],[6,1,20,23]]);
L := array(1..n,1..n);
U := array(1..n,1..n);
LU := array(1..n,1..n);

print(`A is`, A);

# carry out the Gaussian elimination
for k from 1 to n-1 do
   for i from k+1 to n do
      # calculate the multiplier for the ith row at the kth stage
       m := A[i,k]/A[k,k];
      # store the multiplier
      A[i,k] := m;
      for j from k+1 to n do
         A[i,j] := A[i,j] - m*A[k,j];
      od:
   od:
od:
print(`new A is`,A);

#extract L and U from the new A
for i from 1 to n do
   for j from 1 to n do
```

```
      if i<j then
         U[i,j] := A[i,j];
         L[i,j] := 0;
      elif i=j then
         U[i,j] := A[i,j];
         L[i,j] := 1;
      else
         U[i,j] := 0;
         L[i,j] := A[i,j];
      fi;
   od:
od:

print(`L is`, L);
print(`U is`, U);

# verify that A = LU
for i from 1 to n do
   for j from 1 to n do
      LU[i,j] := add(L[i,k]*U[k,j],k=1..n);
   od:
od:

print(`LU is`, LU);
```