LU Decomposition Method

© 2006 Kevin Martin, Autar Kaw, Jamie Trahan

University of South Florida

United States of America

kaw@eng.usf.edu

NOTE: This worksheet demonstrates the use of Maple to illustrate LU Decomposition method, a technique used in solving a system of simultaneous linear equations.

Introduction

When solving multiple sets of simultaneous linear equations with the same coefficient matrix but different right hand sides, LU Decomposition is advantageous over other numerical methods in that it proves to be numerically more efficient in computational time than other techniques. In this worksheet, the reader can choose a system of equations and see how each step of LU decomposition method is conducted. To learn more about LU Decomposition method as well as the efficiency of its computational time click here.

LU Decomposition method is used to solve a set of simultaneous linear equations, [A] [X] = [C], where $[A]_{nxn}$ is a non-singular square coefficient matrix, $[X]_{nx1}$ is the solution vector, and $[C]_{nx1}$ is the right hand side array. When conducting LU decomposition method, one must first decompose the coefficient matrix $[A]_{nxn}$ into a lower triangular matrix $[L]_{nxn}$, and upper triangular matrix $[U]_{nxn}$. These two matrices can then be used to solve for the solution vector $[X]_{nx1}$ in the following sequence:

Recall that
[A] [X] = [C].
Knowing that
[A] = [L] [U]
then first solving with forward substitution
[L] [Z] = [C]
and then solving with back substitution
[U] [X] = [Z]
gives the solution vector [X].

A simulation of LU Decomposition method follows.

Section 1: Input Data

Below are the input parameters to begin the simulation. This is the only section that requires user input. Once the values are entered, Maple will calculate the solution vector [X].

Input Parameters:
n = number of equations
[A] = *nxn* coefficient matrix
[RHS] = *n*x1 right hand side array

```
> restart;
  n:=4;
  A:=Matrix([[12.,7.,3.,6.7],[1.,5.,1.,9.],[13.,12.,4.001,8],[5.6,
  3.,7.,1.003]]);
  RHS:=[22,7.,29.001,5.301];
```

$$n := 4$$

$$A := \begin{bmatrix} 12. & 7. & 3. & 6.7 \\ 1. & 5. & 1. & 9. \\ 13. & 12. & 4.001 & 8 \\ 5.6 & 3. & 7. & 1.003 \end{bmatrix}$$

$$RHS := [22, 7., 29.001, 5.301] \qquad (2.1)$$

Section 2: LU Decomposition Method

The following sections divide LU Decomposition method into 3 steps:
1.) Finding the LU decomposition of the coefficient matrix $[A]_{nxn}$
2.) Forward substitution
3.) Back substitution

2.1 LU Decomposition

How does one decompose a non-singular matrix [A], that is how do you find [L] and [U]?
The following procedure decomposes the coefficient matrix [A] into a lower triangular matrix [L] and upper triangular matrix [U], given [A] = [L][U].
• For [U], the elements of the matrix are exactly the same as the coefficient matrix one obtains at the end of forward elimination steps in Naïve Gauss Elimination.
• For [L], the matrix has 1 in its diagonal entries. The non-zero elements on the non-diagonal elements are multipliers that made the corresponding entries zero in the upper triangular matrix during forward elimination.

Parameter names:
n = number of equations
A = *nxn* coefficient matrix

```
> LUdecompose:=proc(n,A)
```

```
local k,i,multiplier,j,sum,AA,L,U:
#Defining the [L] and [U] matrices.
L:=Matrix(1..n,1..n);
U:=Matrix(1..n,1..n);

#Initializing diagonal of [L] to be unity.
for i from 1 by 1 to n do
   L[i,i]:=1.0;
end do:

#Dumping [A] matrix into a local matrix [AA] because input variables should not be placed
on LHS of an assignment in a procedure.
for i from 1 by 1 to n do
   for j from 1 by 1 to n do
      AA[i,j]:=A[i,j];
   end do:
end do:

#Conducting forward elimination steps of Naïve Gaussian Elimination to obtain [L] and [U]
matrices.
for k from 1 by 1 to n-1 do
   for i from (k+1) by 1 to n do
   #Computing multiplier values.
   multiplier:=AA[i,k]/AA[k,k]:
   #Putting multiplier in proper row and column of [L] matrix.
   L[i,k]:=multiplier:
      for j from (k+1) by 1 to n do
         #Eliminating (i-1) unknowns from the ith row to generate an upper triangular
matrix.
         AA[i,j]:=AA[i,j]-multiplier*AA[k,j]:
      end do:
   end do:
end do:

#Dumping the end of forward elimination coefficient matrix into the [U] matrix.
for i from 1 by 1 to n do
   for j from i by 1 to n do
      U[i,j]:=AA[i,j]:
   end do:
end do:

return(L,U):
```

```
end proc:
```

2.2 Forward Substitution

Now that the [L] and [U] matrices have been formed, the forward substitution step, [L] [Z] = [C], can be conducted, beginning with the first equation as it has only one unknown,

$$z_1 = \frac{c_1}{l_{1,\,1}}$$

$$z_1 = \frac{c_1}{l_{1,\,1}} \tag{3.3.1}$$

Subsequent steps of forward substitution can be represented by the following formula:

$$z_i = \frac{\left(c_i - Sum\left(l_{i,\,j} \cdot z_j, j = 1\,..i - 1\right)[i = 2\,..n]\right)}{l_{i,\,i}}$$

$$z_i = \frac{c_i - \left(\sum_{j=1}^{i-1} l_{i,\,j}\, z_j\right)_{i = 2\,..4}}{l_{i,\,i}} \tag{3.3.2}$$

The following procedure conducts forward substitution steps to solve for [Z].

Parameter names:
n = number of equations
[L]= *n*x*n* lower triangular matrix
[C] = *n*x1 right hand side array

```
> forward_substitution:=proc(n,L,C)
  local Z,i,sum,j:
  #Defining the [Z] vector.
  Z:=Array(1..n):
  #Solving for the first equation as it has only one unknown.
  Z[1]:=C[1]/L[1,1];
  #Solving for the remaining (n-1) unknowns using Equation (3.3.2).
  for i from 2 by 1 to n do
  sum:=0;
     #Generating the summation term in Equation (3.3.2).
     for j from 1 by 1 to i-1 do
        sum:=sum+L[i,j]*Z[j]:
     end do:
  Z[i]:=(C[i]-sum)/L[i,i];
  end do:
  return(Z):
  end proc:
```

2.3 Back Substitution

Now that [Z] has been calculated, it can be used in the back substitution step, [U] [X] = [Z], to solve for solution vector $[X]_{nx1}$, where $[U]_{nxn}$ is the upper triangular matrix calculated in Step 2.1, and $[Z]_{nx1}$ is the right hand side array.

Back substitution begins with the n^{th} equation as it has only one unknown.

$$x_n = \frac{z_n}{u_{n,n}}$$

$$x_4 = \frac{z_4}{u_{4,4}} \tag{3.4.1}$$

The remaining unknowns are solved for using the following formula:

$$x_i = \frac{\left(z_i - \left(Sum\left(u_{i,j} \cdot x_j, j = i+1..n\right)[i = (n-1)..1]\right)\right)}{u_{i,i}}$$

$$x_i = \frac{z_i - \left(\sum_{j=i+1}^{4} u_{i,j}\, x_j\right)_{i=3..1}}{u_{i,i}} \tag{3.4.2}$$

The following procedure solves for [X].

Parameter names:
n = number of equations
[U] = nxn upper triangular matrix
[Z] = nx1 solution vector from forward substitution step

```
> back_substitution:=proc(n,U,Z)
  local i,sum,j,X:
  #Defining the [X] vector.
  X:=Array(1..n);
  #Solving for the nth equation as it has only one unknown.
  X[n]:=Z[n]/U[n,n];
  #Solving for the remaining (n-1) unknowns working backwards from the (n-1)th equation to
  the first equation.
  for i from n-1 by -1 to 1 do
     #Initializing series sum to zero.
     sum:=0;
        #Calculating summation term in Equation (3.4.2).
```

```
            for j from i+1 by 1 to n do
               sum:=sum+U[i,j]*X[j];
            end do:
         #Using Equation (3.4.2) to calculate solution vector [X].
         X[i]:=(Z[i]-sum)/U[i,i];
      end do:
      return(X);
      end proc:
```

Section 3: Results

Below, Maple returns the lower triangular matrix, $[L]_{nxn}$, the upper triangular matrix, $[U]_{nxn}$, the intermediate solution vector $[Z]_{nx1}$, and the final solution vector $[X]_{nx1}$.

```
> LU:=LUdecompose(n,A):
  L:=LU[1];
  U:=LU[2];
  Z:=forward_substitution(n,L,RHS);
  X:=back_substitution(n,U,Z);
```

$$L := \begin{bmatrix} 1.0 & 0 & 0 & 0 \\ 0.08333333333 & 1.0 & 0 & 0 \\ 1.083333333 & 1.000000000 & 1.0 & 0 \\ 0.4666666667 & -0.06037735856 & 5645.277374 & 1.0 \end{bmatrix}$$

$$U := \begin{bmatrix} 12. & 7. & 3. & 6.7 \\ 0 & 4.416666667 & 0.7500000000 & 8.441666667 \\ 0 & 0 & 0.0010000010 & -7.699999998 \\ 0 & 0 & 0 & 43467.02179 \end{bmatrix}$$

$$Z := [22.00000000 \quad 5.166666667 \quad 0.001000000000 \quad -10.29899436]$$

$$X := [1.275252676 \quad 1.310260491 \quad -.8244226876 \quad -0.0002369381185] \qquad (4.1)$$

References

[1] Autar Kaw, *Holistic Numerical Methods Institute, http://numericalmethods.eng.usf.edu/mws,* See
Introduction to Systems of Equations
How does LU Decomposition method work?
Saving of computational time for finding inverse of a matrix using LU decomposition

Conclusion

Maple helped us apply our knowledge of LU Decomposition method to solve a system of *n* simultaneous linear equations.

Question 1: Solve the following set of simultaneous linear equations using LU decomposition method.

Matrix([[5, 6, 2.3, 6], [9, 2, 3.5, 7], [3.5, 6, 2, 3], [1.5, 2, 1.5, 6]]) *Vector*(4, *symbol* = *x*) =< 4 , 5, 6.7, 7.8 >

$$\begin{bmatrix} 5\,x_1 + 6\,x_2 + 2.3\,x_3 + 6\,x_4 \\ 9\,x_1 + 2\,x_2 + 3.5\,x_3 + 7\,x_4 \\ 3.5\,x_1 + 6\,x_2 + 2\,x_3 + 3\,x_4 \\ 1.5\,x_1 + 2\,x_2 + 1.5\,x_3 + 6\,x_4 \end{bmatrix} = \begin{bmatrix} 4 \\ 5 \\ 6.7 \\ 7.8 \end{bmatrix} \tag{6.1}$$

Question 2: Use LU decomposition repeatedly to find the inverse of

Matrix([[5, 6, 2.3, 6], [9, 2, 3.5, 7], [3.5, 6, 2, 3], [1.5, 2, 1.5, 6]])

$$\begin{bmatrix} 5 & 6 & 2.3 & 6 \\ 9 & 2 & 3.5 & 7 \\ 3.5 & 6 & 2 & 3 \\ 1.5 & 2 & 1.5 & 6 \end{bmatrix} \tag{6.2}$$

Question 3: Look at the [Z] matrix in LU decomposition step [L] [Z] = [C] of Question #1. Is it the same as the [RHS] matrix at the end of forward elimination steps in Naive Gauss Elimination method? If no, is this a coincidence?