Implementation of LU Decomposition and Linear Solver using Matlab

Nasser M. Abbasi

sometime in 2009 page compiled on July 1, 2015 at 11:43am

Contents

1 introduction

This is MATLAB implementation for LU decomposition, forward substitution, backward substitution, and linear system solver.

The functions written are:

1. `nma_LU.m.txt` LU decomposition with partial pivoting with threshold support.
2. `nma_ForwardSub.m.txt` solves $Ly = b$ for y
3. `nma_BackSub.m.txt` solves $Ux = y$ for x
4. `nma_LinearSolve.m.txt` driver to solve $Ax = b$ for x using calling sequence $1 \rightarrow 2 \rightarrow 3$

Partial pivoting (P matrix) was added to the LU decomposition function. In addition, the LU function accepts an additional argument which allows the user more control on row exchange.

Matlab lu() function does row exchange once it encounters a pivot larger than the current pivot. This is a good thing to always try to do. But sometimes if the difference between the pivots is small, a user might not want this feature. Hence I added a "threshold" second parameter to the nma_LU.m function to indicate how large a difference should exist for a row exchange to occur.

A row exchange will always occur if the current pivot is zero and a non-zero pivot exist to do the exchange.

To get the same exact behavior as Matlab lu() simply make this parameter zero.

Below are examples calling the nma_LU, nma_ForwardSub.m, nma_BackSub.m and nma_LinearSolve.m.

In each example below, the output is verified against Matlab own functions

2 examples

2.1 using nma_LU()

2.1.1 example 1

```
>> A=[1 1 2;2 -1 1;1 2 0]
A =
     1     1     2
     2    -1     1
     1     2     0

>> [L,U,P]=nma_LU(A,0)
L =
   1.00000000000000                   0                   0
   0.50000000000000   1.00000000000000                   0
   0.50000000000000   0.60000000000000   1.00000000000000

U =
   2.00000000000000  -1.00000000000000   1.00000000000000
                  0   2.50000000000000  -0.50000000000000
                  0                   0   1.80000000000000
P =
     0     1     0
     0     0     1
     1     0     0

>> [L,U,P]=lu(A)

L =

   1.00000000000000                   0                   0
   0.50000000000000   1.00000000000000                   0
   0.50000000000000   0.60000000000000   1.00000000000000

U =
   2.00000000000000  -1.00000000000000   1.00000000000000
                  0   2.50000000000000  -0.50000000000000
                  0                   0   1.80000000000000

P =
     0     1     0
     0     0     1
     1     0     0
>>
```

2.1.2 example 2

```
>> A=rand(4);
>> [L,U,P]=nma_LU(A,0)

L =
   1.00000000000000                   0                   0                   0
   0.01212703756687   1.00000000000000                   0                   0
   0.07119243718995   0.20742768803520   1.00000000000000                   0
   0.43394327408595   0.19377225100868   0.40879105345917   1.00000000000000

U =
   0.81316649730376   0.19872174266149   0.01527392702904   0.46599434167542
                  0   0.60138257315521   0.74660044907755   0.41299833684006
                  0                   0   0.11623493184110   0.32625386921423
                  0                   0                   0   0.51620218784594

P =
     0     0     1     0
     0     0     0     1
     1     0     0     0
     0     1     0     0

>> [L,U,P]=lu(A)
L =
   1.00000000000000                   0                   0                   0
   0.01212703756687   1.00000000000000                   0                   0
   0.07119243718995   0.20742768803520   1.00000000000000                   0
   0.43394327408595   0.19377225100868   0.40879105345917   1.00000000000000

U =
   0.81316649730376   0.19872174266149   0.01527392702904   0.46599434167542
                  0   0.60138257315521   0.74660044907755   0.41299833684006
                  0                   0   0.11623493184110   0.32625386921423
                  0                   0                   0   0.51620218784594

P =
     0     0     1     0
     0     0     0     1
     1     0     0     0
     0     1     0     0
>>
```

2.2 using nma_LinearSolve()

2.2.1 example 1

```
>> A=[1 1 2;2 -1 1;1 2 0]
A =
     1     1     2
     2    -1     1
     1     2     0

>> b=[1 2 1];

>> nma_LinearSolve(A,b)

ans =

     1
     0
     0

>> A\b(:)

ans =

     1
     0
     0
>>
```

2.2.2 example 2

```
>> A=rand(6);
>> b=rand(6,1);
>> nma_LinearSolve(A,b)

ans =

   0.59090034220622
  -0.56523444269280
   0.95687095978224
  -0.97248777153372
   1.00007995741472
   0.24035777097022

>> A\b(:)

ans =

   0.59090034220622
  -0.56523444269280
   0.95687095978223
  -0.97248777153372
   1.00007995741472
   0.24035777097022

>>
```

2.3 using nma_ForwardSub()

2.3.1 example 1

```
>> [L,U,P]=nma_LU(A,0);
>> nma_ForwardSub(L,b)

ans =

   0.83849604493808
   0.36727512318587
   0.12405626870025
  -0.14539724685973
   0.17813906538571
  -0.19809655526705
```

2.4 using nma_BackSub()

2.4.1 example 1

```
>> nma_BackSub(U,ans)

ans =

   0.29867870305809
  -0.84855613142087
   0.48347828223154
  -1.68311779577975
   1.49928530116874
   1.53825192677360
```