

Damian Franco

CS 375 – 001

Homework #6:

- 1) For number one, we were assigned to estimate the time needed to solve a general (full) system by Gaussian elimination (*LU-factorization*). Since we are solving for a general system, then we can use the following counts: $2n^3/3 + O(n^2)$ for GE and n^2 for backward substitution. We are given that the computer is solving for a 1000 variable matrix in the problem. First, let's take the backwards substitution time into account. It is given, it would take 0.5 seconds to do the backwards substitution which would be folded out into this equation because of our given count time: $(1000)^2 * 50$ (50 because of the $0.5 * 10^3$ completion time) which will be equivalent to the number of operations per second the computer can achieve. In this case it would be $5.0 * 10^7$ operations per second. The next step would be to solve the number of operations Gaussian Elimination will take. The equation for GE given to us helps us on this part and is the following: $2*(3000)^3 / 3$ which would be approximately equal to $1.8 * 10^{10}$ operations per second. To solve the runtime of this algorithm with these principles then the runtime would equal the number of operations per second to do GE divided by the number of operations per second while doing the backward substitution. Since both were found above, the values will be plugged in and would look something like $1.8 * 10^{10} / 5.0 * 10^7$ which would eventually come out to 360. Here we found that it would take approximately 360 seconds to solve a general system by Gaussian elimination (*LU-factorization*).

Scratch work:

Counts:

GE – $2n^3/3 + O(n^2)$

Backward substitution – n^2

Operation counts:

GE = $2*(3000)^3 / 3 = 1.8 * 10^{10}$

Backward substitution = $(1000)^2 * 50 = 5.0 * 10^7$

Approximate time:

= GE(operation count) / BS(operation count)

= $1.8 * 10^{10} / 5.0 * 10^7$

≈ **360 seconds.**

2) For this problem we had to solve the system below, by hand first, then solve it numerically through MATLAB. Both had to be solved by finding the LU -factorization.

a) For part (a) we were asked to solve the linear system by finding the LU -factorization and then solve each triangular matrix with the solves shows to us in the narrations and notes. I noticed that this is most likely going to be different from what MATLAB's lu function will output because the way that I solved the linear system was the way taught to us in class, and MATLAB has a different way to do this. First, I found the L and U matrices as the work shows below and then I solved those with the equation given to us for backward and forward substitution. On the next page, the by hand written work for part (a) is shown, which is solving the linear system by finding the LU -factorization and then by carrying out the two-step triangular solves:

2a) Linear system:

$$A = \begin{bmatrix} 3 & 1 & 2 \\ 6 & 3 & 4 \\ 3 & 1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 3 \end{bmatrix}$$

$$M_1 = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \quad M_1^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

$$M_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad M_2^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 6 & 0 & 1 \end{bmatrix} \quad M_3^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \leftarrow \text{Don't need this}$$

$$M_2 \cdot M_1 \cdot A \rightarrow \begin{bmatrix} 3 & 1 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{bmatrix} = U$$

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

$$U = \begin{bmatrix} 3 & 1 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{bmatrix} \quad L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 0 \\ 1 \\ 3 \end{bmatrix}$$

① Backward substitution (U)

$$x_3 = 3/3 = \underline{1}$$

$$x_2 = (b_2 - u_{23} \cdot x_3) / u_{22} = (1 - 0 \cdot 1) / 1 = 1/1 = \underline{1}$$

$$x_1 = (b_1 - u_{12} \cdot x_2 - u_{13} \cdot x_3) / u_{11} = (0 - 1 \cdot 1 - 2 \cdot 1) / 3 \\ = -3/3 = \underline{-1}$$

$$\boxed{x_3 = 1, x_2 = 1, x_1 = -1}$$

② Forward substitution (L)

$$x_1 = 0/1 = 0$$

$$x_2 = (b_2 - l_{21} \cdot x_1) / l_{22} = (1 - 2 \cdot 0) / 1 = 1/1 = \underline{1}$$

$$x_3 = (b_3 - l_{31} \cdot x_1 - l_{32} \cdot x_2) / l_{33} = (3 - 1 \cdot 0 - 0 \cdot 1) / 1 \\ = 3/1 = \underline{3}$$

$$\boxed{x_1 = 0, x_2 = 1, x_3 = 3}$$

- b) For part (b) we were assigned to solve the problem using the $PA = LU$ factorization method. In this part, I just set up two simple lines of code that would run MATLAB's `lu` function to solve for L , U , and p . I noticed that p was not in the correct matrix form, but it did list the 1's in the right indexes of the columns for the P permutation matrix. Since I knew that, I made a new matrix named P that would be the correct permutation matrix and then I set up $P*A$ and $L*U$. Now we can see the linear system being solved in the $PA = LU$ factorization method when printed out side by side in the second screenshot of the MATLAB console.

Source Code:

HW6_2.m

```
1      %  
2      % function: HW6_2  
3      % This function will find the  
4      % LU-factorization of the linear  
5      % system give to us in problem 2.  
6      %  
7      A = [3 1 2; 6 3 4; 3 1 5];  
8      [L, U, p] = lu(A, "vector")
```

MATLAB Ouputs:

```
Command Window  
K>> HW6_2  
  
L =  
  
    1.0000    0    0  
    0.5000    1.0000    0  
    0.5000    1.0000    1.0000  
  
U =  
  
    6.0000    3.0000    4.0000  
    0   -0.5000    0  
    0    0    3.0000  
  
p =  
  
    2    1    3  
  
fx K>> |
```

```
p =  
  
    0    1    0  
    1    0    0  
    0    0    1  
  
>> L*U  
  
ans =  
  
    6    3    4  
    3    1    2  
    3    1    5  
  
>> p*A  
  
ans =  
  
    6    3    4  
    3    1    2  
    3    1    5  
  
fx >> |
```

- c) The vector p returned in part (b) can be constructed by getting the elements within the p vector one at a time and then storing them as an index to the column they want to be inserted at. I noticed that the p vector elements state exactly where the 1 will appear in the same column in the permutation matrix P . Knowing that, after we store the elements of p , the next thing to do would be to note the column in where each of these elements are located and that will help place the 1 in the correct position and the correct column. For example, in this case we are given $p_{11} = 2$, $p_{12} = 1$, and $p_{13} = 3$, we notice that the column (j) is changing. In this case we would keep column j as the same for both p and P when storing elements, but the row i would be swapped with the element from the p list (ex. $i = p_{11} = 2$). At that new i and j index, that is where a new 1 should be added. It'll look something like the MATLAB code below.

MATLAB Code:

```
1     for i = 1:length(A)
2         col = i;
3         new = A(i);
4         for j = 1:length(A)
5             newVal = 1;
6             newA(j,col) = newVal;
7         end
8     end
```

3) For number 3, this problem involves solving tridiagonal linear systems. Each part is shown below.

a) For part (a), we were assigned to solve the linear system above with the correct parameters. The size of matrix T is based on the length of the input that the user gives. In this case, the user gives the input (1, 0, 0, 0, 1) which is of length 5. That means the size n will be assigned to 5 and the matrix dimension will be $n \times n$, in this case 5×5 . I used the `spdiags` function in MATLAB to set up the matrix according to the size n . It will then use the “solve system” function \ in MATLAB to solve for the x ’s and that gave me the output (1, 1, 1, 1, 1) which is exactly the expected output listed in the problem.

Source Code:

trisolvel.m

```
1      %
2      % This is trisolvel from the homework problem
3      % 3 that takes in a vector of outputs and
4      % solves for the linear equation  $Tx = b$ 
5      % where  $T$  is the above tridiagonal matrix
6      %
7      % function  $x = \text{trisolvel}(b)$ 
8      function  $x = \text{trisolvel}(b)$ 
9       $n = \text{length}(b);$ 
10      $T = \text{full}(\text{spdiags}(\text{bsxfun}(@\text{times}, \text{ones}(n,1), [-1 \ 2 \ -1]), [-1 \ 0 \ 1]), n, n));$ 
11      $T$ 
12      $x = T \backslash b;$ 
```

MATLAB Output:

Command Window

```
>>  $x = \text{trisolvel}(\text{olddb})$ 

 $T =$ 

     2    -1     0     0     0
    -1     2    -1     0     0
     0    -1     2    -1     0
     0     0    -1     2    -1
     0     0     0    -1     2

 $x =$ 

    1.0000
    1.0000
    1.0000
    1.0000
    1.0000

fx >>
```

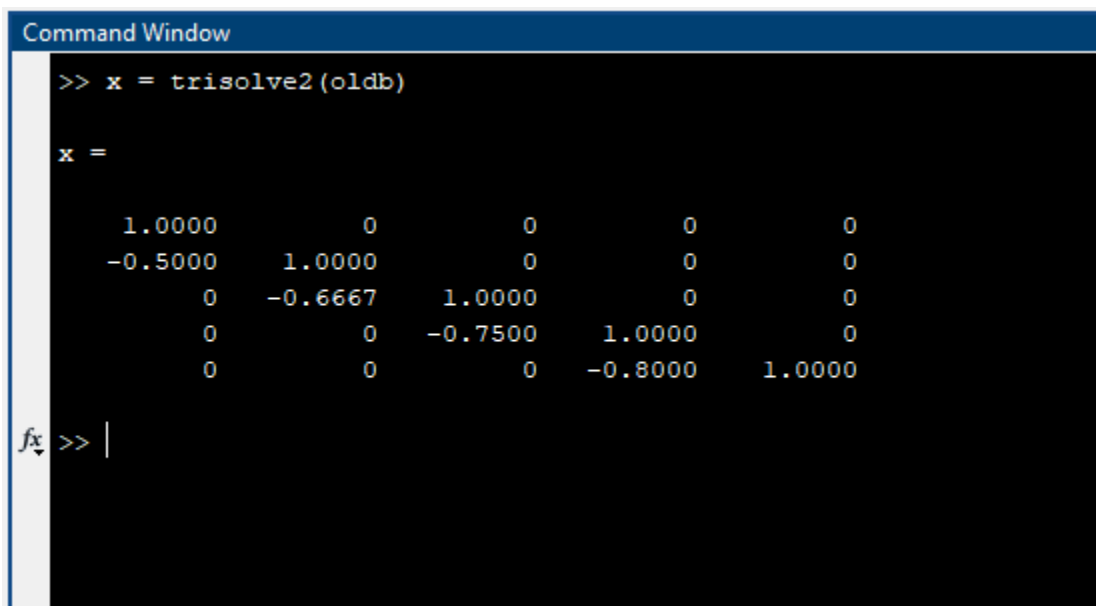
- b) For part (b) we are asked to show that a LU decomposition of T consists of two bi-diagonal matrices. This part was a little troubling for me because I was producing the correct output, but some random variable thrown in there in the under section, I used all the resources Professor Lau gave us to finish this question. First, I found the upper and lower bi-diagonal matrices and then I plugged them into the modified MATLAB functions that Professor Lau provided us in the module.

Source Code:

trisolv21.m

```
1      %
2      % This is trisolve2 from the homework problem
3      % 3 that takes in a vector of outputs and
4      % solves for the linear equation  $Tx = b$ 
5      % where  $T$  is the above tridiagonal matrix
6      %
7      % function  $x = \text{trisolve2}(b)$ 
8      function [L U] = trisolve2(b)
9      n = length(b);
10     T = full(spdia(b,ones(n,1),[-1 2 -1]),[-1 0 1],n,n));
11     [L U] = lu(T);
12     l = LBiDiSol(L,b);
13     u = UBiDiSol(U,b);
```

MATLAB Output:



The screenshot shows a MATLAB Command Window with the following text:

```
Command Window
>> x = trisolve2(olddb)

x =

    1.0000         0         0         0         0
   -0.5000    1.0000         0         0         0
         0   -0.6667    1.0000         0         0
         0         0   -0.7500    1.0000         0
         0         0         0   -0.8000    1.0000
```

At the bottom, there is a cursor and the text `fx >> |`.

- c) For part (c), there are a couple things to note here, I realize that my part (b) for this question is now wrong and void for this part so I will only be sharing screenshots and solution times for trisolve1 here. Otherwise, I notice as the n value increases, then the outputs and answers continuously get smaller. My interpretation of the results is that as you increase the n value, then the more precision errors occur and the run time continuously increases.

$n = 200$:

```
Elapsed time is 0.007406 seconds.  
fx >> |
```

```
Command Window  
  
>> HW6_3c  
  
x =  
  
1.0e+03 *  
  
0.0502  
0.1003  
0.1502  
0.1997  
0.2483  
0.2965  
0.3443  
0.3919  
0.4385  
0.4847  
0.5301  
0.5748  
0.6192  
0.6627  
0.7055  
0.7477  
0.7891  
0.8297  
0.8698  
0.9096  
0.9488  
0.9870  
1.0246  
1.0618  
1.0983  
1.1338  
1.1688  
1.2032  
fx
```


$n = 400$:

```
fx Elapsed time is 0.010809 seconds.  
>>
```

```
Command Window  
  
>> HW6_3c  
  
x =  
  
1.0e+04 *  
  
0.0105  
0.0209  
0.0312  
0.0415  
0.0518  
0.0621  
0.0723  
0.0825  
0.0926  
0.1027  
0.1128  
0.1228  
0.1327  
0.1427  
0.1525  
0.1623  
0.1721  
0.1818  
0.1915  
0.2012  
0.2108  
0.2205  
0.2300  
0.2394  
0.2488  
0.2581  
0.2673  
0.2766  
fx
```

$n = 800$:

```
Elapsed time is 0.023638 seconds.  
fx >> |
```

```
Command Window  
  
>> HW6_3c  
  
x =  
  
1.0e+04 *  
  
0.0204  
0.0407  
0.0609  
0.0812  
0.1014  
0.1215  
0.1416  
0.1616  
0.1816  
0.2015  
0.2213  
0.2411  
0.2608  
0.2805  
0.3001  
0.3197  
0.3392  
0.3587  
0.3781  
0.3974  
0.4168  
0.4361  
0.4553  
0.4745  
fx 0.4936  
0.5128
```

$n = 1600$:

```
fx Elapsed time is 0.105108 seconds.  
>> |
```

```
Command Window  
  
>> HW6_3c  
  
x =  
  
1.0e+05 *  
  
0.0041  
0.0082  
0.0123  
0.0164  
0.0205  
0.0246  
0.0287  
0.0327  
0.0368  
0.0408  
0.0449  
0.0489  
0.0530  
0.0570  
0.0610  
0.0650  
0.0690  
0.0730  
0.0770  
0.0810  
0.0850  
0.0890  
0.0930  
0.0969  
0.1009  
fx
```

$n = 3200$:

```
Elapsed time is 0.596379 seconds.  
fx >> |
```

```
Command Window  
  
>> HW6_3c  
  
x =  
  
1.0e+05 *  
  
0.0079  
0.0157  
0.0236  
0.0315  
0.0393  
0.0472  
0.0550  
0.0629  
0.0707  
0.0785  
0.0864  
0.0942  
0.1020  
0.1098  
0.1176  
0.1254  
0.1332  
0.1410  
0.1488  
0.1566  
0.1643  
0.1721  
0.1798  
0.1876  
fx 0.1953  
0.2031
```

$n = 6400$:

```
Elapsed time is 3.623586 seconds.  
fx >> |
```

```
Command Window  
0.0947  
0.0932  
0.0916  
0.0900  
0.0885  
0.0869  
0.0853  
0.0838  
0.0822  
0.0806  
0.0791  
0.0775  
0.0759  
0.0743  
0.0728  
0.0712  
0.0696  
0.0681  
0.0665  
0.0649  
0.0633  
0.0618  
0.0602  
0.0586  
0.0570  
0.0555  
0.0539  
0.0523  
0.0507  
0.0492  
0.0476  
fx 0.0460
```

$n = 12800$:

```
fx Elapsed time is 23.956032 seconds.  
>> |
```

```
Command Window  
0.0092  
0.0089  
0.0086  
0.0083  
0.0079  
0.0076  
0.0073  
0.0070  
0.0067  
0.0064  
0.0060  
0.0057  
0.0054  
0.0051  
0.0048  
0.0045  
0.0041  
0.0038  
0.0035  
0.0032  
0.0029  
0.0025  
0.0022  
0.0019  
0.0016  
0.0013  
0.0010  
0.0006  
0.0003
```