

LAB 2: Row Reduction and $A = LU$ Matrix Factorization

In this lab you will use MATLAB to obtain the $A = LU$ factorization of an invertible square matrix (L unit lower triangular, U upper triangular). This factorization expresses the row reduction algorithm in matrix form. From this factorization it is easy (and fast) to solve the system of linear equations $A\mathbf{x} = \mathbf{b}$ (where \mathbf{b} is a given vector and \mathbf{x} is an unknown vector): first solve the triangular system $L\mathbf{c} = \mathbf{b}$ for \mathbf{c} . Then solve the triangular system $U\mathbf{x} = \mathbf{c}$ for \mathbf{x} . Furthermore, once A is given, the same matrices L and U can be used for any other \mathbf{b} .

Reading from Textbook: Before beginning the Lab, read through Sections 1.6 and 1.7 of the text and work the suggested problems for these sections.

MATLAB Help: In Lab 1 you learned the basic MATLAB commands. Remember that every MATLAB command is documented in a `help` file, which you can access during a MATLAB session. For example typing `help format` gives information about the command `format`. Go to the mathematics department web site for this course for additional MATLAB documents if you want further information. For this lab, create a `diary` file and edit it as you did for Lab 1.

Script Files: For more complicated MATLAB calculations you should use *scripts*. A script contains one or several MATLAB commands and is stored as a text file with a descriptive name such as `mymatrix.m`, for example (the extension `.m` is required). When you type the name of a script (without the extension `.m`) at the MATLAB command prompt the commands within the script are executed, affecting the variables in the global workspace. Such scripts are called *m-files*. The advantage of having scripts is that you can execute the commands in the script at any time by typing the *name* of the script instead of the *contents* of the script.

Writing Scripts: When you need to write a script file for this lab and subsequent labs, use the following procedure: Start MATLAB and click on *File*, then *New*. Move the pointer to the right and click on *m-Files*. This will open the MATLAB Editor/Debugger Window, and you can type the script commands in this window. You can take any m-file, edit it (just as you would edit any text file), and then save it under a different name to obtain a new m-file.

Running Scripts: After you have created an m-file and saved it to your diskette, you must set the *Path* so that MATLAB can find this file on your diskette. Click on *File* and then *Set Path* and follow the directions to add your directory to the list of path names. If you are running MATLAB on a computer system where you can save your work in your own permanent directory on a hard disk, use that directory instead of `a:` in your path name.

Lab Write-up: You should open a diary file at the beginning of each MATLAB session (see Lab 1 for details). Begin the diary file with the comment line

```
% Math 250 MATLAB Lab Assignment #2
```

Type `format compact` so that your diary file will not have unnecessary spaces. Put labels to mark the beginning of your work on each part of each question. For example,

```
% Question 1 (a) ...
:
% Question 1 (b) ...
```

and so on.

Be sure to answer all the questions in the lab assignment. Insert comments in your diary file as you work through the assignment.

Random Seed: When you start your MATLAB session, initialize the random number generator by typing

```
rand('seed', abcd)
```

where $abcd$ are the last four digits of your Student ID number. This will ensure that you generate your own particular random vectors and matrices.

BE SURE TO INCLUDE THIS LINE IN YOUR LAB WRITE-UP

The lab report that you hand in must be your own work. The following problems all use randomly generated matrices and vectors, so the matrices and vectors in your lab report will not be the same as those of other students doing the lab. Sharing of lab report files is not allowed in this course.

Question 1. LU Factorization

In this problem you will use MATLAB to obtain the matrix factorization $A = LU$ for a square 3×3 matrix A .

- (a) Use the MATLAB editor to create an m-file called `newmat.m` with the following commands

```
A = fix(10*rand(3))
U = A
```

Type `newmat` at the MATLAB prompt. You should get a 3×3 integer matrix A with random entries between 0 and 9, and the initial value of U (at the end of the LU algorithm U will be upper triangular).

- (b) Use the MATLAB editor to create an m-file called `col1.m` with the following MATLAB commands:

```
L1 = eye(3);
L1(2,:) = L1(2,:) + (U(2,1)/U(1,1))*L1(1,:);
L1(3,:) = L1(3,:) + (U(3,1)/U(1,1))*L1(1,:);
L1
```

Here U is the matrix generated by `newmat.m` (notice the use of `;` to suppress screen output of the intermediate results). Execute this file by typing `col1` at the MATLAB prompt. If you get an error message, go back to step (a) and generate another A . Describe in words how the matrix L_1 is obtained from row operations on the 3×3 identity matrix. (remember that $U = A$ at the beginning of the algorithm). Use MATLAB to calculate L_1^{-1} . Describe in words how the matrix entries of L_1^{-1} are obtained from those of L_1 .

Now use MATLAB to replace the current value of the matrix U by the new value $L_1^{-1} * U$ (remember that the command $X = Y$ in MATLAB means to replace the current value of the variable X by the current value of the variable Y). Describe in words how the new value of U is obtained from the old value of U by row operations.

- (c) Use the MATLAB editor to create an m-file called `col2.m` with the commands

```
L2 = eye(3);
L2(3,:) = L2(3,:)+ (U(3,2)/U(2,2))*L2(2,:); L2
```

This will be used with the matrix U modified as in (b). Execute this file by typing `col2` at the MATLAB prompt. If you get an error message, go back to step (a) and generate another A . Use MATLAB to calculate L_2^{-1} . Describe in words how the matrix entries of L_2^{-1} are obtained from those of L_2 .

Now use MATLAB to replace the current value of the matrix U by the new value $L_2^{-1} * U$. Describe in words how the new value of U is obtained from the old value of U by row operations.

(d) Use MATLAB to get $L = L_1 * L_2$. Describe in words how the entries of L are obtained from those of L_1 and L_2 . Show *by a hand calculation* that for any matrices with the pattern of 1's and 0s in L_1 and L_2 , the matrix product $L_1 L_2$ can be written down without using any multiplication of numbers.

(e) Verify by MATLAB that $A = L * U$ (where U has the value from (c)). Show *by hand calculation* (without MATLAB) that $A = L * U$. Use the definitions of U and L in terms of A , L_1 and L_2 to do this (don't use numerical calculations of individual matrix entries).

Question 2. Using LU Factorization to Solve $Ax = b$

(a) **Inverting A , L and U :** If M is a square matrix that has an inverse, then the MATLAB command `inv(M)` will calculate the inverse matrix M^{-1} . Use this on the matrices L and U that you obtained in Question 1. Which entries in `inv(L)` and `inv(U)` are certain to be zero? Which entries in `inv(L)` are certain to be 1? For the matrices L_1 and L_2 , you found that the inverse matrices are simply obtained by putting a minus sign in front of the entries below the main diagonal. Does this method work to give the inverse matrix to L ?

(b) **Generating a Random Vector:** Use the text editor in MATLAB to create an m-file called `newb.m` with the command

```
b = fix(10*rand(3,1))
```

After you have saved this file, test the file by clicking on the MATLAB window and typing `newb` at the prompt. You should get a three-component column vector (3×1 matrix) \mathbf{b} with entries that are (random) integers between 0 and 9.

(c) **Solving $Ax = b$ using L and U :** Calculate the solution

```
c = inv(L)*b
```

to the triangular system $Lc = b$. Then calculate the solution

```
x = inv(U)*c
```

to the triangular system $Ux = c$. Now calculate Ax and check that it is \mathbf{b} (since the entries in \mathbf{b} are integers, this should be obvious by inspection).

Question 3. LU vs. RREF for solving $Ax = b$

In this question you will compare the speed and accuracy of two methods of solving the equation $Ax = b$ when A is an invertible square matrix. You will measure the execution time of MATLAB commands using the internal computer clock and the MATLAB function `etime`. Clear your workspace before beginning this question.

Before starting work on this question, type `rrefmovie` at the MATLAB prompt. You will see a step-by-step example of the row operations that transform a matrix A into `rref(A)`. Each pivot is chosen to be the *largest* in its column (for numerical stability), so extra row interchanges are used. Since `rref(A)` is *uniquely* determined by A , this does not affect the final answer. (Do not include this part in your lab write-up.)

(a) Create an m-file called `newAb.m` with the commands

```
A = round(10*rand(20)) ; b = round(10*rand(20,1));
```

(note the use of `;` to suppress the display of A and b). Use this m-file to generate a 20×20 random matrix A and a 20×1 random vector b with single-digit integer entries.

(b) You can solve $Ax = b$ by using RREF (Gauss-Jordan reduction). Use the MATLAB command `U = rref([A b])`. The last column y of U is then the solution to the system (because `rref(A)` is the identity matrix if A is a generic square matrix). Write an m-file called `rrefmeth.m` with the commands

```
t0 = clock; U = rref([A b]); y = U(:,21), rreftime = etime(clock, t0)
```

Now type `rrefmeth` to find the solution y and the calculation time using RREF.

(c) You can also solve $Ax = b$ by using the `\` operation $x = A \setminus b$ (this is carried out in MATLAB by using the LU method, with row interchanges if necessary). Write an m-file called `lumeth.m` with the commands

```
t0 = clock; x = A\b, ltime = etime(clock, t0)
```

Now type `lumeth` to find the solution x and the calculation time using the LU method.

(d) Compare the methods (b) and (c) for speed. Which method of solution was faster? What was the *ratio* of computation times for the two methods?

(e) The solutions x and y obtained from the two methods look to be the same, but if you calculate $x - y$ you will find that the solutions differ. The length $\|z\|$ of a vector z is calculated using the MATLAB function `norm(z)`. Use this to calculate the *relative size* $\|x - y\|/\|x\|$ of the difference between x and y .

Because the computer arithmetic uses only a fixed number of digits, neither x nor y is an *exact* solution to the system of equations. This means the *residual vectors* $b - A * x$ and $b - A * y$ are not zero. Calculate the *ratio* of the norms of the residual vectors for the two methods. Using this ratio as a comparison, decide whether LU or RREF is more accurate.

Optional Extra-Credit Question: Wavelets and Data Compression

Read **Section 2.5 Introduction to Wavelets** in the text. The following problem is very similar to Exercises 2.5 #1 and Supplementary Exercise # 5 on page 139, so you can work these first by hand and refer to the answers in the back of the book for additional help. Then work the following using MATLAB.

(a) Let $\mathbf{y} = [2 \ 3 \ 2 \ 4]^T$ represent a sample of a function f at the equispaced points $x_1 = -4$, $x_2 = -2$, $x_3 = 0$, $x_4 = 2$. Let A_1 and A_2 be the 4×4 wavelet averaging matrices (see Exercises 2.5 #6). Use MATLAB to calculate the wavelet vector $\mathbf{w} = A_2 A_1 \mathbf{y}$ that contains the final average and detail coefficients of the data. (Note that \mathbf{y} must be entered into MATLAB as a column vector. For this either separate the entries by `;` or else type `[2 3 2 4]'` at the prompt.) Verify by hand calculation (or MATLAB) that the first entry in \mathbf{w} is the average of all the entries in \mathbf{y} .

(b) Use the compression threshold $\epsilon = 1$ on \mathbf{w} to obtain the *compressed* wavelet vector \mathbf{u} (every entry in \mathbf{w} smaller than 1 in absolute value is replaced by zero to get \mathbf{u}). What percentage of the entries of \mathbf{u} are *not* zero? (This is a measure of the data compression that has been achieved.) Then compute the *compressed* data vector $\mathbf{z} = (A_2 A_1)^{-1} \mathbf{u}$.

(c) On the same set of coordinate axes graph the original data points \mathbf{y} and the points given by the compressed data vector \mathbf{z} , connecting successive points by straight line segments. You can do this in MATLAB by creating a column vector \mathbf{x} whose entries are the values x_1, x_2, x_3, x_4 . Then the MATLAB commands

```
plot(x, y), hold
plot(x, z)
```

will plot the original data and the compressed data on the same axes. You should be able to print the graph (if this is not possible on your computer system, then copy the screen plot by hand to graph paper).

Final Editing of Lab Write-up:

After you have worked through all the parts of the lab assignment, edit your diary file. Include the MATLAB calculations, but remove errors that you made in entering commands and remove other material that is not directly related to the questions, as you did for Lab 1.