

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 2.3, 2.5, 2.6, and 2.7 of the text and work the suggested problems for each section.

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 rank` gives information about the command `rank`. 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. Type the script commands in this window (insert comment lines beginning with `%` that describe the purpose of the commands in the script file). Then click on *File* in the Editor/Debugger toolbar to save your script on your diskette in drive `a:`. 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 the path browser icon in the MATLAB tool bar
- Click on *Browse* to change the current directory to `a:`
- Click on *Path* to add this 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. After you have worked through all the parts of a lab, you will need to edit your diary file, as you did for the previous Labs.

Remove all errors and other material that is not directly related to the questions. For example, remove the material associated with writing and testing your script files.

Preview the document before printing and remove unnecessary page breaks and blank space. Put your name, section number, and student ID number on each page. (If you have difficulty doing this using your text editor, you can write this information by hand after printing the report.)

Important: An unedited diary file without comments will get a GRADE OF ZERO as a lab writeup.

Question 1. Row Reduction without Row Exchanges

We begin by using MATLAB to carry out row reduction on a random 3×3 matrix A .

(a) Generating a Random Square Matrix: Use the text editor in MATLAB to create an m-file called `newA.m` with the command

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

After you have saved this file and set the *Path* (as described above), test the file by clicking on the MATLAB window and typing `newA` at the prompt (note the capitalized A ; remember that MATLAB is case-sensitive). You should get a 3×3 matrix A with entries that are (random) integers between 0 and 9. You can toggle between the MATLAB window and Editor/Debugger window to modify your script if necessary. You will find it convenient to resize the two windows so that they are displayed side by side.

Use the up arrow key \uparrow to generate several matrices A . Notice that for most of the matrices, the first pivot $A(1,1)$ is *not* zero.

(b) Reducing Column 1: Write a script file with the following commands.

```
B(1,:) = A(1,:);
B(2,:) = A(2,:) - A(1,:)*A(2,1)/A(1,1);
B(3,:) = A(3,:) - A(1,:)*A(3,1)/A(1,1)
```

(Note the semi-colons at the ends of the first two lines; remember that these suppress the display of the intermediate calculations, so that only the final matrix B is displayed.) Save this file under the name `col1.m`. Now type `newA` at the MATLAB prompt to generate a random matrix A . Then type `col1` to obtain the 3×3 matrix B . Repeat this several times, getting a different initial random matrix A and final matrix B each time. Can you ever get an error message when you run the file `col1`? Insert a comment line to explain. If you do get an error message, just generate another matrix A and type `col1` again.

Insert comment lines that describe in words how B is obtained from A by row operations. Which entries of B are *always* zero?

(c) Reducing Column 2: Write a script file with the following commands.

```
U(1,:) = B(1,:);
U(2,:) = B(2,:);
U(3,:) = B(3,:) - B(2,:)*B(3,2)/B(2,2)
```

Save this file under the name `col2.m`. Now type `newA` at the MATLAB prompt to generate a random matrix A . Then type `col1` to obtain the 3×3 matrix B , and type `col2` to obtain the 3×3 matrix U . Repeat this several times, getting a different initial random matrix A and final matrix U each time. Can you ever get an error message when you run the files `col1` followed by `col2`? Explain. If you do get an error message, just generate another matrix A and type `col1` and `col2` again.

Insert comment lines that describe in words how U is obtained from B by row operations. Which entries of U are *always* zero?

Question 2. $A = LU$ Factorization

Now we express the row reduction process in Question 1 in terms of the matrix factorization $A = LU$.

(a) Generating L_1 : Write a script file with the following commands.

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

Save this file under the name `low1.m`. Now type `newA` to generate a random matrix A . Then type `low1` to obtain the 3×3 matrix L_1 . Repeat this several times, getting a different initial random matrix A and final matrix L_1 each time. Can you ever get an error message when you run the file `low1`? Explain.

Insert comment lines that describe in words how L_1 is obtained from the identity matrix I by row operations. Which entries of L_1 are *always* 0? Which entries of L_1 are *always* 1? Use MATLAB to check that $L_1 * B$ is the matrix A (since A has integer entries, this is easy by inspection). Verify this for several random matrices A and the associated matrices B , as in Question 1. Then explain, in terms of row operations, why $L_1 * B = A$.

(b) Generating L_2 : Write a script file with the following commands.

```
I = eye(3);
L2(1,:) = I(1,:);
```

```
L2(2,:) = I(2,:);
L2(3,:) = I(3,:) + I(2,:)*B(3,2)/B(2,2)
```

Save this file under the name `low2.m`. Now type `newA` to generate a random matrix A . Then type `col1` to obtain the 3×3 matrix B , and `low2` to obtain the 3×3 matrix L_2 . Repeat this several times, getting a different initial random matrix A and final matrix L_2 each time. Can you ever get an error message when you run the file `low2`? Explain.

Insert comment lines that describe in words how L_2 is obtained from the identity matrix I by row operations. Which entries of L_2 are *always* 0? Which entries of L_2 are *always* 1? Use MATLAB to check that $L_2 * U - B = 0$. Verify this for several random matrices A and the associated matrices B and U , as in Question 1. Then explain, in terms of row operations, why $L_2 * U = B$.

(c) $A = LU$ Factorization: Set $L = L_1 * L_2$. Use MATLAB to check that $A = L * U$ (since A has integer entries, this will be obvious by inspection). Do this for several random matrices A . Which entries of L are *always* 0? Which entries of L are *always* 1?

Use the results of parts (a) and (b) of this question to show from the properties of matrix multiplication why $A = L * U$ for any A .

Question 3. 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. Do this with M each of the matrices A , L , and U from Question 2. In each case, also calculate `inv(M)*M` and `M*inv(M)`. These should be the 3×3 identity matrix I .

Which entries in `inv(L)` and `inv(U)` are *always* zero? Which entries in `inv(L)` are *always* 1?

(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 $Ax = b$: Generate a random vector \mathbf{b} as in (a). Calculate the solution

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

to the triangular system $L\mathbf{c} = \mathbf{b}$. Then calculate the solution

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

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

More about MATLAB:

Tcodes: A special set of *m-files* called *Tcodes* has been written to accompany Strang's textbook. To obtain any of these files, use a web browser (such as Netscape) and go to the Math Department Home page <http://www.math.rutgers.edu>. Click on *course materials*, then on *Math 250 Introduction to Linear Algebra*, and then on *MATLAB Teaching Codes*. You will see a directory of the m-files. Click on the particular m-file that you need. Then in the menu bar click on *Files* and *Save As*. Fill in the directory information that is requested.

slu.m and slv.m: To complete Lab 2 you will need the m-files `slu.m` and `slv.m`. To copy these m-files onto your diskette in drive `a:`, for example, give the directory information as `a:\slu.m` (do not change the name of the file). Repeat for `slv.m`.

(Optional): If you plan to do the optional extra-credit question, you should also copy the files `splu.m` and `splv.m` now.

Question 4. LU Factorization and $Ax = b$ (General Case)

(a) slu.m file: This is a *function* file (see page 82 of Strang's book). It takes as an input a square matrix A (of any size), which must be already defined in the current MATLAB workspace. To use it, type

```
[L, U] = slu(A)
```

MATLAB then calculates L and U so that $A = L * U$ (if the factorization exists; otherwise it gives an error message). Check that $A = L * U$. Check that the matrix U you have just obtained is the same as the matrix U calculated in Question 2 by typing `col1` and `col2`.

(b) Generating Random A and b : Use the text editor in MATLAB to create an m-file called `newAb.m` with the commands

```
A = fix(10*rand(6))
b = fix(10*rand(6,1))
```

After you have saved this file, test the file by clicking on the MATLAB window and typing `newAb` at the prompt. You should get a 6×6 matrix A and a six-component column vector (6×1 matrix) b with entries that are (random) integers between 0 and 9.

(c) Solving $Ax = b$ by `slv`: The m-file `slv.m` is a *function* file. It takes as inputs a square matrix A (of any size $n \times n$) and a column vector b of size $n \times 1$. It then calculates the $A = L * U$ factorization using `slu(A)`, and solves $Ax = b$ by solving $Lc = b$ and $Ux = c$ by back substitution (see page 83 of Strang's book).

Generate a random 6×6 matrix A and random vector b as in part (b). Then type

```
x = slv(A,b)
```

If you get an error message, generate another random A and b and repeat. Now check that $A*x = b$.

(d) Computational Cost of LU factorization: One arithmetic operation (addition, subtraction, multiplication or division of a pair of number) is called a *flop* (floating point operation). MATLAB has an internal variable called `flops` that counts the number of flops that are performed in a calculation.

Create the following m-file and save it under the name `luflops.m`.

```
A = rand(n);
flops(0);
slu(A);
flops
```

To use this script for a matrix of size 8×8 , for example, type

```
n = 8; luflops
```

at the MATLAB prompt. MATLAB will generate a random 8×8 matrix A , set the flop count to zero, calculate the LU factorization of A , and display the number of flops. (Be sure to include the semicolons at the end of all but the last line; otherwise your screen and diary file will be filled with unnecessary pages of output that you must remove before printing.)

For a random $n \times n$ matrix A , the number $f(n)$ of flops to calculate the LU factorization of A is approximately $(2/3)n^3$ (see Strang, p. 81 for a discussion). Run the file `luflops` for $n = 16, 32, 64$ and 128 and compare the actual flop count to the predicted flop count.

If you double the size of A from $n \times n$ to $2n \times 2n$, what can you predict for the ratio

$$\frac{f(2n)}{f(n)}$$

of the two flop counts? Calculate the actual ratios of the flop counts when the matrix size is doubled (three ratios to calculate). How does the calculated value compare with your predicted value for this ratio?

Optional Extra-Credit Question: Row Reduction with Row Exchanges

Even when the equation $Ax = b$ has a unique solution (the *regular* case), the method used in Question 1 may fail. It may be necessary to permute the rows of A and \mathbf{b} during the row reduction process to obtain the pivots (which must be nonzero). This can be carried out using a matrix factorization $PA = LU$, where P is a permutation matrix and L and U are as before. The MATLAB teaching code `splu.m` calculates the $PA = LU$ factorization (see Strang, page 94). The teaching code `splv.m` then applies this factorization to solve $Ax = \mathbf{b}$. You will need both of these codes to do this question.

(a) Generating an Exceptional Random Matrix: Use `newA` repeatedly until you obtain a matrix A with $A(1,1) = 0$. Then type `slu(A)`. Put in a comment that explains the source of the error message. Now type

```
[P, L, U] = splu(A)
```

This will generate a permutation matrix P and lower/upper triangular matrices L, U . Calculate PA and describe in words what the permutation matrix P does to the rows of A . Use MATLAB to check that $PA = LU$.

(b) Solving $Ax = b$ with Row Exchanges: Generate a random vector \mathbf{b} by `newb`. Using the same exceptional matrix A that you generated in part (a), type `splv(A, b)`. Put in a comment that explains the source of the error message. Now type

```
x = splv(A, b)
```

Check that $Ax = \mathbf{b}$.