EEEN203 Analogue Circuits and Systems
Lab 2
Introduction to Matlab

Not assessed

Matlab is a fast and enjoyable way to solve problems numerically. The computational problems arising in most undergraduate courses can be solved much more quickly with Matlab, than with the standard programming languages (Fortran, C, Java, etc.). It is particularly easy to generate some results, draw graphs to look at the interesting features, and then explore the problem further. By minimising human time, Matlab is particularly useful in the initial investigation of real problems; even though they may eventually have to be solved using more computationally efficient ways.

Matlab is widely used in engineering industry. It is in fact the standard mathematical programming language used by electronic engineers of all types. This is in part due to the availability of a wide range of specialised toolboxes to help solve problems in control, signal processing and communication engineering (amongst many others). You will find Matlab useful in almost all future EEEN subjects.

Having said that, if you feel comfortable using `numpy` and `scipy` libraries in Python, feel free to use those instead – but you will be a bit more on your own if you do so.

One drawback of Matlab is that it is not free. However you can download the freeware packages Octave or Scilab that are similar to Matlab for home use.

This introduction is condensed, modified, and updated from the introduction by Graeme Chandler (`http://www.maths.uq.edu.au/~gac/mlb/mlb.pdf`).

1 Getting Started

This Lab familiarizes you with the most important parts of Matlab. This very first section provides an overview of some basic information needed to get started. You should work through the notes trying out the various commands as discussed. When you reach section **??** you should have the foundations necessary to solve the circuit analysis problems posed.

1.1 What and How

The lab requires the use of Matlab or Octave (a freeware Matlab look-alike). The name Matlab was originally derived from MATrix LABoratory, because the fundamental data type is the matrix.

The best way to use this introduction is to sit down at a computer and type in the commands as they are described. Look at Matlab's response, and check that the answers are what you expect. It is also a good idea to do the exercises. That makes sure that the commands become part of an active Matlab vocabulary. More information about any Matlab command can be found by using the on-line help features.

These notes assume basic familiarity with the basic computer interface. For instance, you need to know about cutting and pasting within and between windows, editing files, and moving between windows and resizing them. If you are unsure about this, seek help from another student or tutor. In the first part of this LabThe we learn simple arithmetic, matrix-vector operations (including solving systems of equations), and graphing functions and data. The later sections describe some more advanced features. There are also some suggestions about using Matlab to do larger projects, and including Matlab results and graphs in reports. Finally we move on to some real statistics tasks.

Take the time to work through the examples carefully so you are comfortable using Matlab. The descriptions of the laboratories will not provide detailed instructions of how to perform every step like the introductory tutorials do, so you'll need to have some idea about how to perform the various operations.

1.2 Further References

More information on Matlab can be found in the books

1. D.M. Etter; *Engineering Problem Solving with Matlab,* Prentice-Hall, 1993.
2. Duane C. Hanselman & Bruce Littlefield; *Mastering MATLAB 5: A Comprehensive Tutorial and Reference,* Prentice Hall, 1998
3. Kermit Sigmon, *A Matlab Primer* 4th Ed. 1995

or try the Mathworks web site `http://www.mathworks.com`

If you wish to use Matlab at home, the student edition of Matlab is available from bookshops. The student version is 'crippled' and only works for small problems. Nevertheless, it should suffice for almost all assignments in undergraduate courses. It is usually possible to develop a program on the student version and do the final full size problem on the university computers if necessary.

1.3 Starting Up Matlab

Here we describe the steps needed to start Matlab on the PCs in the laboratories. On other machines the start-up procedure will be different. The Matlab commands are the same on all machines, if they are available. Machines may or may not have one or more of the Toolboxes (statistics, signal processing, etc.), and that means they have a different set of more sophisticated commands.

1. First sit down at a PC. Login and/or if necessary, close any programs left running by a previous user.
2. Somewhere on the desktop there will be a MATLAB icon for starting Matlab. When you click on this the Matlab logo appears briefly, then the 'MATLAB Command Window' remains on the screen. It ends with the words:-

```
To get started, type one of these commands:
helpwin, helpdesk, or demo.
For information on all of the MathWorks products, type tour.

>>
```

Matlab is now ready!

1.4 Typing Commands

All commands to Matlab are typed in after the Matlab prompt, i.e., the symbol `>>`. They are then sent to Matlab to be implemented by pressing the [Enter] key.

Any typing error can be corrected before the [Enter] key is pressed.

- Use the keypad left and right arrows or the mouse to move to the error.

- Use the [Del] key to delete the mistakes, and then type in the correction.
- When the line is correct, just press [Enter] to send the command to Matlab. (It is not necessary to go to the end of the line.)

Usually errors are not noticed until Matlab beeps and displays an error message. However *it is not necessary to retype the whole command.* Previous commands can be corrected and reused to save typing.

- Just press the keypad up arrow and the previous command appears.
- Make the necessary corrections and press [Enter] to run the corrected command.

If you want to clear the command window at any time you can do so with the `clc` command.

Now go on to the first lesson (Section **??**). Type in all the commands as they are shown, and make sure the Matlab response is what you expect.

Exercise:

Type in the Matlab command `logo`. A nice 3D graph of the Matlab logo should appear. This is produced by advanced use of Matlab's graphics capabilities, discussed later.

2 Simple Calculations

2.1 Basic Arithmetic

The simplest way to start with Matlab is to use it for simple calculations. Matlab has a wide range of functions and is able to use complex numbers as well as reals. The following simple commands have obvious meanings. Type them in and see what happens. (In the examples below, all the text after the % is just a comment or an explanation. You do not have to type it in. It would just be ignored by Matlab.)

```
% The words after '%' are comments
% and explanations. Do not type them in.

(-1+2+3)*5 - 2/3    % The four arithmetic operations
2^3                 % Means 2 to the power 3
exp( sin( pi/2 ) )  % The usual functions are provided
                    % log, log10, cos, tan, asin, ...
```

Like a calculator, Matlab does all calculations correct only to about 16 significant decimal digits. This is enough accuracy for most purposes. For convenience, usually only the first 5 significant digits are displayed on the screen. Some more examples:

```
pi
22/7                % pi is not the same as 22/7!
11*(15/11) - 15     % This shows there is roundoff error
                    % when Matlab uses fractions.
cos( pi/3 )         % These are familiar trig functions.
sin( pi/6 )         % Note Matlab always uses radians.
```

Matlab also uses variables to store intermediate answers. A variable can have almost any name, but it must begin with a letter. Matlab distinguishes between upper and lower case letters.

```
x = 2+3
y = 4+5
result1 = x/y
```

Sometimes the values of intermediate calculations are not needed and we do not want them to be displayed. If a semicolon (;) is placed after a command, the results are not displayed.

```
p = 2+3;          % The semicolons suppress
q = 3+5;          % unwanted output.
ratio = p/q
```

Several commands can be placed on one line, provided they are separated by a comma (,) or a semicolon (;). (Use a semicolon, unless you want the answer displayed.)

```
p = 2+3 ; q2 = x + 4 , ratio2 = p/q2   % All on one line
```

Parentheses can be used to make expressions clearer. Thus the last calculation could have been written as

```
ratio = (2+3)/(x+4)
```

Exercise:

In this exercise, save retyping by using the up arrow to edit the previous commands! Remember that multiplication is done with a '*'; it is not enough to put numbers next to each other.

1. Use Matlab to evaluate $\log(s^2 - 2s\cos(\pi/5) + 1)$ where $s = 5$.
2. Now use Matlab to evaluate $\log(s^2 - 2s\cos(\pi/5) + 1)$ where $s = .95$.
3. Finally evaluate $\log(s^2 - 2s\cos(\pi/5) + 1)$ where $s = 1$.

2.2 Complex Numbers

Matlab handles complex numbers as easily as real numbers. When you start Matlab the variables i and j stand for $\sqrt{-1}$.

```
x = 2 + 3*i , y = 1 - 1*i
z1 = x - y , z2 = x * y , z3 = x / y
abs( x )
angle( y )

sin(x)         % Matlab quickly calculates the
               %  sine of a complex number.
```

Matlab's facility with complex numbers is handy, as using complex numbers often involves complicated arithmetic. Indeed, as the previous example shows, Matlab will effortlessly work out functions of complex numbers that are difficult to do from first principles.

But be careful about the name used for $\sqrt{-1}$. When Matlab starts, the variables i and j contain $\sqrt{-1}$, but often the user overrides this and uses i or j to stand for another number (in coding $\sum_{i=1}^{n}$ for example). In this case the Matlab command `i=(-1)^.5` may be used to reset i to $\sqrt{-1}$ before it is used to represent complex numbers. A better method is to always use a different syntax to avoid this problem; and that is to immediately prepend a number before the i or j.

```
i = 2 ; j = 3 ; i+j   % i is used in another calculation
z = 2 + 3* i          % This is now not a complex calculation
z = 2 + 3i            % This is still complex
```

Here are two more complex calculations. Matlab can be used to demonstrate the elegant "Euler's identity": $e^{j\pi} = -1$ (At least to within round off error!).

```
exp( pi*1i ) + 1
1i^1i
```

3 Help in Matlab

There are three main ways to get help in Matlab.

3.1 The Help Command

This is the easiest way to find out more about specific Matlab commands. If you have forgotten small details, for example. The command `help <name>` gives information about the Matlab command `<name>`.

```
help sin            % help about sin
help i              % help about i
help log            % This is information about log
                    %   and shows that log means log to base e
```

By itself, the command `help` gives a list of topics in Matlab. Initially, many of the topics will be mysterious. The most familiar topic from the list is probably 'elementary functions'. To explore this topic use the command `help elfun`. The `lookfor` command can also be used to search for relevant information.

```
help                % Gives a list of topics
help elfun          % More help in the area ‘elfun’, i.e.,
                    % elementary functions like sin, exp,..
help sign           % Information about a command
                    % from the list in ’elfun’
lookfor logarithm   % Gives the name of some
                    % functions related to logarithms
```

3.2 The Help Window

It is also possible to get help by clicking on the 'Help' menu above the command window. From the 'Help' menu, select the 'Help Window' item (by clicking) and the list of help topics is displayed. The advantage of this method is that it is possible to navigate around various topics by double clicking on them. For example, in the help window double click on elfun to find the same information given by the help elfun command. This is a good way to explore Matlab's commands.

3.3 The Help Desk

Matlab also comes with help documentation that can be read by Web browsers. This may be used by

- typing the command `helpdesk` in the command window; or
- using the "Help" menu and selecting the item 'Help Desk (HTML)'

After a slight pause the initial documentation window should come up. Interesting links to pursue are:

- "Getting Started in Matlab" for introductory information
- "Documentation Road Map" for the complete set of manuals online, and
- the 'search' box to look for more commands.

Exercise:

1. Is the inverse sine function one of Matlab's elementary functions? (The inverse sine is also known as sin, arcsin, or asin.) Use help to find out.

2. Does Matlab have a specialised mathematical function to calculate the greatest common divisor? (Use `help` or `lookfor`)

   Use Matlab to find the greatest common divisor of 30 and 24. (Check the answer the same as the answer calculated manually.) What is the greatest common divisor of 3072 and 288?

3. Does Matlab have a function to convert numbers to base 16, i.e., to hexadecimal form? (Hint: Use `lookfor` to find a way to convert a decimal number to hexadecimal.) What is 61453 in base 16?

4. Use the search box in the Help Desk and look for information about logarithms.

4 Matrices and Vectors

Although Matlab is a useful calculator, its main power is that it gives a simple way of working with matrices and vectors. Indeed we have already seen how vectors are used in graphs.

Remember to keep typing in the commands as they appear here, and observe and understand the Matlab response. If you mistype, it is easy to correct using the arrows and the [Del] key. Try to use the help facility to find out about unfamiliar commands. Otherwise ask another student or a tutor.

4.1 Solving Equations

Most people will have seen systems of equations from school. For example, we may need to find x_1, x_2 and x_3 so that

$$\begin{aligned} x_1 + 2x_2 - x_3 &= 1 \\ -2x_1 - 6x_2 + 4x_3 &= -2 \\ -x_1 - 3x_2 + 3x_3 &= 1 \end{aligned}$$

The problem is rewritten in matrix-vector notation. We introduce a matrix A and a vector b by

$$A = \begin{bmatrix} 1 & 2 & -1 \\ -2 & -6 & 4 \\ -1 & -3 & 3 \end{bmatrix}, b = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

Now we want to find the solution vector In Matlab, we can set up the equations and find the solution using simple commands.

```
A = [ 1 2 -1; -2 -6 4 ; -1 -3 3 ]  % Set up a system of equations.
b = [ 1; -2; 1 ]
x = A\b                            % Find x with A x = b.
```

Matlab should give the solution

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 2 \end{bmatrix}$$

A sound idea from manual computations is to substitute the computed solution back into the system, and make sure all equations are indeed satisfied. In Matlab we can do this by checking that the matrix vector product Ax gives the vector b, or better still, we check that $Ax - b$ is exactly the zero vector.

```
A*x             % Check that Ax and b are the same.
b
A*x - b         % So that we do not miss small differences,
                % it is better to check that A x - b = 0 .
```

The vector $Ax - b$ is known as the *residual.*

Exercise

Change the middle element of A from -6 to 5 (i.e., $s_{22} = 5$). What is the solution to this new system? What is the new residual? Why is the residual not exactly 0? (*Hint:* In Matlab the number 1.23×10^{-5} is displayed as `1.23e-005`. The column vector $[1.23 \times 10{-}5; 4.44 \times 10^{-6}]$ could be displayed as

```
1.0e-004 *

0.123
0.0444
```

or

```
1.2300e-005
4.4400e-6
```

The user can control which form is used by the using the `format` command.

4.2 Matrices and Vectors

Much scientific computation involves solving very large systems of equations with many millions of unknowns. This section gives practice with the commands that are needed to work larger matrices.

The following code sets up a 4×4 matrix A, and then finds some of its important properties. When `[ ]` is used to set up matrices, either blanks or commas (,) can be used to separate entries in a row. Semicolons (;) are used to begin a new row.

```
A = [1 2 3 4 ; 1 4 9 16 ; 1 8 27 64 ; 1 16 81 256 ]
A'
det(A)
eig(A)
inv(A)
```

In Matlab, determinants and inverses can be quickly calculated and used.

4.2.1 Indexing

Indices can be used to show parts of a larger matrix. This feature is extremely powerful and you will need to master it to write efficient Matlab code. Rather than explain how to use indexing in great detail, we will simply look at some examples.

Try the following code:

```
A(2,3),   A(1:2,2:4),  A(:,2),  A(3,:)
```

In general `A( i:j, k:l )` means the rectangular sub-block of A between rows i to j and columns k to l. The ranges can be replaced by just ':' if all rows or columns are to be included.

Exercise

How would you display the bottom left 2×3 corner of A? How would you find the determinant of the upper left 3×3 block of A?

4.3 Creating Matrices

Matlab also provides quick ways to create special matrices and vectors.

```
c = ones(4,3)
d = zeros(20,1)
I = eye(5)
D = diag( [2 1 0 -1 -2] )
L = diag( [1 2 3 4], -1 )
U = rand(5,5)     % U is a 5 x 5 matrix of random number
                  % uniformly distributed between 0 and 1
N = randn(5,5)    % N is a 5 x 5 matrix of normally
                  % distributed numbers of zero mean
                  % and unit variance
```

More information on each of these commands can be found with `help`.

Matrix-matrix and matrix-vector multiplication work as expected, provided the dimensions agree. Matrices and vectors can both be multiplied by scalars. In the next exercise, the matrix B is another 4×4 matrix and c is a column vector of length 4 (i.e., a 4×1 matrix):

```
B = [1 1 0 0
     0 2 1 0
     0 0 3 1
     0 0 0 4]      % Rows can be separated by
                   %making a new line as well as using ‘;‘.

c = [1; 0; 0; -1]
5*B                % Multiply scalars and matrices.
B*c                % Multiply matrices and vectors.
A*B                % Matrix by matrix multiplication.
B*A                % Note: AB is not the same as BA !
```

Some of the following commands do not work. There is no harm in trying them.

```
c*c                % Wrong dimensions for matrix multiplication.
c*A                % Wrong dimensions for matrix multiplication.
c’                 % The transpose of c, i.e., a row vector.
c’*A               % Now matrix multiplication is permitted.
c*c’               % This multiplies a 4 x 1 by a 1 x 4 matrix.
c’*c               % What is this quantity usually called?
```

Exercise:

1. Calculate `inv(A)*A` and `A*inv(A)`. Do they give the expected result? (Use the help command if you don’t know what the `inv` command does.)
2. Verify for the matrices A and B above that $(AB)^{-1} = B^{-1}A^{-1}$
3. Verify that the Matlab command `A^(-1)` can also be used to form the inverse of `A`.

4.4 Systems of Equations

Consider again the problem of solving the system of equations $Ax = b$. One obvious way is to calculate $A^{-1}b$. As in the previous 3×3 case, we should also check that the solution is correct. We do this by calculating the residual $Ax - b$ for our computed solution x. Because of roundoff errors this residual may not be exactly zero, but all its components should be small; say less than 1×10^{-13}.

```
b = [1 0 0 1]';
x = A^(-1)*b          % Solve the equation A x = b
A*x , b               % Check x is correct by making sure
                      % it satisfies the equations.
resid = A*x - b       % Check the residual is zero, at
                      % least to within roundoff.
```

In fact it is far quicker, and usually more accurate, to solve equations using the backslash operator (\ introduced in the first section) rather than calculating with the inverse. The next lines use the backslash command to solve the equations $Ax = b$ and check that it gives the same answers as those that were calculated using A^{-1}.

```
x1 = A \ b           % This is the fastest way to solve Ax=b
x - x1               % The answers should be very close
```

Exercise

1. Solve the system of equations

$$\begin{bmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{bmatrix} x = \begin{bmatrix} 1/4 \\ 1/5 \\ 1/6 \end{bmatrix}$$

   using \ and check that the computed solution does actually satisfy the equations (i.e., check that $Ax - b = 0$, apart from rounding errors)

2. Use `rand` to generate a 700 × 700 matrix, A, and a column vector b of length 700. Solve the system of equations $Ax = b$ using the commands `x = A b`and `x = inv( A ) * b`,timing each command with a watch. Which command is the faster and why? (Hint: before working with these large matrices use the `clear` command to remove all variables from Matlab's memory. This frees up enough memory to squeeze in this largish problem. If you have problems, try a smaller system of equations that fits onto the computer you are using.)

   This exercise should show that the backslash operator \ is both faster and more accurate than the inverse. Unless you need to multiply the inverse by a large number of different vectors, it is better to use the backslash and avoid calculating inverses in numerical work!

5 Circuit Analysis

We can now apply the fundamentals of Matlab described above to solve problems in circuit analysis.

Consider the mesh analysis example in the DC analysis slides (around slide 24).

Exercise

1. Use Matlab's backslash operator to solve the matrix equation and ensure that you obtain the same answer as provided in the notes.

2. What would the currents be if you were to change the voltage source to 12V?

3. Use Matlab indexing notation (see **??**) and the `det` command to solve Cramer's rule for the circuit. Does your answer agree with that obtained previously?

Exercise Consider the circuit shown in Figure **??** where I_1, I_2, I_3 and I_4 are the current flowing clockwise through their respective loops. Write a set of simulataneous equations to describe the current flow in the circuit and then use Matlab to solve for the four currents.

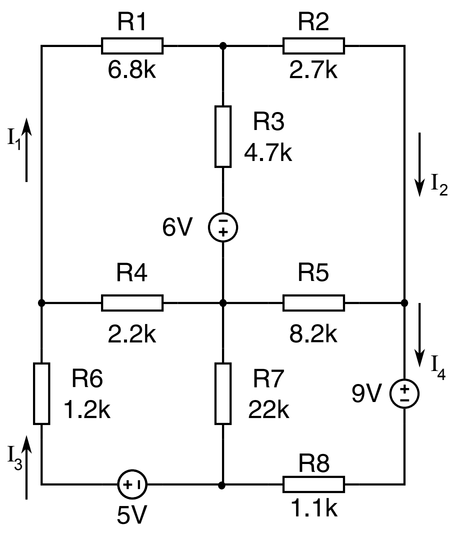

Figure 1: Example circuit, drawn from Boylestad "Introductory Circuit Analysis" 12th ed, figure 118.

You should find that

$$\begin{aligned}I_1 &= 0.0321 \text{ mA}\\ I_2 &= -0.8839 \text{ mA}\\ I_3 &= -0.6390 \text{ mA}\\ I_4 &= -0.9682 \text{ mA}\end{aligned}$$