# AN INTERACTIVE INTRODUCTION TO MATLAB



The University of Edinburgh, School of Engineering © 2010–2017

This course was developed in the School of Engineering at The University of Edinburgh to provide appropriate material for teaching MATLAB<sup>1</sup> in all engineering disciplines as well as to a wider audience. It is a self-study, self-paced course that emphasises responsible learning. The course material consists of this document used in conjunction with a series of video screencasts that are hosted on the School of Engineering YouTube channel.

## Who should use this document?

This document is targeted at those with no prior knowledge of MATLAB, and no previous programming experience. The aim, upon completion of the course, is to be competent using the most common features in MATLAB and be able to apply them to solve engineering problems.

# What is in this document?

This document forms part of a self-study course to help you get started with MATLAB. It should be used along with the supporting video screencasts and any additional material that maybe hosted on your institutions virtual learning environment. The main body of this document contains the fundamental topics for the course and there are also several more advanced topics given in appendices.

# How to use this document?

This document contains different elements designed to make your learning experience as smooth as possible. To benefit the most from these elements you are encouraged to use the online PDF version of this document. One of the first things you'll notice is that this document contains many links, those in red indicate a link to online material, and those in blue indicate a link to another section of this document.

A key part of this course are the screencasts, which are video screen captures (http://en.wikipedia.org/wiki/Screencast). In this document screencasts are indicated by a link in a blue box with a clapperboard icon, like the example shown.

You can use the commenting tools in Adobe Reader to add your own notes to this PDF document

Watching the

quickly!



screencasts and trying the examples for yourself will help you develop your skills in MATLAB more



Figure 1: A University of Edinburgh screencast

Clicking on a link to a screencast will take you to the appropriate page on the course website where you will see the opening image to a University of Edinburgh screencast presented in the video player (Figure 1). Watching and learning from the screencasts are an essential part of the course and will help you develop your skills in MATLAB more quickly.

You will also notice two other types of blue box environments in this document: one is for *Hints and Tips* (with a question mark icon), and the other contains exercises that you should complete (with an inkwell icon).



Throughout this document you will also see *Hints and Tips* boxes like this one. Please read these as they contain *useful* hints!



Additionally there are grey box environments in this document. Like the example shown (Listing 1), these contain code listings that demonstrate actual MATLAB code. Line numbers are given to the left of the listings to make is simpler to refer to specific

bits of code. Very often you will be required to copy and paste the listing into MATLAB and try running it for yourself.

Listing 1: Example of a code listing

```
>> 5+5
2 ans =
10
```

## Sources of help and further reading

There are a huge number of textbooks published on the subject of MATLAB! A user-friendly textbook that provides a good introduction to MATLAB is:

• Gilat, A. (2008). MATLAB: An Introduction With Applications. John Wiley & Sons, Inc., 3rd edition.

There are a couple of further textbooks listed in the Bibliography section at the end of this document. However, throughout this course and beyond, the most important source of help is the documentation built-in to MATLAB. It is easily searchable, and because MATLAB contains many built-in functions it is worth checking out before starting to write your own code.

- MATLAB help documentation (http://www.mathworks.com/access/helpdesk/help/techdoc/) Accessed through the help menu in MATLAB, or online.
- MATLAB CENTRAL

  (http://www.mathworks.co.uk/matlabcentral/)
  An open exchange for users, with code snippets, help forums and blogs. A great place to search for specific help!

Available from Amazon for c.£15

# DEVELOPMENT OF THE COURSE

The development of this course was funded through The Edinburgh Fund Small Project Grant which is part of The University of Edinburgh Campaign

(http://www.edinburghcampaign.com/alumni-giving/grants).

The material for this course was developed by Prof. Tina Düren, Dr. Antonis Giannopoulos, https://www.imperial.ac.uk/people/g.rein, Prof. John Thompson, and Dr. Craig Warren. The original screencasts were created by Dr. Craig Warren. Revised screencasts (to reflect an update to the MATLAB User Interface - MATLAB R2013a) were made by Joe Burchell.

#### LICENSE AND USAGE OF THIS COURSE

This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License. To view a copy of this license, visit <a href="http://creativecommons.org/licenses/by-nc-sa/3.0/">http://creativecommons.org/licenses/by-nc-sa/3.0/</a> or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

Under the terms of the license we would be grateful if you would cite the following paper:

• Warren, C. (2014). MATLAB for Engineers: Development of an Online, Interactive, Self-study Course. Engineering Education, 9(1), 86-93.

# ACKNOWLEDGMENTS

An Interactive Introduction to MATLAB makes use of the classic thesis template for LaTeX, which was created by Prof. Dr.-Ing. André Miede.

# CONTENTS

	0.1	What is MATLAB?	1
	0.2	How is MATLAB used in industry?	1
1	BAS	IC CONCEPTS	3
	1.1	MATLAB in the School of Engineering, University of Edinburgh	3
	1.2	The MATLAB environment	3
	1.3	Basic calculations	4
	1.4	Variables and arrays	6
	1.5	Solving systems of linear equations	12
2	PLO	TTING	19
	2.1	Simple 2D plotting	19
		2.1.1 Multiple plots in one Figure Window	22
	2.2	Curve-fitting	23
	2.3	3D plotting using plot3 and surf	25
3	SCR	IPTS AND FUNCTIONS	33
	3.1	Script files	33
	3.2	Functions	38
4	DEC	ISION MAKING	43
	4.1	Relational and logical operations	43
	4.2	The if-else statement	46
5	LOO	PS	55
	5.1	for loops	55
	5.2	while loops	58
A	ADV	ANCED TOPIC: THE SWITCH STATEMENT	65
В	ADV	ANCED TOPIC: VECTORISATION	67
С	ADI	OITIONAL EXERCISES	69
	C.1	Basic Concepts	69
	C.2	Plotting	72
	C.3	Scripts and Functions	80
	c.4	Decision Making	85
ВΙ	BLIO	GRAPHY	89

# LIST OF SCREENCASTS

The MATLAB desktop	3
Exercise 1 Solutions	6
Variables and simple arrays	9
The dot operator	10
Indexing arrays	12
Exercise 2 Solutions Q 1 - 4	17
Exercise 2 Solutions Q 5 - 8	17
Creating a simple plot	21
Plotting experimental data	21
Exercise 3 Solutions	23
Basic Curve-fitting	24
Exercise 4 Solutions	28
Creating a simple script	35
Exercise 5 Solutions	37
Creating a function	41
Exercise 6 Solutions	42
The if-else statement	49
Exercise 7 Solutions	53
The for loop	56
The while loop	59
Exercise 8 Solutions	62

# LIST OF EXERCISES

Exercis	1: Basic	calculation	ons .									
Exercis	2: Varial	oles and a	irrays									1
Exercis	3: Simple	2D plott	ing.									2
Exercis	4: 3D plo	tting										2
Exercis	5: Script	s										3
Exercis	6: Functi	ions										4
Exercis	7: Decisi	on makin	g									5
Exercis	8: Loops											6

# LIST OF TABLES

Arithmetic operations	5
Element-by-element arithmetic operations	10
Line styles in plots	20
Colours in plots	21
Function definitions, filenames, input and output vari-	
ables	39
Relational operators	43
Logical operators	43
Friction experiment results	71
Results of a tension test on an aluminium specimen	73
Coefficients for the cubic equation for the heat capacity	
of gases	81
	Element-by-element arithmetic operations  Line styles in plots  Colours in plots  Function definitions, filenames, input and output variables  Relational operators  Logical operators  Friction experiment results  Results of a tension test on an aluminium specimen  Coefficients for the cubic equation for the heat capacity

# 0.1 WHAT IS MATLAB?

MATLAB is produced by MathWorks, and is one of a number of commercially available software packages for numerical computing and programming. MATLAB provides an interactive environment for algorithm development, data visualisation, data analysis, and numerical computation. MATLAB, which derives its name from MATrix LABoratory, excels at matrix operations and graphics. Its main competitors are Maple, Mathematica, and Mathcad, each with their own strengths and weaknesses.

MATLAB is available in both commercial and academic versions with new releases binannually e.g. R2016a (released around March 2016), and R2016b (released around September 2016). MATLAB itself is the core product and is augmented by additional toolboxes, many of which have to be purchased separately. If you want to run MATLAB on your own computer MathWorks offers a student version (http://www.mathworks.com/academia/student version/) with some of the most commonly used toolboxes for around £55 +VAT. The accompanying online material, and the first version of screenshots for this document were originally based on the R2009a release of MATLAB. As the MATLAB user interface was substantially redeveloped by Mathworks few years later the current version of the course has been redesigned to conform to this change and is now based on the new User interface of MATLAB R2013a running under Microsoft Windows. MATLAB is available for other operating systems such are Mac OS X and Linux. The User Interface is very similar and the material in this course is directly applicable to versions of MATLAB running under these operating systems. Finally, as the course aims to present very basic concepts of MATLAB the material is widely applicable to most versions of MATLAB.

MATLAB student version is available for around £55 + VAT

#### 0.2 HOW IS MATLAB USED IN INDUSTRY?

The ability to use tools such as MATLAB is increasingly required by employers of graduate engineers in industry. Many job adverts specifically mention knowledge of MATLAB as an essential skill.

Knowing how to use MATLAB is a vital skill for many engineering jobs!

MATLAB is a widely-used tool in many different fields of engineering and science. The following is a brief list of examples from Chemical, Civil, Electrical, and Mechanical Engineering:

- Motorsport Teams Improve Vehicle Performance with MathWorks Tools (http://uk.mathworks.com/company/user\_stories/penske-technology-group-enables-motorsport-teams-to-improve-vehicle-performance.html)
- Bell Helicopter Develops the First Civilian Tiltrotor (http://www.mathworks.com/company/newsletters/news\_notes/oct06/bellhelicopter.html)

- Modeling Flexible Bodies in SimMechanics and Simulink (http://uk.mathworks.com/company/newsletters/articles/modeling-flexible-bodies-in-simmechanics-and-simulink.html?s\_tid=srchtitle)
- Samsung UK Develops 4G Wireless Systems with Simulink (http://uk.mathworks.com/company/user\_stories/samsung-uk-develops-4g-wireless-systems-with-simulink.html?s\_tid=srchtitle)
- Elektrobit Testing Ltd. Develops High-Resolution Radio Channel Measuring System
  (http://uk.mathworks.com/company/user\_stories/elektrobit-testing-ltd-develops-high-resolution-radio-channel-measuring-system.html?s\_tid=srchtitle)
- Halliburton Makes Oil Exploration Safer Using MATLAB and Neural Networks
   (http://uk.mathworks.com/company/user\_stories/halliburton-makes-oil-exploration-safer-using-matlab-and-neural-networks.html?s\_tid=srchtitle)

BASIC CONCEPTS

1.1 MATLAB IN THE SCHOOL OF ENGINEERING, UNIVERSITY OF EDINBURGH

MATLAB is currently available under Microsoft Windows and Linux operating systems in the School of Engineering Computing Labs, and also under Microsoft Windows in all of the University's Open Access Computing Labs<sup>1</sup>. MATLAB is available to both students and staff of the University under a site license. More information on how to get access to MATLAB on your computer can be found at the University's Information Services website<sup>2</sup>

#### 1.2 THE MATLAB ENVIRONMENT

When you launch MATLAB you are presented with the MATLAB desktop (Figure 2) which, by default, is divided into 4 windows:

- 1. Command Window: This is the main window, and contains the command prompt (\*\*). This is where you will type all commands.
- Command History: Displays a list of previously typed commands. The command history persists across multiple sessions and commands can be dragged into the Command Window and edited, or double-clicked to run them again.
- 3. Workspace: Lists all the variables you have generated in the current session. It shows the type and size of variables, and can be used to quickly plot, or inspect the values of variables.
- 4. Current Directory: Shows the files and folders in the current directory. The path to the current directory is listed near the top of the MAT-LAB desktop. By default, a MATLAB folder is created in your home directory, and this is where you should save your work.

You will use and become more familiar with the different areas of the MAT-LAB desktop as you progress through this course.



Remember you can pause the screencasts at any time and try the examples for yourself.

 $<sup>1\ (</sup>http://www.ed.ac.uk/information-services/computing/desktop-personal/open-access/locations)$ 

 $<sup>2\ (</sup>http://www.ed.ac.uk/information-services/computing/desktop-personal/software/main-software-deals/matlab)$ 

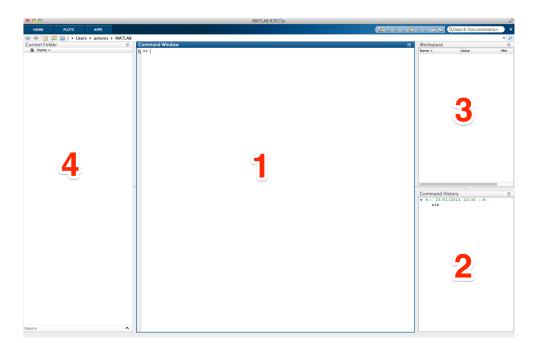


Figure 2: The MATLAB desktop

# 1.3 BASIC CALCULATIONS

MATLAB can perform basic calculations such as those you are used to doing on your calculator. Listings 1.1–1.5 gives some simple examples (and results) of arithmetic operations, exponentials and logarithms, trigonometric functions, and complex numbers that can be entered in the Command Window.

Listing 1.3: Trigonometry

The arguments to trigonometric functions should be given in radians.

#### Comments:

- MATLAB has pre-defined constants e.g.  $\pi$  may be typed as pi.
- You must explicitly type all arithmetic operations e.g. sin(2\*pi) not sin(2pi).

•  $\sin(x)$  and  $\exp(x)$  correspond to  $\sin(x)$  and  $e^x$  respectively.

Listing 1.4: Complex numbers

```
>> 5+5j
ans =
5.0000 + 5.0000i
```

## Comments:

• Complex numbers can be entered using the basic imaginary unit i or j.

Listing 1.5: More trigonometry

# Comments:

• atan(x) and log10(x) correspond to  $tan^{-1}(x)$  and  $log_{10}(x)$  respectively.

Table 1: Arithmetic operations

COMMAND	DESCRIPTION
+	Addition
_	Subtraction
*	Multiplication
/	Division
^	Exponentiation

# Built-in functions

There are many other built-in MATLAB functions for performing basic calculations. These can be searched from the Help Browser which is opened by clicking on its icon (like the icon used to indicate this Hints and Tips section) in the MATLAB desktop toolbar.



# Exercise 1: Basic calculations

- 1. Launch MATLAB and explore the different areas of the MATLAB desktop.
- 2. Try the basic calculations given in Listings 1.1–1.5, and check you get the correct answers.
- 3. Arithmetic operations Compute the following:
  - $\frac{2^5}{2^5-1}$  and compare with  $\left(1-\frac{1}{2^5}\right)^{-1}$

[Answers: 1.0323, 1.0323, 0.1180]

- 4. Exponentials and logarithms Compute the following:

  - $ln(e^3)$
  - $log_{10}(e^3)$
  - $log_{10}(10^5)$

[Answers: 20.0855, 3, 1.3029, 5]

- 5. Trigonometric operations Compute the following:
  - $\sin(\frac{\pi}{6})$
  - $\cos(\pi)$
  - $\tan(\frac{\pi}{2})$
  - $\sin^2(\frac{\pi}{6}) + \cos^2(\frac{\pi}{6})$

[Answers: 0.5, -1, 1.6331E16, 1]



# Exercise 1 Solutions

(https://youtu.be/rZuAns0iEt4)

You may have noticed that the result of each of the basic calculations you performed was always assigned to a variable called ans. Variables are a very important concept in MATLAB.

#### 1.4 VARIABLES AND ARRAYS

A variable is a symbolic name associated with a value. The current value of the variable is the data actually stored in the variable. Variables are very important in MATLAB because they allow us to easily reference complex and changing data. Variables can reference different data types i. e. scalars, vectors, arrays, matrices, strings etc.... Variable names must consist of a letter which

can be followed by any number of letters, digits, or underscores. MATLAB is case sensitive i.e. it distinguishes between uppercase and lowercase letters e.g. A and a are not the same variable.

Variables you have created in the current MATLAB session can be viewed in a couple of different ways. The Workspace (shown in Figure 2) lists all the current variables and allows you to easily inspect their type and size, as well as quickly plot them. Alternatively, the whos command can be typed in the Command Window and provides information about the type and size of current variables. Listing 1.6 shows the output of the whos command after storing and manipulating a few variables.

Listing 1.6: Using the whos command

```
a = 2
   >>
                2
3
             3
4
   >>
      b
         =
5
                3
7
                6
9
   >> edinburgh
10
   edinburgh
11
12
      whos
13
      Name
                           Size
                                                  Bytes
                                                            Class
14
           Attributes
15
                           1x1
                                                        8
                                                            double
16
      b
                           1 x 1
                                                        8
                                                            double
17
      С
                           1 \times 1
                                                        8
                                                            double
18
      edinburgh
                           1 \times 1
                                                        8
                                                            double
19
```

Arrays are lists of numbers or expressions arranged in horizontal rows and vertical columns. A single row, or single column array is called a vector. An array with m rows and n columns is called a matrix of size  $m \times n$ . Listings 1.7–1.10 demonstrate how to create row and column vectors, and matrices in MAT-LAB.

Listing 1.7: Creating a row vector

- Square brackets are used to denote a vector or matrix.
- Spaces are used to denote columns.

Listing 1.8: Creating a column vector

```
1 >> y = [4; 5; 6]
2 y =
3 4
4 5
5 6
```

• The semicolon operator is used to separate columns.

Listing 1.9: The transpose operator

```
>> x,
2
  ans =
                 1
3
                 2
4
                 3
5
6
7
  >> y,
8
  ans
                 4
                        5
                               6
9
```

• The single quotation mark ' transposes arrays, i.e. the rows and columns are interchanged so that the first column becomes the first row etc...

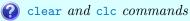
A more efficient method for entering vectors, especially those that contain many values, is to use ranges. Instead of entering each individual value separately, a range of values can be defined as shown in Listing 1.10.

Listing 1.10: Creating vectors using ranges

```
>> z = 8:1:10
1
2
  z =
             8
                   9
                         10
3
        = linspace(0,10,5)
5
                   2.5000
                               5.0000
             0
                                           7.5000
                                                       10.0000
7
```

# Comments:

- A range can be created using the colon operator, e.g. 8:1:10 means create a range that starts at 8 and goes up in steps of size 1 until 10.
- A range can also be created using the linspace function,
   e.g. linspace(0,10,5) means create a range between 0 and 10 with 5 linearly spaced elements.



The clear command can be used if you want to clear the current workspace of all variables. Additionally, the clc command can be used to clear the Command Window, i.e. remove all text.

```
Variables and simple arrays
(https://youtu.be/Nsme7btg75U)
```

MATLAB excels at matrix operations, and consequently the arithmetic operators such as multiplication (\*), division (/), and exponentiation (^) perform matrix multiplication, division, and exponentiation, when used on a vector, by default. To perform an element-by-element multiplication, division, or exponentiation you must precede the operator with a dot. Table 2 and Listing 1.11 demonstrate the dot operator.

Listing 1.11: The dot operator

```
>> clear
   >>
       a = [2 \ 3; \ 5 \ 1]
2
3
                2
                       3
4
                5
                       1
             [4 7; 9 6]
   >>
6
7
                       7
                4
                9
                       6
   >> a*b
10
11
   ans
                           32
12
              35
              29
                           41
13
   >> a.*b
   ans =
15
                       21
                8
```

```
45
                       6
17
           [1 2 3 4]
      С
18
19
                  2
                        3
20
21
   >> a*c
   ??? Error using ==> mtimes
22
   Inner matrix dimensions must agree.
24
   ??? Error using ==> mtimes
   Matrix dimensions must agree.
```

## Comments:

- The dot operator signifies an element-by-element operation. The dot can be used for multiplication .\*, division ./, or exponentiation .^ of elements of vectors that are the same size. Omitting the dot before an arithmetic operator means MATLAB performs the matrix version of the operation.
- On Line 21 we tried to perform a matrix multiplication of a  $2\times 2$  matrix with a  $1\times4$  matrix. This results in an error because you can only multiply two matrices if the number of columns in the first equals the number of rows in the second.
- On Line 24 we get a similar error if we try to perform an element-byelement multiplication, as this does not make any sense for matrices of different sizes.

Table 2: Element-by-element arithmetic operations

COMMAND	DESCRIPTION
.*	Element-by-element multiplication
./	Element-by-element division
.^	Element-by-element exponentiation



The dot operator

(https://youtu.be/gvxxr2R0S-o)



Read MATLAB error messages!

??? Error using ==> mtimes

Inner matrix dimensions must agree.

This error message example usually indicates you tried to perform a matrix operation when you intended an element-by-element operation. You should check your code for a missing dot operator.

You can access individual elements, entire rows and columns, and subsets of matrices using the notation matrix\_name(row,column). Listing 1.12 demonstrates how to access elements in a matrix.

Listing 1.12: Accessing elements of matrices

```
[1 2 3 4; 5 6 7 8; 9 10 11 12]
2
                             3
                1
                      2
                                    4
3
                5
                      6
                             7
                                    8
4
                9
                     10
                                   12
                            11
5
6
   >> size(w)
7
   ans
8
                3
                      4
9
10
   >> w(1,1)
11
12
   ans =
                1
13
14
   >> w(3,1)
15
16
                9
17
18
   >> w(3,:)
19
20
                9
                      10
                               11
                                       12
21
22
      w(2,4) = 13
23
24
                1
                      2
                             3
                                    4
25
                5
                      6
                             7
                                   13
26
                9
                     10
                            11
                                   12
27
          = w(1:2,2:3)
29
30
                2
                      3
31
                6
                      7
32
33
   >> z = w([2,3],[2,4])
34
35
                6
                     13
36
              10
                     12
37
```

Comments:

- On Line 7 the size command returns the number of rows and columns in the matrix.
- On Lines 11, 15 and 19, when accessing an individual element in a matrix, the first number after the round bracket refers to the row number (row index), and second number refers to the column number (column index).

Square brackets [] are used when creating vectors, arrays and matrices, and round brackets () when accessing elements in them.

- On Line 19 the colon operator is used to denote all of the columns, i.e. all the columns in the third row are selected. The colon operator can also be used as a row index to denote all rows.
- Line 23 demonstrates accessing a single element in the matrix w to change its value.
- On Line 29 a new matrix v is created as a sub-matrix of w.
- Finally, on Line 34 a new matrix z is created as a sub-matrix of w. Square brackets are used within the round brackets to enclose the list of row and column numbers.



Indexing arrays

(https://youtu.be/eEhwnCq0\_Qg)



Self Test Exercise: Indexing arrays

1. <sup>2</sup>The following matrix is defined:

$$M = \left[ \begin{array}{ccccccc} 6 & 9 & 12 & 15 & 18 & 21 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 2 & 1 & 0 & -1 & -2 & -3 \\ -6 & -4 & -2 & 0 & 2 & 4 \end{array} \right]$$

Evaluate the following expressions without using MATLAB. Check your answers with MATLAB.

- a) A = M([1,3], [2,4])
- b) B = M(:, [1,4:6])
- c) C = M([2,3], :)

# SOLVING SYSTEMS OF LINEAR EQUATIONS

Solving systems of linear equations is one of the most common computations in science and engineering, and is easily handled by MATLAB. Consider the following set of linear equations.

$$5x = 3y - 2z + 10$$
$$8y + 4z = 3x + 20$$

$$2x + 4y - 9z = 9$$

This set of equations can be re-arranged so that all the unknown quantities are on the left-hand side and the known quantities are on the right-hand side.

$$5x - 3y + 2z = 10$$
$$-3x + 8y + 4z = 20$$

$$2x + 4y - 9z = 9$$

<sup>2</sup> Question adapted from Gilat, A. (2008). MATLAB: An Introduction With Applications. John Wiley & Sons, Inc., 3rd edition. Copyright ©2008 John Wiley & Sons, Inc. and reprinted with permission of John Wiley & Sons, Inc.

This is now of the form AX = B, where A is a matrix of the coefficients of the unknowns,

$$A = \left[ \begin{array}{rrr} 5 & -3 & 2 \\ -3 & 8 & 4 \\ 2 & 4 & -9 \end{array} \right]$$

x is the vector of unknowns,

$$X = \left[ \begin{array}{c} x \\ y \\ z \end{array} \right]$$

and B is a vector containing the constants.

$$B = \left[ \begin{array}{c} 10 \\ 20 \\ 9 \end{array} \right]$$

Listing 1.13 shows the code used to solve the system of linear equations in MATLAB. The rules of matrix algebra apply i.e. the result of multiplying a  $N \times N$  matrix by a  $N \times 1$  vector, is a  $N \times 1$  vector.

Listing 1.13: Solving a system of linear equations

Using a semi-colon at the end of a command prevents the results being displayed in the Command Window.

## Comments:

- On Line 1 the matrix, A, of coefficients of the unknowns is entered.
- On Line 2 the vector, B, containing the constants is entered.
- On Line 3 the vector, X, containing the unknowns, is calculated by using the matrix left divide operator to divide A by B.

Listing 1.14 demonstrates how to check the solution obtained in Listing 1.13.

Listing 1.14: Checking the solution of a system of linear equations

Not all systems of linear equations have a unique solution. If there are fewer equations than variables, the problem is under-specified. If there are more equations than variables, it is over-specified.

The left division or backslash operator (\)

In MATLAB the left division or backslash operator (\) is used to solve equations of the form AX=B i.e.  $X=A\setminus B$ . Gaussian elimination is used to perform this operation.

# Exercise 2: Variables and arrays

1. Create the variables to represent the following matrices:

$$A = \begin{bmatrix} 12 & 17 & 3 & 4 \end{bmatrix} \qquad B = \begin{bmatrix} 5 & 8 & 3 \\ 1 & 2 & 3 \\ 2 & 4 & 6 \end{bmatrix} \qquad C = \begin{bmatrix} 22 \\ 17 \\ 4 \end{bmatrix}$$

- a) Assign to the variable  $\mathtt{x1}$  the value of the second column of matrix  $\mathtt{A}$ .
- b) Assign to the variable x2 the third column of matrix B.
- c) Assign to the variable x3 the third row of matrix B.
- d) Assign to the variable x4 the first three values of matrix A as the first row, and all the values in matrix B as the second, third and fourth rows.
- 2. If matrix A is defined using the MATLAB code A = [1 3 2; 2 1 1; 3 2 3], which command will produce the following matrix?

$$B = \left[ \begin{array}{cc} 3 & 2 \\ 2 & 1 \end{array} \right]$$

3. Create variables to represent the following matrices:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 2 & 2 \\ -1 & 2 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 1 & 2 \end{bmatrix}$$

- a) Try performing the following operations: A+B, A\*B, A+C, B\*A, B-A, A\*C, C-B, C\*A. What are the results? What error messages are generated? Why?
- b) What is the difference between A\*B and A.\*B?
- 4. Solve the following systems of linear equations. Remember to verify your solutions.

a)

$$-2x + y = 3$$
$$x + y = 10$$

Exercise 2: Variables and arrays (continued)

4. continued

b)

$$5x + 3y - z = 10$$
$$3x + 2y + z = 4$$
$$4x - y + 3z = 12$$

c)

$$x_1 - 2x_2 - x_3 + 3x_4 = 10$$
$$2x_1 + 3x_2 + x_4 = 8$$
$$x_1 - 4x_3 - 2x_4 = 3$$
$$-x_2 + 3x_3 + x_4 = -7$$

5. Create a vector t that ranges from 1 to 10 in steps of 1, and a vector theta that ranges from 0 to  $\pi$  and contains 32 elements. Now compute the following:

$$x = 2\sin(\theta)$$
$$y = \frac{t-1}{t+1}$$
$$z = \frac{\sin(\theta^2)}{\theta^2}$$

6. A discharge factor is a ratio which compares the mass flow rate at the end of a channel or nozzle to an ideal channel or nozzle. The discharge factor for flow through an open channel of parabolic cross-section is:

$$K = \frac{1.2}{x} \left[ \sqrt{16x^2 + 1} + \frac{1}{4x} \ln \left( \sqrt{16x^2 + 1} + 4x \right) \right]^{-\frac{2}{3}},$$

where x is the ratio of the maximum water depth to breadth of the channel at the top of the water. Determine the discharge factors for x in the range 0.45 to 0.90 in steps of 0.05.

7. Points on a circle

All points with coordinates  $x = r\cos(\theta)$  and  $y = r\sin(\theta)$ , where r is a constant, lie on a circle with radius r, i.e. they satisfy the equation  $x^2 + y^2 = r^2$ . Create a column vector for  $\theta$  with the values 0,  $\pi/4$ ,  $\pi/2$ ,  $3\pi/4$ ,  $\pi$ , and  $5\pi/4$ . Take r=2 and compute the column vectors x and y. Now check that x and y indeed satisfy the equation of a circle, by computing the radius  $r = \sqrt{(x^2 + y^2)}$ .



Exercise 2: Variables and arrays (continued)

# 8. Geometric series

The sum of a geometric series  $1 + r + r^2 + r^3 + \ldots + r^n$  approaches the limit  $\frac{1}{1-r}$  for r < 1 as  $n \to \infty$ . Take r = 0.5 and compute sums of series 0 to 10, 0 to 50, and 0 to 100. Calculate the aforementioned limit and compare with your summations. Do the summation using the built-in sum function.



Exercise 2 Solutions Q 1 - 4

(https://youtu.be/pTdkD1UpGjU)



Exercise 2 Solutions Q 5 - 8

(https://youtu.be/nHroWoJApT8)



Additional Exercises

You should now attempt questions from Chapter C.1.

MATLAB is very powerful for producing both 2D and 3D plots. Plots can be created and manipulated interactively or by commands. MATLAB offers a number of different formats for exporting plots, including EPS (Encapsulated PostScript), PDF (Portable Document Format) and JPEG (Joint Photographic Experts Group), so you can easily include MATLAB plots in your reports.

#### 2.1 SIMPLE 2D PLOTTING

The simplest and most commonly used plotting command is plot(x,y), where x and y are simply vectors containing the x and y coordinates of the data to be plotted. Listing 2.1 demonstrates the commands used to create a plot of the function,  $f(x) = e^{-\frac{x}{10}} \sin(x)$ , which is shown in Figure 3.

Listing 2.1: A simple plot

```
1 >> x = 0:0.1:20;
2 >> y = exp(-x/10).*sin(x);
3 >> plot(x,y), grid on, xlabel('x'), ...
4 ylabel('f(x) = e^{-x/10} sin(x)'), title('A simple plot')
```

# Comments:

- The vectors containing the x and y data must be the same length.
- The plot command can be used to plot multiple sets of data on the same axes, i.e. plot(x1,y1,x2,y2).
- The dot-dot-dot ... (ellipsis) notation is used to indicate that Lines 3 and 4 are one long line. The ellipsis notation just allows the line to be broken to make it more readable. Each comma-separated command could also have been typed on a separate line.

When MATLAB executes a plotting command, a new Figure Window opens with the plot in it. The following list gives the most common commands for changing plot properties.

- grid on displays the grid!
- xlabel('My x-axis label'), ylabel('My y-axis label'), and title('My title') can be used to label the corresponding parts of the plot. You must enclose your labels with single quotes which denotes a string of text.

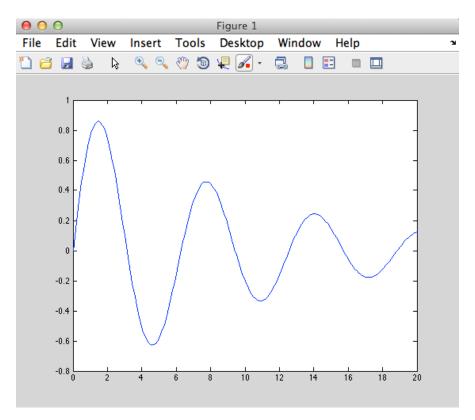


Figure 3: Plot of  $f(x) = e^{-\frac{x}{10}} \sin(x)$ 

- legend('Data1', 'Data2') is used to place a legend and label the datasets when you have multiple data-sets on one plot.
- You can specify line style and colour within the plot command e.g. plot (x1,y1,'b-',x2,y2,'r--'). This command would make the first data-set a solid blue line, and the second data-set a dashed red line. Tables 3-4 gives some of the most common line styles and colours.

STRING SPECIFIER LINE STYLE Solid line (default) Dashed line Dotted line Dash-dot line

Table 3: Line styles in plots

Plot properties can also be manipulated interactively (without having to issue commands) by clicking on the Show Plot Tools icon in the Figure Window toolbar, shown in Figure 4. Properties such as the axis limits, gridlines, line style, colour and thickness, text font type and size, and legend etc... can all be adjusted be clicking on the appropriate parts of the plot.

rabie ii colear	s in pious
STRING SPECIFIER	LINE COLOUR
r	Red
g	Green
b	Blue (default)
W	White
k	Black

Table 4: Colours in plots

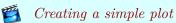


Figure 4: Show Plot Tools toolbar icon in Figure Window

# ? Producing good plots

Whether you manipulate your plots via commands or interactively, here is some useful advice for producing good plots in MATLAB.

- Give your plot an informative title
   e.g. title('Stress vs. strain of steel')
- Label your axes and remember to include units where appropriate e.g. xlabel('Strain'), ylabel('Stress (MPa)')
- Use line colours and styles carefully so that multiple data-sets can be easily distinguished e.g. plot(x1,y1,'b-',x2,y2,'r--'), grid on
- Remember to insert a legend when you are plotting multiple data-sets on one plot e.g. legend('Carbon steel', 'Stainless steel')



(http://www.eng.ed.ac.uk/teaching/courses/matlab/unit02/simple-plot.shtml)

MATLAB has many built-in plot types, and a great way of getting a quick overview of all the different plot types is to select a variable in your Workspace Browser, click on the disclosure triangle next to the *plot* toolbar icon and select *More plots...*, as shown in Figure 5a. This will launch the *Plot Catalog* shown in Figure 5b.



(http://www.eng.ed.ac.uk/teaching/courses/matlab/unit02/plot-exp-data.shtml)

(2) Importing data from external sources

You can import data from other programs into MATLAB using the Copy  $\rightarrow Paste$  method, or using the Import Data Wizard, found at File  $\rightarrow Import$ Data..., for Microsoft Excel data, Comma-separated value files and more. There are also functions, xlsread and xlswrite.

#### 2.1.1 Multiple plots in one Figure Window

The subplot command can be used to display a number of different plots in a single Figure Window, as shown in Figure 6. The subplot command takes three arguments that determine the number and location of plots in the Figure Window. For example, subplot(2,2,1) specifies that the Figure Window will be divided into 2 rows and 2 columns of plots, and selects the first subplot to plot into. Listing 2.2 shows an example of usage of the subplot command.

Listing 2.2: Using the subplot command

```
>> x = linspace(0,2*pi,50);
>> subplot(2,2,1), plot(x,sin(x)), xlabel('x'), ylabel('
   sin(x);
>> subplot(2,2,2), plot(x,cos(x)), xlabel('x'), ylabel('
   cos(x)');
>> subplot(2,2,3), plot(x,sin(2*x)), xlabel('x'), ylabel('
   sin(2x)');
>> subplot(2,2,4), plot(x,cos(2*x)), xlabel('x'), ylabel('
   cos(2x)');
```

Exercise 3: Simple 2D plotting

Please save all the plots you produce using the File  $\rightarrow Save$  option in the Figure Window. This should save a file with the MATLAB default Figure format which uses a .fig file extension.

- 1. Plot the following functions (you will need to decide on appropriate ranges for x):
  - $y = \frac{1}{x}$ , with a blue dashed line.
  - $y = \sin(x)\cos(x)$ , with a red dotted line.
  - $y = 2x^2 3x + 1$ , with red cross markers.

Turn the grid on in all your plots, and remember to label axes and use a title.

2. Given the following function:

$$s = a\cos(\phi) + \sqrt{b^2 - (a\sin(\phi) - c)^2}$$

Plot s as a function of angle  $\phi$  when a=1, b=1.5, c=0.3, and  $0 \le \phi \le 360^{\circ}$ .

- 3. Plot the following parametric functions (you will need to use the axis equal command after your plot command to force MATLAB to make the x-axis and y-axis the same length):
  - a) A circle of radius 5 (revisit Ex2 Q7)
  - b) Leminscate  $(-\pi/4 \le \phi \le \pi/4)$

$$x = \cos(\phi)\sqrt{2\cos(2\phi)}$$
$$y = \sin(\phi)\sqrt{2\cos(2\phi)}$$

c) Logarithmic Spiral  $(0 \le \phi \le 6\pi; k = 0.1)$ 

$$x = e^{k\phi}\cos(\phi)$$
$$y = e^{k\phi}\sin(\phi)$$



Exercise 3 Solutions

(http://www.eng.ed.ac.uk/teaching/courses/matlab/unit02/Ex3-Solutions.shtml)

# CURVE-FITTING

MATLAB provides a number of powerful options for fitting curves and adding trend-lines to data. The Basic Fitting Graphical User Interface (GUI) can be selected from Figure Windows by selecting Basic fitting from the Tools menu, and offers common curve-fitting options for 2D plots. More advanced functionality, including 3D fits, can be accessed from the Curve Fitting Toolbox using

tools such as cftool (for curve fitting) and sftool (for surface fitting)<sup>1</sup>.



Basic Curve-fitting

(http://www.eng.ed.ac.uk/teaching/courses/matlab/unit02/basic-curvefitting.shtml)

An alternative to the Basic Fitting GUI are the functions polyfit and polyval which can be used to do basic curve-fitting programmatically. Listing 2.3 demonstrates how polyfit can be used to fit a polynomial to a dataset.

Listing 2.3: Syntax of polyfit command

```
coeff = polyfit(xdata,ydata,n);
```

# Comments:

• coeff is a vector containing the coefficients for the polynomial of best fit, xdata and ydata are vectors containing the independent and dependent variables, and n denotes the degree of the polynomial to be fitted.

After using polyfit you can use the polyval function to evaluate the polynomial of best fit, given by the set of coefficients coeff, at specific values of your data. This creates a vector of points of the fitted data, y\_fit. Listing 2.4 and Figure 7 demonstrate the use of both the polyfit and polyval functions. The data used for fitting can be downloaded (linear\_fit\_data.mat) and upon double-clicking the .mat file, the data will be loaded into MATLAB and assigned to the variables x and y. This data is best fitted using a linear or straight-line fit.

Listing 2.4: Using polyfit and polyval for curve-fitting

```
>> coeff = polyfit(x,y,1);
>> y_fit = polyval(coeff,x);
>> plot(x,y,'r+',x,y_fit), grid on, xlabel('x-data'),
ylabel('y-data'), title('Basic curve-fitting'), ...
legend('Original data','Line of best fit','Location','
   SouthEast')
```

# Comments:

- 'r+' plots the x and y data using red crosses.
- You can insert a legend from the Command Window using the legend command, and specifying the text in the legend using strings.

<sup>1</sup> Only basic curve-fitting will be covered in this course.

## 2.3 3D PLOTTING USING PLOT3 AND SURF

MATLAB is hugely powerful and versatile at visualising data in 3D. There are a number of built-in functions for producing different types of 3D plots e.g. points, lines, surfaces and volumes.

The 2D plot function becomes plot3(x,y,z) for plotting points and lines in 3D space. Listing 2.5 and Figure 8 demonstrate using plot3 to plot the points on a helix in 3D space.

Listing 2.5: Using plot3 to plot points on a helix

```
1 >> t = 0:pi/50:10*pi;
2 >> plot3(sin(t),cos(t),t,'r.'), grid on, ...
3 xlabel('x'), ylabel('y'), zlabel('z'), title('3D helix')
```

For plotting surfaces and contours two commonly used functions are surf(x,y,z) and mesh(x,y,z) where x, y, and z are coordinates of points on a surface. Before you use either of these functions you must use the meshgrid function to define a grid of points which the surface will be plotted onto. Listing 2.6 demonstrates the typical use of meshgrid. In this example, assume z = f(x,y) where x is a vector of values (1,2,3,4) and y is a vector of values (5,6,7). meshgrid takes the vectors x and y and returns two matrices, in this case called xx and yy, which contain the coordinates of the grid that the surface z will be plotted onto. Figure 9 shows the coordinates of the points in the matrices returned by the meshgrid function.

Listing 2.6: Using meshgrid

```
>> x = [1 2 3 4];
   >>
      y = [5 6 7];
   >> [xx, yy] = meshgrid(x,y)
                                 4
         1
                 2
                         3
5
                 2
                         3
                                 4
         1
         1
                 2
                         3
                                 4
7
   уу
                 5
                         5
                                 5
         5
         6
                 6
                         6
                                 6
10
         7
                 7
                         7
                                 7
11
```

#### Comments:

- xx is an array consisting of rows of the vector x.
- yy is an array consisting of columns of vector y.
- xx and yy are then used in the calculation of z, and the plotting of the surface.

Listing 2.7 and Figures 10–11 demonstrate using meshgrid in combination with the surface plotting functions surf (creates a colour-filled surface) and mesh (creates a colored mesh) to plot the function:

$$z = c \cdot \sin\left(2\pi a\sqrt{x^2 + y^2}\right),\,$$

where a = 3, c = 0.5,  $-1 \le x \le 1$ , and  $-1 \le y \le 1$ .

Listing 2.7: Plotting a surface

```
>> x = linspace(-1,1,50);
>> y = x;
>> a = 3;
>> c = 0.5;
>> [xx, yy] = meshgrid(x,y);
>> z = c*sin(2*pi*a*sqrt(xx.^2+yy.^2));
>> surf(xx,yy,z), colorbar, xlabel('x'), ylabel('y'), zlabel('z'), ...
>> title('f(x,y)=csin(2\pia\surd(x^2+y^2))')
>> figure;
>> mesh(xx,yy,z), colorbar, xlabel('x'), ylabel('y'), zlabel('z'), ...
>> title('f(x,y)=csin(2\pia\surd(x^2+y^2))')
```

Exercise 4: 3D plotting

- 1. Plot the following 3D curves using the plot3 function:
  - a) Spherical helix

$$x = \sin\left(\frac{t}{2c}\right)\cos(t)$$
$$y = \sin\left(\frac{t}{2c}\right)\sin(t)$$
$$z = \cos\left(\frac{t}{2c}\right)$$

where c = 5 and  $0 \le t \le 10\pi$ .

b) Sine wave on a sphere

$$x = \cos(t)\sqrt{b^2 - c^2\cos^2(at)}$$
$$y = \sin(t)\sqrt{b^2 - c^2\cos^2(at)}$$
$$z = c * \cos(at)$$

where a = 10, b = 1, c = 0.3, and  $0 \le t \le 2\pi$ .

- 2. Plot the following surfaces using the surf function:
  - a) Sine surface

$$x = \sin(u)$$
$$y = \sin(\nu)$$
$$z = \sin(u + \nu)$$

where  $0 \le u \le 2\pi$ , and  $0 \le \nu \le 2\pi$ .

b) Spring

$$x = [1 - r_1 \cos(\nu)] \cos(u)$$
$$y = [1 - r_1 \cos(\nu)] \sin(u)$$
$$z = r_2 \left[ \sin(\nu) + \frac{tu}{\pi} \right]$$

where  $r_1 = r_2 = 0.5$ , t = 1.5,  $0 \le u \le 10\pi$ , and  $0 \le \nu \le 10\pi$ .

c) Elliptic torus

$$x = [c + \cos(\nu)] \cos(u)$$
$$y = [c + \cos(\nu)] \sin(u)$$
$$z = \sin(\nu) \cos(\nu)$$

where  $c = 0.5, -\pi \le u \le \pi$ , and  $0 \le \nu \le \pi$ .



Exercise 4: 3D plotting (continued)

- Use the shading interp command after surf to change the shading type.
- Add a colorbar to the plots.

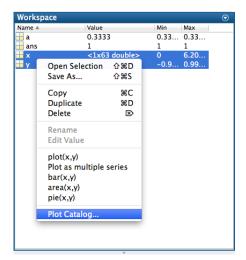


Exercise 4 Solutions

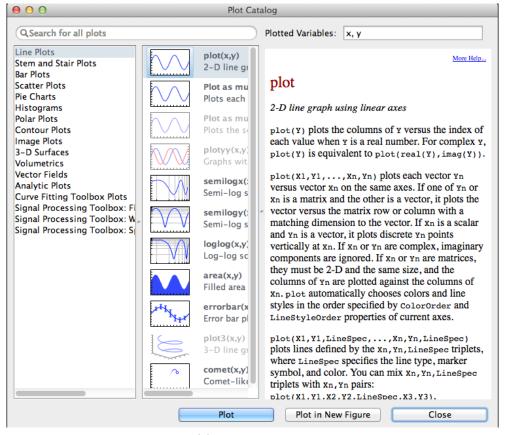
(http://www.eng.ed.ac.uk/teaching/courses/matlab/unit02/Ex4-Solutions.shtml)



Additional Exercises
You should now attempt questions from Chapter C.2.



(a) Accessing the Plot Catalog



(b) The Plot Catalog

Figure 5: The Plot Catalog

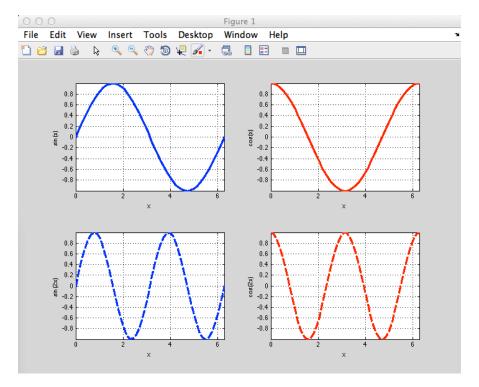


Figure 6: Example of subplots

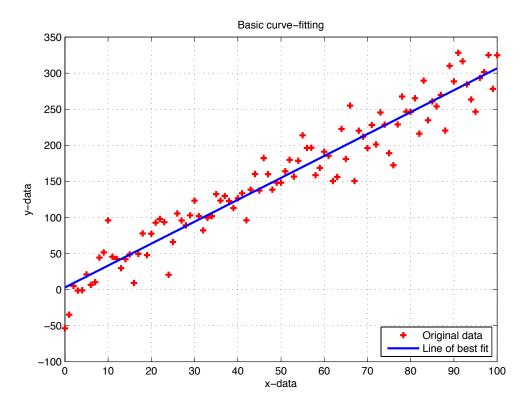


Figure 7: Using polyfit and polyval for curve-fitting

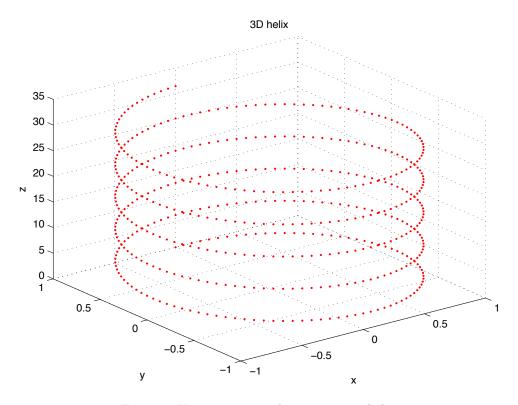


Figure 8: Using plot3 to plot points on a helix

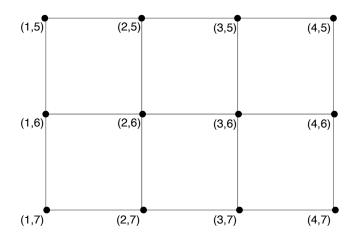


Figure 9: Operation of meshgrid function

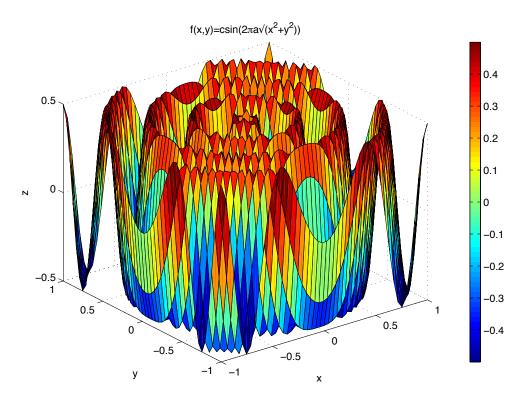


Figure 10: Surface plot (using surf) of the function  $z=c\cdot\sin(2\pi a\sqrt{x^2+y^2})$ 

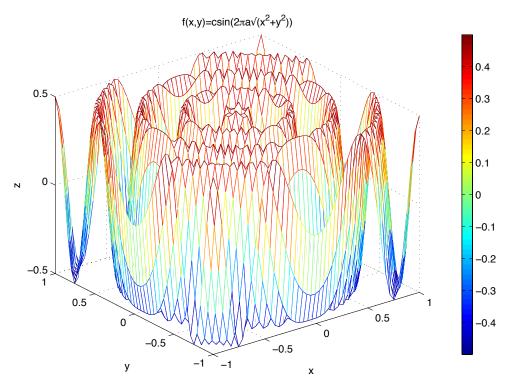


Figure 11: Surface plot (using mesh) of the function  $z = c \cdot \sin(2\pi a \sqrt{x^2 + y^2})$ 

#### 3.1 SCRIPT FILES

A script file is a text file that contains a series of MATLAB commands that you would type at the command prompt. A script file is one type of m-file (.m file extension), the other type being a function file which will be examined in Section 3.2. Script files are useful when you have to repeat a set of commands, often only changing the value of one variable every time. By writing a script file you are saving your work for later use. Script files work on variables in the current workspace, and results obtained from running a script are left in the current workspace.

New script files can be created by clicking on the *New M-File* icon in the MATLAB Window toolbar, shown in Figure 12. This launches the MATLAB Editor with a blank M-File.



Figure 12: New M-File toolbar icon in MATLAB Window

Listing 3.1 presents the commands from Listing 2.7 in the form of a script file. The script file has been saved as  $my\_surf.m$ , and can be run by either typing my\_surf at the command prompt, or clicking the Save and run icon in the Editor Window toolbar, as shown in Figure 13. Copy and paste the example into a new script file, and run it to see the results for yourself.



Figure 13: Save and run toolbar icon in Editor Window

Listing 3.1:  $my\_surf.m$  - Script to plot a surface

```
% my_surf.m
  % Script to plot a surface
  % Craig Warren, 08/07/2010
  % Variable dictionary
         Vectors of ranges used to plot function z
          Coefficients used in function z
  % xx,yy Matrices generated by meshgrid to define points on
      grid
          Definition of function to plot
10
11
  clear all; % Clear all variables from workspace
12
          % Clear command window
13
14
  x = linspace(-1,1,50);
                          % Create vector x
15
16
  y = x;
            % Create vector y
  a = 3;
17
  c = 0.5;
  [xx,yy] = meshgrid(x,y); % Generate xx & yy arrays for
     plotting
  z = c*sin(2*pi*a*sqrt(xx.^2+yy.^2)); % Calculate z (
     function to plot)
  surf(xx,yy,z), xlabel('x'), ylabel('y'), zlabel('z'), ...
  title('f(x,y)=csin(2\pi x^2+y^2))')
     filled-in surface
```

#### Comments:

- It is extremely useful, for both yourself and others, to put comments in your script files. A comment is always preceded with a percent sign (%) which tells MATLAB not to execute the rest of the line as a command.
- Script file names MUST NOT contain spaces (replace a space with the underscore), start with a number, be names of built-in functions, or be variable names.
- It is a good idea to use the clear all and clc commands as the first commands in your script to clear any existing variables from the MATLAB workspace and clear up the Command Window before you begin.



# Writing good scripts

Here are some useful tips that you should follow to make your script files easy to follow and easy to understand by others, or even yourself after a few weeks! a

- Script files should have a header section that identifies:
  - What the program does
  - Who the author is
  - When the program was written or last revised
  - The variable dictionary i.e. a list of all variables their meanings and units
- Use plenty of white space to make your program easy to read.
- Use plenty of comments! In particular define all variables and their units in the variable dictionary.
- Use meaningful names for variables. Don't be afraid of being verbose e.g. use steel\_area in preference to sa.
- Remember to use the clear all and clc commands at the start of your script.
- a Adapted from Patzer [3].



# Creating a simple script

(http://www.eng.ed.ac.uk/teaching/courses/matlab/unit03/simplescript.shtml)



# (2) The input function

The input function is used to request user input and assign it to a variable. For example x = input('Enter a number: '); will display the text Enter a number : in the Command Window and then wait until the user enters something. Whatever is entered will be assigned to the variable x.



# The disp function

The disp function can be used display strings of text to the Command Window e.g. disp('I am a string of text'). You can also display numbers by converting them to strings e.g. disp(num2str(10)). The num2str function simply converts the number 10 to a string that can be displayed by the disp function. You can also combine the display of text and numbers e.g. disp (['Factorial 'num2str(x)' is 'num2str(y)]). Notice the use of spaces to denote the separate elements of the string, and square brackets around the string to concatenate it together.

Exercise 5: Scripts

Write your own script files to solve the following problems:

1. The absolute pressure at the bottom of a liquid store tank that is vented to the atmosphere is given by:

$$P_{\rm abs, bottom} = \rho g h + P_{\rm outside},$$

where:

 $P_{\text{abs,bottom}}$  = the absolute pressure at the bottom of the storage tank (Pa) $\rho = \text{liquid density } (kg/m^3)$  $g = \text{acceleration due to gravity } (m/s^2)$ h = height of the liquid (m) $P_{\text{outside}} = \text{outside atmospheric pressure } (Pa)$ 

Find  $P_{\text{abs,bottom}}$  in SI units if  $\rho = 1000 \text{ kg/m}^3$ ,  $g = 32.2 \text{ ft/s}^2$ , h = 7 yd, and  $P_{\text{outside}} = 1 \ atm.$ 

Here are some tips to help you get started:

- Remember your header section and variable dictionary.
- Use input functions to gather information from the user.
- Convert all units to SI before performing the calculation. Use the following conversion factors:

• Calculate  $P_{\text{abs,bottom}}$ 

[Answers: 164121 Pa]

Example adapted from Patzer [3]



# Exercise 5: Scripts (continued)

2. A pipeline at an oil refinery is carrying oil to a large storage tank. The pipe has a 20 inch internal diameter. The oil is flowing at 5 ft/s and its density is  $57 lb/ft^3$ . What is the mass flow rate of oil in SI units? What is the mass and volume of oil, in SI units, that flows in a 24 hour period? The flow rate of oil is given by:

$$\dot{M} = \rho \nu A$$
,

where:

 $\dot{M} = \text{mass flow rate of oil } (kg/s)$ 

 $\rho = \text{liquid density } (kg/m^3)$ 

 $\nu = \text{flow speed } (m/s)$ 

 $A = \text{cross-sectional area of pipe } (m^2)$ 

[Answers:  $282 \text{ kg/s}, 24362580 \text{ kg}, 26688 \text{ m}^3$ ]

Example adapted from Moore [2]

3. The current in a resistor/inductor circuit is given by:

$$I(t) = \frac{\nu_0}{|Z|} \left[ \cos(\omega t - \phi) - e^{-\frac{tR}{L}} \cos(\phi) \right],$$

where:

$$\omega = 2\pi f$$
,

$$Z = (R + j\omega L),$$

$$\phi = \tan^{-1} \left( \frac{\omega L}{R} \right),\,$$

and where:

 $\nu_0 = \text{voltage } (V)$ 

 $\omega = \text{angular frequency } (rads/s)$ 

 $R = \text{resistance} (\Omega)$ 

L = inductance (H)

Find and plot I(t) if  $\nu_0 = 230 V$ , f = 50 Hz,  $R = 500 \Omega$ , and L = $650 \ mH.$ 

• You'll need to explore different values of t to find one that best plots the behaviour of the current.



# Exercise 5 Solutions

(http://www.eng.ed.ac.uk/teaching/courses/matlab/unit03/Ex5-Solutions.shtml)



# The abs function

The abs function can be used to calculate the absolute value or magnitude of a number.

#### 3.2 FUNCTIONS

Another type of m-file (.m file extension) is a function file. Functions are similar to scripts, except that the variables in a function are only available to the function itself i.e. are local to the function. This is in contrast with script files, where any variables you define exist in the Workspace (are global) and can be used by other scripts and commands. You have used many of the built-in functions in MATLAB e.g. size, plot, surf etc..., and as you become more familiar with MATLAB you will learn to write your own functions to perform specific tasks.

A function file always begins with a function definition line. This specifies the input and output variables that the function will use, and defines a name for the function. Listing 3.2 presents the syntax of a function definition line, and Table 5 gives some examples.

Listing 3.2: Syntax of a function definition

```
function [outputVariables] = functionName (inputVariables)
  % Comments describing function and variables
2
  commands
```

#### Comments:

- The first word, function, is mandatory, and tells MATLAB this m-file in a function file.
- On the lefthand side of the equals sign is a list of the output variables that the function will return. You will notice when there is more than one output variable, that they are enclosed in *square* brackets.
- On the righthand side of the equals sign is the name of the function. You must save your function file with the same name that you use here.
- Lastly, within the round brackets after the function name, is a comma separated list of the input variables.
- It is good practice to put some comments after the function definition line to explain what task the function performs and how you should use the input and output variables. This is in addition to comments you would usually include at the top of a script file.

Functions are executed at the command prompt by typing their function definition line without the function command. Listing 3.3 demonstrates how you would execute the motion function from Table 5.

The name used to save a function file must match the function name.

	ď	
-	Ξ	27.75
-	_	Š
	5	4
	٤	4
	σ	3
	2	•
	+	0
	Ξ	Š
		)
-	į.	2
	Ξ	3
	5	2
-	,	٠
	$\succeq$	4
	Ξ	j
	כבת	3113
-	۲	0
	Ξ	Š
	$\subseteq$	)
	Ξ	4
	Ξ	ì
	ď.	2
	ق	4
	⊱	j
	7	₹
		4
	'n	3
5	_	4
C	F	20112011
	U.	2
	₽	i
	711101	)
•		5
•	Ξ	۹
	₽	i
C	F	4
	٩	
-	C	
	_	4
	Έ	į
	$\succeq$	4
	÷	٥
	٥	2
	٤	j
	Ξ	3
ļ	1906 S. HIINCTION	4
1	۲.	)
	_	
-	4	4
_	$\subset$	2
	σ	3
F	_	ĺ

FUNCTION DEFINITION	FILENAME	INPUT VARIABLES	FILENAME INPUT VARIABLES OUTPUT VARIABLES NOTES	NOTES
<pre>function [rho, H, F] = motion(x, y, t)</pre>	motion.m	x, y, t	rho, H, F	
<pre>function [theta] = angleTH(x, y)</pre>	angle TH.m	х, у	theta	
			:	If there is only one output variable the
inction theta = $IHEIA(x, y)$	інь ім.т	х, у	theta	square brackets can
				be omitted
function $[] = circle(r)$	circle.m	Ы	None	
				If there are no
				output variables the
function circle(r)	circle.m	Я	None	square brackets and
				the equals sign
				can be omitted

# Listing 3.3: Executing a function

```
>> [rho, H, F] = motion(x, y, t)
```

# Comments:

- Input variables x, y, t must be defined in the workspace before you execute the function. This is because variables defined within a function file are local to the function, i.e. do not exist in the workspace.
- When you execute the function the names for the input and output variables do not have to match those used in the function file.

Listings 3.4 presents an example of a simple function that multiplies two numbers, x and y, together to calculate an area. Listing 3.5 demonstrates how to execute this function in the command window.

Listing 3.4: A simple function

```
function area = calculateArea(x, y)
\% Function to calculate an area given two lengths (x, y)
area = x*y;
```

Listing 3.5: Execution of a simple function

```
>> x = 5; y = 10;
  >> area = calculateArea(x, y)
3
  area
      50
4
```

The same function could also be executed using variables with different names, as shown in Listing 3.6.

Listing 3.6: Execution of a simple function

```
>> length1 = 25; length2 = 100;
>> myArea = calculateArea(length1, length2)
myArea =
      2500
```



# Creating a function

(http://www.eng.ed.ac.uk/teaching/courses/matlab/unit03/simplefunction.shtml)



Exercise 6: Functions

Write your own functions to solve the following problems:

1. Produce a conversion table for Celsius and Fahrenheit temperatures. The input to the function should be two numbers:  $T_{lower}$  and  $T_{upper}$  which define the lower and upper range, in Celsius, for the table. The output of the function should be a two column matrix with the first column showing the temperature in Celsius, from  $T_{lower}$  and  $T_{upper}$  with an increment of 1 °C, and the second column showing the corresponding temperature in Fahrenheit.

Here are some tips to help you get started:

- Start with a function definition line. What are your input and output variables?
- Create a column vector to hold the range Celsius = [T\_lower: T\_upper],
- Calculate the corresponding values in Fahrenheit using Fahrenheit = 9/5 \* Celsius + 32
- Create a matrix to hold the table using temp\_table = [Celsius Fahrenheit]

Test your function for  $T_{\text{lower}} = 0 \, ^{\circ}C$  and  $T_{\text{upper}} = 25 \, ^{\circ}C$ .

2. The angles of cosines of a vector in 3D space are given by:

$$\cos(\alpha_j) = \frac{a_j}{|a|}, \text{ for } j = 1, 2, 3$$

Given the magnitude, |a|, and angles of cosines,  $\alpha_j$ , calculate the Cartesian components,  $a_i$ , of the vector.



Exercise 6 Solutions

(http://www.eng.ed.ac.uk/teaching/courses/matlab/unit03/Ex6-Solutions.shtml)



Additional Exercises

You should now attempt questions from Chapter C.3.

# DECISION MAKING

All the problems you have solved so far have been problems with a *straight-line* logic pattern i.e. you followed a sequence of steps (defining variables, performing calculations, displaying results) that flowed directly from one step to another. Decision making is an important concept in programming and allows you to control which parts of your code should execute depending on certain conditions. This flow of control in your program can be performed by branching with if and else statements, which will be discussed in this chapter, or looping, which will be discussed in Chapter 5.

# 4.1 RELATIONAL AND LOGICAL OPERATIONS

Relational and logical operators are used in branching and looping to help make decisions. The result of using a relational or logical operator will always be either true, given by a 1, or false, given by a 0. Tables 6–7 list the most common relational and logical operators in MATLAB.

Table 6: Relational operators

OPERATOR	MATHEMATICAL SYMBOL	MATLAB SYMBOL
Equal	=	==
Not equal	$\neq$	~=
Less than	<	<
Greater than	>	>
Less than or equal	<u>≤</u>	<=
Greater than or equal	≥	>=

Table 7: Logical operators

OPERATOR	MATHEMATICAL SYMBOL	MATLAB SYMBOL
And	AND	&
Or	OR	
Not	NOT	~

Listings 4.1 presents a simple example of using relational operators.

Listing 4.1: Simple relational operators

# Comments:

• Lines 3 and 6 are called logical expression because the result can only be either true, represented by 1, or false, represented by 0.

Listings 4.2–4.3 present more examples of using relational and logical operators.

Listing 4.2: Relational operators

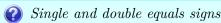
```
[1 5 3 7];
   \Rightarrow y = [0 2 8 7];
2
   >> k = x < y
3
   k =
4
              0
                                   0
                     0
                            1
5
   >> k = x <= y
6
7
   k =
              0
                     0
                                   1
                            1
8
   >> k = x>y
   k =
10
                                   0
              1
                     1
                            0
11
   >> k = x>=y
12
   k =
13
              1
                     1
                                   1
14
   >> k = x == y
15
   k =
16
              0
                     0
                            0
                                   1
17
   >> k = x~=y
18
   k =
19
                                   0
20
```

Listing 4.3: Logical operators

```
9 >> k = ~((x>y) | (x>4))
10 k =
11 0 0 1 0
```

# Comments:

- The relational and logical operators are used to compare, element-byelement, the vectors  $\mathbf{x}$  and  $\mathbf{y}$ .
- The result of each comparison is a logical vector i. e. k only contains 1's and 0's (corresponding to true or false).



The difference between = and == is often misunderstood. A single equals sign is used to assign a value to a variable e.g. x=5. A double equals sign is used to test whether a variable is equal to given value e.g.  $my_test=(x==5)$  means test if x is equal to 5, and if so assign the value 1 (true) to  $my_test$ .



- 1.  $^2$ Evaluate the following expressions without using MATLAB. Check your answer with MATLAB.
  - a) 14 > 15/3
  - b)  $y = 8/2 < 5 \times 3 + 1 > 9$
  - c)  $y = 8/(2 < 5) \times 3 + (1 > 9)$
  - d)  $2+4\times3 \sim = 60/4-1$
- 2.  $^2$ Given: a=4, b=7. Evaluate the following expressions without using MAT-LAB. Check your answer with MATLAB.
  - a)  $y = a + b >= a \times b$
  - b)  $y = a + (b >= a) \times b$
  - c) y = b a < a < a/b
- 3.  $^2$ Given: v=[4 -2 -1 5 0 1 -3 8 2], and w=[0 2 1 -1 0 -2 4 3 2]. Evaluate the following expressions without using MATLAB. Check your answer with MATLAB.
  - a)  $v \le w$
  - b) w = v
  - c) v < w + v
  - d) (v < w) + v

<sup>2</sup> Question adapted from Gilat, A. (2008). MATLAB: An Introduction With Applications. John Wiley & Sons, Inc., 3rd edition. Copyright ©2008 John Wiley & Sons, Inc. and reprinted with permission of John Wiley & Sons, Inc.

# 4.2 THE IF-ELSE STATEMENT

The if, else, and elseif statements in MATLAB provide methods of controlling which parts of your code should execute based on whether certain conditions are true or false. The syntax of the simplest form of an if statement is given in Listing 4.4.

Listing 4.4: Syntax of an if statement

```
if logical_expression
statements
end
```

# Comments:

- Line 1 contains the if command, followed by an expression which must return true or false.
- Line 2 contains the body of the if statement which can be a command or series of commands that will be executed if the logical expression returns true.
- Line 3 contains the end command which must always be used to close the if statement.
- If the logical expression returns true MATLAB will execute the statements enclosed between if and end. If the logical expression returns false MATLAB will skip the statements enclosed between if and end and proceed with any following code.

Listing 4.5 presents a very simple example of using an if statement to test if a user has entered a number greater than 10.

Listing 4.5:  $basic\_if.m$  - Script to show simple if statement

```
% basic if.m
  % Script to show simple if statement
  %
3
  % Craig Warren, 08/07/2010
5
  % Variable dictionary
         Variable to hold entered number
7
  clear all; % Clear all variables from workspace
9
          % Clear command window
10
11
  x = input('Enter a number: ');
                                      % Get a number from the
12
          % Test if x is greater than
  if x>10
13
14
      disp('Your number is greater than 10')
  end
15
```

#### Comments:

- On Line 12 the input command is used, which prompts the user for input with the request Enter a number: and assigns the number entered to the variable x.
- On Line 13 an if statement is used with the logical expression x>10. If this expression is true then the text Your number is greater than 10 is displayed, otherwise if the expression is false nothing is executed.

The else and elseif commands can be used to apply further conditions to the if statement. Listing 4.6 presents the syntax of these commands.

Listing 4.6: Syntax of an if statement with else and elseif

# Comments:

- On Line 3 the logical expression associated with the elseif command will only be evaluated if the preceding logical expression associated with the if command returns false.
- Notice that the else command on Line 5 has no associated logical expression. The statements following the else command will only be executed if all the logical expressions for the preceding elseif and if commands return false.

Listing 4.7 presents a simple of example of decision making using the if, else, and elseif functions. Copy and paste the example into a new script file, and run it to see the results for yourself.

Listing 4.7:  $number\_test.m$  - Script to test sign and magnitude of numbers

```
% number_test.m
  % Script to test sign and magnitude of numbers
2
3
  % Craig Warren, 08/07/2010
4
  % Variable dictionary
6
         Variable to hold entered number
7
  clear all; % Clear all variables from workspace
9
         % Clear command window
10
11
                                    % Get a number from the
  x = input('Enter a number: ');
12
     user
  if x<0 % Test if x is negative
13
      disp('Your number is a negative number')
14
  elseif x<100
                   % Otherwise test if x is less than 100
15
      disp('Your number is between 0 and 99')
16
        % Otherwise x must be 100 or greater
^{17}
      disp('Your number is 100 or greater')
18
  end
```

#### Comments:

- On Line 12 the input command is used, which prompts the user for input with the request Enter a number: and assigns the number entered to the variable x.
- On Lines 14, 16, and 18 the disp command is used, which simply displays text to the Command Window.

# The if-else statement

(http://www.eng.ed.ac.uk/teaching/courses/matlab/unit04/if-elsestatement.shtml)

The following example is solved in the screencast:

 $Water\ level\ in\ a\ water\ tower^{\dagger}$ 

The tank in a water tower has the geometry shown in Figure 14 (the lower part is a cylinder and the upper part is an inverted frustum cone). Inside the tank there is a float that indicates the level of the water. Write a user-defined function that determines the volume of water in the tank from the position (height) of the float. The volume for the cylindrical section of the tank is given by:

$$V = \pi \cdot 12.5^2 \cdot h$$

The volume for the cylindrical and conical sections of the tank is given by:

$$V = \pi \cdot 12.5^2 \cdot 19 + \frac{1}{3}\pi(h - 19)(12.5^2 + 12.5r_h + r_h^2),$$
 where  $r_h = 12.5 + \frac{10.5}{14}(h - 19)$ 

$$[h = 8 m, V = 3927 m^3; h = 25.7 m, V = 14115 m^3]$$

† Question adapted from Gilat, A. (2008). MATLAB: An Introduction With Applications. John Wiley & Sons, Inc., 3rd edition. Copyright ©2008 John Wiley & Sons, Inc. and reprinted with permission of John Wiley & Sons, Inc.

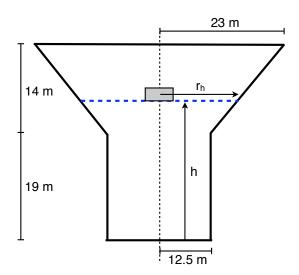


Figure 14: Water level in a water tower

Self Test Exercise: The if-else statement Evaluate the following expressions without using MATLAB.

1. Which of the following shows a correct if, else statement?

2. What will the following code print?

```
a = 10;
if a ~= 0

disp('a is not equal to zero')
end
```

3. What will the following code print?

```
a = 10;
if a > 0

disp('a is positive')

else

disp('a is not positive')
end
```

4. <sup>2</sup>What will the following code print?

5. <sup>2</sup>For what values of the variable will the following code print Hello world ?

6.  $^2$ For what values of the variable will the following code print Hello world  $^7$ 

```
if a < 7 | a >= 3
disp('Hello world')
selse
disp('Goodbye world')
send
```

<sup>2</sup> Questions from Morrell, D., *Programming with M-files: If-Statement Drill Exercises*, Connexions, <a href="http://cnx.org/content/m13432/1.4/">http://cnx.org/content/m13432/1.4/</a>, [Last assessed: Nov 2011]

# Exercise 7: Decision making

Write your own script files to solve the following problems:

- 1. Write a script file that asks the user for the input of a number and returns the natural logarithm of the number if the number is positive, and displays an error message otherwise.
- 2. The cost per kilometre for a rental car is £0.50 for the first 100 kilometres, £0.30 for the next 200 kilometres and £0.20 for all kilometres in excess of 300 kilometres. Write a function that determines the total cost for a given number of kilometres.
- 3. Write a function to evaluate f(x, y) for any two user specified values x and y. The function f(x, y) is defined as:

$$f(x,y) = \begin{cases} x+y & x \ge 0 \text{ and } y \ge 0\\ x+y^2 & x \ge 0 \text{ and } y < 0\\ x^2+y & x < 0 \text{ and } y \ge 0\\ x^2+y^2 & x < 0 \text{ and } y < 0 \end{cases}$$

4. The energy loss due to fluid flow through a pipe can be calculated using the following equations:

$$h_L = f\left(\frac{L}{D}\right)\left(\frac{V^2}{2}\right), \quad V = \frac{Q}{A}, \quad A = \frac{\pi D^2}{4}, \quad Re = \frac{DV\rho}{\mu},$$

where:

 $h_L = \text{energy loss per mass of fluid flowing } (J/kg)$ 

f = friction factor (dimensionless)

L = pipe length (m)

D = pipe diameter (m)

V =average fluid velocity (m/s)

 $Q = \text{volumetric flow rate } (m^3/s)$ 

 $A = \text{pipe cross-sectional area } (m^2)$ 

Re = Reynolds number (dimensionless)

 $\rho = \text{fluid density } (kg/m^3)$ 

 $\mu = \text{fluid viscosity } (kg/ms)$ 



**X** Exercise 7: Decision making (continued)

4. (continued) The friction factor, f, is calculated as:

$$f = \begin{cases} \frac{64}{Re} & \text{when } Re \le 2000\\ \left[ -2.01 \cdot ln \left[ \frac{-5.0452}{Re} ln \left( \frac{5.8506}{Re^{0.8981}} \right) \right] \right]^{-2} & \text{when } Re > 2000 \end{cases}$$

Write a function that calculates the energy loss per mass of flowing fluid for a fluid flow in a pipe, given the pipe diameter, pipe length, fluid volumetric flow rate, fluid density, and fluid viscosity (all in SI units). Test your function with: D = 0.2 m, L = 10 m,  $Q = 1 m^3/s$ ,  $\rho = 1000 \ kg/m^3 \ \text{and} \ \mu = 0.001 \ kg/ms.$ 

[Answer:  $47.0948 \ J/kg$ ]



Exercise 7 Solutions

(http://www.eng.ed.ac.uk/teaching/courses/matlab/unit04/Ex7-Solutions.shtml)



Additional Exercises

You should now attempt questions from Chapter C.4.



Advanced Topic

If you are interested, read about the switch statement in Appendix A.

# LOOPS

Loops are another way of altering the flow of control in your program, and provide methods for repeatedly executing commands. You might want to repeat the same commands, changing the value of a variable each time, for a fixed number of iterations. Alternatively, you might want to repeat the same commands, changing the value of a variable each time, continually until a certain condition is reached. Two of the most common types of loops, for and while, will be examined in this chapter.

#### 5.1 FOR LOOPS

A for loop is used to repeat a command, or set of commands, a fixed number of times. Listing 5.1 shows the syntax of a for loop.

Listing 5.1: Syntax of a for loop

```
for variable = f:s:t
statements
end
```

#### Comments:

• Line 1 contains the for command, followed by the loop counter variable which is defined by an expression. f is the value of the loop counter on the first iteration of the loop, s is the step size or increment, and t is the value of the loop counter on the final iteration of the loop. As an example if the loop counter is defined by the expression n = 0:5:15, which means n = [0 5 10 15], on the first iteration of the loop n = 0, on the second iteration n = 5, on the third iteration n = 10, and on the forth, and final, iteration n = 15.

Refer to Chapter 1 if you need a refresher on ranges

- Line 2 contains the body of loop which can be a command or series of commands that will be executed on each iteration of the loop.
- Line 3 contains the end command which must always be used at the end of a loop to close it.

# (?) Indenting for readability

It is good practice to indent the body of loops for readability in your script files. MATLAB will usually do this for you, but if not, highlight your code and choose *Smart Indent* from the *Text* menu in the Editor Window toolbar.

Listing 5.2 gives an example of a simple for loop which displays the value of the variable x.

Listing 5.2: Simple example of a for loop

```
>> for x=1:1:9
2
   x
   end
3
          1
          2
9
10
11
12
13
15
16
17
20
          9
```

# Comments:

- In Line 1 the loop counter variable, in this case x, is defined to start at 1 and count up in steps of 1 until 9.
- In Line 2 the body of the loop prints the value of the loop counter variable x.
- When the loop is executed, initially the value of 1 is assigned to x, and then the body of the loop is executed. The value of x is then incremented by 1, the body of the loop is executed again, and so on until x is 9. Whereupon, the body of the loop is executed for a final time and then the loop terminates.
- Lines 4–21 contain the results of running the for loop.



The for loop

(http://www.eng.ed.ac.uk/teaching/courses/matlab/unit05/for-loop.shtml)



Self Test Exercise: for loops

Evaluate the following expressions without using MATLAB. Check your answers with MATLAB.

1. How many times will this code print Hello World?

```
for a=0:50
           disp('Hello World')
2
  end
3
```

2. How many times will this code print Guten Tag Welt?

```
for a=-1:-1:-50
disp('Guten Tag Welt')
end
```

3. How many times will this code print Bonjour Monde?

```
for a=-1:1:50
disp('Bonjour Monde')
end
```

4. How many times will this code print Hola Mundo?

```
for a=10:10:50
for b=0:0.1:1
disp('Hola Mundo')
end
end
end
```

Listing 5.3 demonstrates an example of using a for loop to take the sum of a geometric series (the same example posed in Exercise 2, Question 8).

Listing 5.3: for\_loop\_sum.m - Script to sum a geometric series using a for loop

```
% for loop sum.m
  % Script to sum a geometric series using a for loop
3
  % Craig Warren, 01/09/2011
  % Variable dictionary
         Number of terms to sum
7
             Sum of geometric series
  % my_sum
         Constant (set to 0.5 for this example)
  % m
         Loop counter
10
11
  clear all; % Clear all variables from workspace
12
         % Clear command window
13
14
  n = input('Enter the number of terms to sum: ');
15
  my_sum = 0;
16
r = 0.5;
  for m = 0:n
18
      my_sum = my_sum + r^m;
20
                 % Sets display format to 15 digits
^{21}
  format long
  my_sum
```

# Comments:

• Line 15 contains the <u>input</u> command, which is used to get the number of terms to be summed from the user.

- On Line 16 a variable my\_sum is created (and set to zero) to hold the sum of the geometric series. It is necessary to create any variables outside of loops before using them within loops.
- Lines 18–20 contain the for loop. The loop counter m counts in steps of one from zero until the number of terms specified by the user n.
- On Line 19 (the body of the loop) the new term in the sum r^m is added to the previous value of my\_sum and this becomes the new value of my\_sum.
- Lines 21–22 display the result of the summation. The format command is used to set the display to 15 digits instead of the default 4 digits so that the result of taking more terms in the summation can be seen.

#### 5.2 WHILE LOOPS

A while loop is similar to for loop in that it is used to repeat a command, or set of commands. Listing 5.4 shows the syntax of a while loop. The key difference between a for loop and a while loop is that the while loop will continue to execute until a specified condition becomes false.

Listing 5.4: Syntax of a while loop

```
while condition is true
statements
end
```

# Comments:

- Line 1 contains the while command, followed by a condition e.g. x>10. This means as long as the condition, x>10, remains true the loop will continually repeat.
- Line 2 contains the body of loop which can be a command or series of commands that will be executed on each iteration of the loop.
- Line 3 contains the end command which must always be used at the end of a loop to close it.

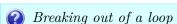
Listing 5.5 gives an example of a simple while loop which displays the value of the variable x.

Listing 5.5: Simple example of a while loop

```
2
10
             3
11
12
             4
13
14
             5
15
16
             6
17
18
19
20
             8
21
             9
23
```

# Comments:

- Line 1 assigns the value of 1 to the variable x. Notice this is outside of the while loop. If you don't do this you will get an error because you are testing whether x<10 but x has never been defined.
- In Line 2 the condition, x<10, is specified. In this case the loop will continue to repeat as long as x is less than 10. As soon as x is equal to 10 execution of the loop is stopped.
- Lines 3-4 contain the body of the loop, in this case the value of the loop counter variable x is printed. Then the value of x is incremented by 1. The value of x must be explicitly incremented otherwise x will always be equal to 1, the condition x<10 will always be true, and the loop will therefore execute continuously.
- Lines 6-23 contain the results of running the while loop.



If you end up stuck in an infinitely repeating loop use CTRL + C to force MATLAB to break out of the loop. However, under certain conditions you may want your code to break out of a loop before it is finished. To do this you can use the break command. Statements in your loop after the break command will not be executed.



The while loop

(http://www.eng.ed.ac.uk/teaching/courses/matlab/unit05/whileloop.shtml)



M Self Test Exercise: while loops

Evaluate the following expressions without using MATLAB. Check your answers with MATLAB.

Remember to define variables you use in loops before you start the loops themselves.

1. How many times will this code print Hello World?

```
n = 10;
while n > 0
disp('Hello World')
n = n - 1;
end
```

2. How many times will this code print Hello World?

```
n = 1;
while n > 0
disp('Hello World')
n = n + 1;
end
```

3. What values will this code print?

```
a = 1
while a < 100
a = a*2
end
```

4. What values will this code print?

Listing 5.6 demonstrates an example of using a while loop to take the sum of a geometric series (the same example posed in Exercise 2, Question 8). Compare Listings 5.3 and 5.6.

Listing 5.6: while\_loop\_sum.m - Script to sum a geometric series using a while loop

```
% while_loop_sum.m
2
  % Script to sum a geometric series using a while loop
3
  % Craig Warren, 01/09/2011
  % Variable dictionary
6
         Number of terms to sum
            Sum of geometric series
  % my_sum
          Constant (set to 0.5 for this example)
  % r
10
          Loop counter
11
  clear all; % Clear all variables from workspace
12
  clc;
           % Clear command window
13
14
  n = input('Enter the number of terms to sum: ');
15
  my_sum = 0;
  r = 0.5;
17
  m = 0;
18
  while m <= n
19
      my_sum = my_sum + r^m;
20
      m = m + 1;
21
  end
22
                  % Sets display format to 15 digits
  format long
23
24 my_sum
```

# Comments:

- Line 18 contains a variable m (defined outside of the loop) which is used as a loop counter.
- Lines 19–22 contain the while loop. The condition for the loop to execute is that the value of the loop counter m must be less than or equal to the number of terms to be summed n. When this condition becomes false the loop will terminate.
- On Line 20 the summation is performed, and on Line 21 the loop counter is incremented.

Which loop to use: for or while?

Well it depends on the problem! Several of the code listings in this chapter have demonstrated that the same problem can be solved using either a for or while loop.



**Exercise** 8: Loops

Write your own scripts to perform the following tasks:

- a) A for loop that multiplies all even numbers from 2 to 10.
  - b) A while loop that multiplies all even numbers from 2 to 10.
- a) A for loop that assigns the values 10, 20, 30, 40, and 50 to a vector.
  - b) A while loop that assigns the values 10, 20, 30, 40, and 50 to a vector.
  - c) Is there a simpler way to do this avoiding loops?
- 3. Given the vector  $x=[1\ 8\ 3\ 9\ 0\ 1]$  use a for loop to:
  - a) Add up the values of all elements in x.
  - b) Compute the cumulative sum, i.e 1, 9, 12, 21, 21, 22, of the elements

You can check your results using the built-in functions sum and cumsum.

4. The factorial of a non-negative integer is defined as:

$$n! = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 1,$$

where n! = 1 when n = 0. For example,  $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$  which is 120.

Use a for loop to compute and print factorials. You should prompt the user for a non-negative integer and check it is indeed non-negative. There is a built-in function called factorial, therefore you should use a different name for your script to avoid any confusion.

5. Use a while loop to determine and display the number of terms that it takes for the series,

$$S_N = \sum_{n=1}^N \frac{1}{n^2},$$

to converge to within 0.01% of its exact value, which is  $S_{\infty} = \frac{\pi^2}{6}$ .



Exercise 8 Solutions

(http://www.eng.ed.ac.uk/teaching/courses/matlab/unit05/Ex8-Solutions.shtml)



Additional Exercises

You should now attempt questions from Chapter ??.



② Advanced Topic
If you are interested, read about vectorisation in Appendix B.

\*\*



### ADVANCED TOPIC: THE SWITCH STATEMENT

The switch statement in MATLAB is similar to the if, else, and elseif statements, and provides a method of executing different bits of code dependent on which case is true. Typically you would use a switch statement in preference to if, else, and elseif statements if you have specific cases (values of a variable) to test. Listing A.1 presents the syntax of the switch statement.

Listing A.1: Syntax of a switch statement

```
switch switch_expression
case case_expression1
statements
case case_expression2
statements
case case_expression3
statements
...
end
```

#### Comments:

- Line 1 contains the switch command followed by the expression to be compared to the various cases.
- Lines 2–7 contain the different cases and their corresponding statements to be executed.
- If the switch expression matches one of the case expressions then the statements underneath that case will be executed and the switch statement will end.

Listing A.2 presents a simple example of the usage of a switch statement. MATLAB has a built-in function called computer that returns a string of text corresponding to the operating system that you are running. By analysing this string you can print out some information about the operating system. Copy and paste the example into a new script file, and run it to see the results for yourself.

Listing A.2:  $computer\_test.m$  - Script to test type of computer MATLAB is running on

```
% computer_test.m
  % Script to test type of computer MATLAB is running on
2
4 % Craig Warren, 08/07/2010
  % Variable dictionary
6
                  Variable to hold information about
  % myComputer
     computer
8
  clear all; % Clear all variables from workspace
9
         % Clear command window
10
11
                          % Assign result of built-in
  myComputer = computer;
      computer
13
                              % function to myComputer
                                 variable
  switch myComputer
      case 'PCWIN'
15
           disp('This computer is running Microsoft Windows,
              32-bit');
       case 'PCWIN64'
^{17}
           disp('This computer is running Microsoft Windows,
18
              64-bit');
      case 'GLNX86'
19
           disp('This computer is running Linux, 32-bit');
20
      case 'GLNXA64'
^{21}
           disp('This computer is running Linux, 64-bit');
       case 'MACI'
23
           disp('This computer is running Mac OS X, Intel,
^{24}
              32-bit');
      case 'MACI64'
25
           disp('This computer is running Mac OS X, Intel,
              64-bit');
       case 'SOL64'
27
           disp('This computer is running Sun Solaris, 64-bit
28
              ');
  end
29
```

#### ADVANCED TOPIC: VECTORISATION

To make your MATLAB code run faster it is important to vectorise, where possible, the algorithms that you use. Where loops, especially nested loops, are being used, it is often possible to substitute a vector or matrix equivalent which will run much faster.

Listing B.1 presents a simple example of a for loop being used to calculate a series of voltages across different resistors, with a different current flowing in each resistor.

Listing B.1: Simple for loop to vectorise

```
I = [2.35 2.67 2.78 3.34 2.10]; % Vector of currents
R = [50 75 100 125 300]; % Vector of resistances
for n = 1:5
    V(n) = I(n) * R(n); % Calculate voltage
end
```

Listing B.2 presents a vectorised solution to the same problem, and indeed you may well have gone straight to the vectorised solution without considering use of a loop.

Vectorise your loops to make your code run faster!

```
Listing B.2: Vectorised for loop
```

```
I = [2.35 2.67 2.78 3.34 2.10]; % Vector of currents
R = [50 75 100 125 300]; % Vector of resistances
V = I.*R; % Calculate voltage
```

Listings B.3–B.4 present a more advanced example of vectorisation. A matrix of random numbers called data is generated and two nested for loops are used to iterate through every element in the matrix. An if statement is used to check to see if each element is less than 0.1, and if so that element is set to zero. Copy and paste the example into a new script file, and run it to see the results for yourself.

Listing B.3: Nested loops

```
data=rand(8000,8000); % Generate some random data
  tic
2
  for i=1:size(data,1)
3
          for j=1:size(data,2)
                   if data(i,j)<0.1 % Is data sample is less
                      than 0.1?
                           data(i,j)=0; % Set data sample to
6
                              zero
                   end
          end
  end
9
 Elapsed time is 6.876476 seconds.
```

#### Comments:

- On Line 1 the rand function is used to generate a matrix,  $8000 \times 8000$ , of uniformly distributed random numbers in the interval [0,1].
- On Line 2 and Line 10 the tic and toc commands are used to time how long the code took to execute. Line 11 lists the result.

The code in Listing B.3 can be vectorised to produce the code given in Listing B.4.

Listing B.4: Vectorisation of nested loops

```
data=rand(8000,8000);
tic
data(data<0.1)=0;
toc
Elapsed time is 0.927503 seconds.</pre>
```

#### Comments:

- On Line 3 the nested for loops have been replaced with a single line of vectorised code. data<0.1 returns a matrix of 1's and 0's corresponding to values of data less than 0.1. The values of these elements are then set to 0.
- Line 5 lists the time taken to execute the vectorised code. A difference of approximately 5 seconds may not seem like much of a speed increase, but in complex MATLAB scripts with lots of loops performing many iterations it is can be significant.

Listings B.3–B.4 are an extreme example of vectorisation, but clearly demonstrate that MATLAB can execute vectorised code much faster than conventional loops.

## ADDITIONAL EXERCISES

The aim of this chapter is to provide more realistic problems that can be solved using MATLAB. The questions are adapted from Gilat, A. (2008). *MATLAB: An Introduction With Applications*. John Wiley & Sons, Inc., 3rd edition. Copyright ©2008 John Wiley & Sons, Inc. and reprinted with permission of John Wiley & Sons, Inc.

## C.1 BASIC CONCEPTS

#### 1. Variables

An object with an initial temperature of  $T_0$  that is placed at time t = 0 inside a chamber that has a constant temperature of  $T_s$ , will experience a temperature change according to the equation:

$$T = T_s + (T_0 - T_s)e^{-kt},$$

where T is the temperature of the object at time t, and k is a constant. A soda can at a temperature of 49 °C (was left in the car) is placed inside a refrigerator where the temperature is 3 °C. Determine, to the nearest degree, the temperature of the can after three hours. Assume k=0.45. First define all the variables and then calculate the temperature using one MATLAB command.

## 2. Variables

Radioactive decay is modeled with the exponential function  $f(t) = f(0)e^{kt}$ , where t is time, f(0) is the amount of material at t = 0, f(t) is the amount of material at time t, and k is a constant. Gallium-67, which has a half-life of 3.261 days, is used for tracing cancer. If 100 mg are present at t = 0, determine the amount that is left after 7 days. You should first determine the constant k and then calculate f(7).

## 3. Variables

The magnitude M of an earthquake on the Richter scale is given by:

$$M = \frac{2}{3}log_{10}\left(\frac{E}{E_0}\right),\,$$

where E is the energy release by the earthquake, and  $E_0 = 10^{4.4}$  J is a constant (the energy of a small reference earthquake). Determine how many more times energy is released from an earthquake that registers 7.1 on the Richter scale than an earthquake that registers 6.9.

#### 4. Variables

The temperature dependence of vapor pressure p can be estimated by the Antoine equation:

$$ln(p) = A - \frac{B}{C+T},$$

where ln is the natural logarithm, p is in mm Hg, T is in Kelvin, and A, B, and C are material constants. For toluene ( $C_6H_5CH_3$ ) in the temperature range from 280 to 410 K the material constants are: A = 16.0137, B = 3096.52, and C = -53.67. Calculate the vapour pressure of toluene at 315 and 405 K.

## 5. Adding vectors

Three forces are applied to a bracket as shown in Figure 15. Determine the total (equivalent) force applied to the bracket.

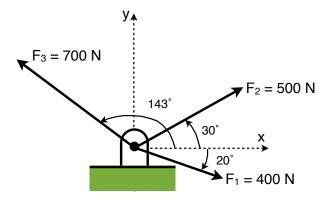


Figure 15: Forces on a bracket

# Addition of forces in 2D space

A force is a vector (physical quantity that has magnitude and direction). In a Cartesian coordinate system a 2D vector  $\mathbf{F}$  can be written as:

$$F = F_x i + F_u j = F\cos(\theta) i + F\sin(\theta) j = F(\cos(\theta) i + \sin(\theta) j),$$

where F is the magnitude of the force, and  $\theta$  is its angle relative to the x axis,  $F_x$  and  $F_y$  are the components of  $\mathbf{F}$  in the directions of the x and y axis, respectively, and  $\mathbf{i}$  and  $\mathbf{j}$  are unit vectors in these directions. If  $F_x$  and  $F_y$  are known, then F and  $\theta$  can be determined by:

$$F = \sqrt{{F_x}^2 + {F_y}^2}$$
 and  $\tan(\theta) = \frac{F_x}{F_y}$ 

## 6. Element-by-element calculations

The coefficient of friction  $\mu$  can be determined in an experiment by measuring the force F required to move a mass m as shown in Figure 16. When F is measured and m is known, the coefficient of friction can be

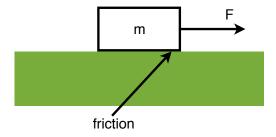


Figure 16: Friction

calculated by:

$$\mu = \frac{F}{mq},$$

where  $g = 9.81 \text{ m/s}^2$ . Results from measuring F in six tests are given in Table 8. Determine the coefficient of friction in each test, and the average from all tests.

Table 8: Friction experiment results

TEST NO.	1	2	3	4	5	6
MASS (KG)	2	4	5	10	20	50
FORCE (N)	12.5	23.5	30	61	117	294

## 7. Solving linear equations

The electrical circuit shown in Figure 17 consists of resistors and voltage sources. Determine the current in each resistor using the mesh current method which is based on Kirchhoff's voltage law.

$$V_1 = 20 \ V$$
  $V_2 = 12 \ V$   $V_3 = 40 \ V$   $R_1 = 18 \ \Omega$   $R_2 = 10 \ \Omega$   $R_3 = 16 \ \Omega$   $R_4 = 6 \ \Omega$   $R_5 = 15 \ \Omega$   $R_6 = 8 \ \Omega$   $R_7 = 12 \ \Omega$   $R_8 = 14 \ \Omega$ 

## Kirchhoff's voltage law

Kirchhoff's voltage law states that the sum of the voltage around a closed circuit is zero. In the mesh current method a current is assigned to each mesh  $(i_1, i_2, i_3, i_4)$ . Then, Kirchhoff's voltage law is applied for each mesh, resulting in a system of linear equations for the currents (four equations in this case). The solution of the equations gives the values of the mesh currents. The current in a resistor that belongs to two meshes is the sum of the currents in the corresponding meshes. It is convenient to assume that all the currents are in the same direction (clockwise in this case). In the equation for each mesh, the voltage source is positive if the current flows to the cathode (negative electrode), and the voltage of the resistor is negative for current in the direction of the mesh current.

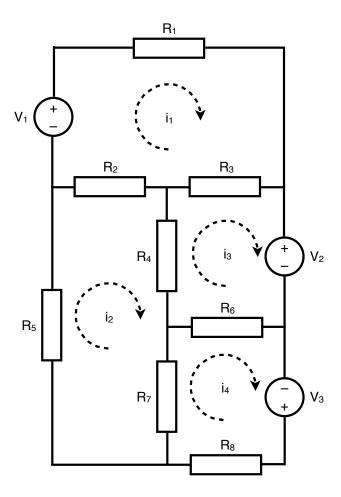


Figure 17: Network of voltage sources and resistors

## C.2 PLOTTING

## 8. 2D plotting

In a typical tension test a dog-bone shaped specimen, as shown in Figure 18, is pulled in a machine. During the test, the force F needed to

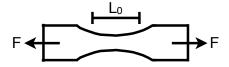


Figure 18: Tension test specimen

pull the specimen and the length L of a gauge section are measured. This data is used for plotting a stress-strain diagram of the material. Two definitions, engineering and true, exist for stress and strain. The engineering stress  $\sigma_e$  and strain  $\epsilon_e$  are defined by:

$$\sigma_e = \frac{F}{A_0}$$
 and  $\epsilon_e = \frac{L - L_0}{L_0}$ ,

where  $L_0$  and  $A_0$  are the initial gauge length and the initial cross-sectional area of the specimen, respectively. The true stress  $\sigma_t$  and strain  $\epsilon_t$  are defined by:

$$\sigma_t = \frac{F}{A_0} \frac{L}{L_0}$$
 and  $\epsilon_t = ln\left(\frac{L}{L_0}\right)$ 

In Table 9 are measurements of force and gauge length from a tension test with an aluminium specimen. The specimen has a round cross section with a radius of 6.4 mm (before the test). The initial gauge length is  $L_0 = 25 \ mm$ . Use the data to calculate and plot the engineering and true stress-strain curves, both on the same plot, of the material.

Table 9: Results of a tension test on an aluminium specimen

FORCE (N)	0	13345	26689	40479	42703	43592	44482	44927
LENGTH (MM)	25	25.037	25.073	25.113	25.122	25.125	25.132	25.144
FORCE (N)	45372	46276	47908	49035	50265	53213	56161	
LENGTH (MM)	25.164	25.208	25.409	25.646	26.084	27.398	29.250	

## 9. 2D plotting

A resistor,  $R=4~\Omega$ , and an inductor, L=1.3~H, are connected in a circuit to a voltage source as shown in Figure 19a. When the voltage

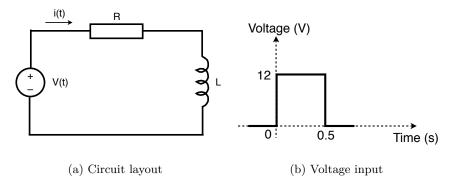


Figure 19: RL circuit

source applies a rectangular pulse with an amplitude of V = 12 V and a duration of 0.5 s, as shown in Figure 19b, the current i(t) in the circuit as a function of time is given by:

$$i(t) = \frac{V}{R} \left( 1 - e^{\frac{-Rt}{L}} \right) \quad \text{for} \quad 0 \le t \le 0.5 \ s$$
$$i(t) = e^{\frac{-Rt}{L}} \frac{V}{R} \left( e^{\frac{0.5R}{L}} - 1 \right) \quad \text{for} \quad 0.5 \le t \ s$$

Make a plot of the current as a function of time for  $0 \le t \le 2$  s.

## 10. 2D plotting

The vibrations of a helicopter due to the periodic force applied by the rotation of the rotor can be modelled by a frictionless spring-mass-damper system subjected to an external periodic force as shown in Figure 20. The position x(t) of the mass is given by the equation:

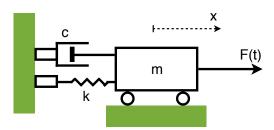


Figure 20: Modelling helicopter rotor vibrations with spring-mass-damper system

$$x(t) = \frac{2f_0}{{\omega_n}^2 - {\omega_2}} \sin\left(\frac{{\omega_n} - \omega}{2}t\right) \sin\left(\frac{{\omega_n} - \omega}{2}t\right),\,$$

where  $F(t) = F_0 \sin(\omega t)$ , and  $f_0 = \frac{F_0}{m}$ ,  $\omega$  is the frequency of the applied force, and  $\omega_n$  is the natural frequency of the helicopter. When the value of  $\omega$  is close to the value of  $\omega_n$  the vibration consists of fast oscillation with slowly changing amplitude called beat. Use  $\frac{F_0}{m} = 12 \ N/kg$ ,  $\omega_n = 10 \ rad/s$ , and  $\omega = 12 \ rad/s$  to plot x(t) as a function of t for  $0 \le t \le 10 \ s$ .

## 11. 2D plotting

The ideal gas equation states that  $\frac{PV}{RT} = n$ , where P is the pressure, V is the volume, T is the temperature,  $R = 0.08206 \; (L \; atm)/(mole \; K)$  is the gas constant, and n is the number of moles. For one mole (n=1) the quantity  $\frac{PV}{RT}$  is a constant equal to 1 at all pressures. Real gases, especially at high pressures, deviate from this behaviour. Their response can be modelled with the van der Waals equation:

$$P = \frac{nRT}{V - nb} - \frac{n^2a}{V^2},$$

where a and b are material constants. Consider 1 mole (n=1) of nitrogen gas at  $T=300~K~(a=1.39~L^2atm/mole^2, {\rm and}~b=0.0391~L/mole)$ . Use the van der Waals equation to calculate P as a function of V for  $0.08 \leq V \leq 6~L$ , using increments of 0.02~L. At each value of V calculate the value of V and make a plot of V versus V. Does the response of nitrogen agree with the ideal gas equation?

## 12. 2D plotting

A simply supported beam that is subjected to a constant distributed load w over two-thirds of its length is shown in Figure 21. The deflection

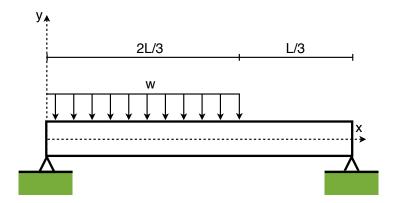


Figure 21: A simply supported beam

y, as a function of x, is given by the equations:

$$y = \frac{-wx}{24LEI} \left( Lx^3 - \frac{16}{9}L^2x^2 + \frac{64}{81}L^4 \right) \quad \text{for} \quad 0 \le x \le \frac{2}{3}L,$$

$$y = \frac{-wL}{54EI} \left( 2x^3 - 6Lx^2 + \frac{40}{9}L^2x - \frac{4}{9}L^3 \right) \quad \text{for} \quad \frac{2}{3}L \le x \le L,$$

where E is the elastic modulus, I is the moment of inertia, and L is the length of the beam. For the beam shown in Figure 21, L=20~m,  $E=200\times10^9~Pa$  (steel),  $I=348\times10^{-6}~m^4$ , and  $w=5\times10^3~N/m$ . Make a plot of the deflection of the beam y as a function of x.

## 13. 3D plotting

An anti-symmetric cross-ply composite laminate has two layers where fibres are aligned perpendicular to one another. A laminate of this type will deform into a saddle shape due to residual thermal stresses as described by the equation:

$$w = k(x^2 - y^2),$$

where x and y are the in-plane coordinates, w is the out-of-plane deflection,, and k is the curvature (a complicated function of material properties and geometry). Make a 3D surface plot showing the deflection of a  $100 \times 100$  mm square plate ( $-50 \le x \le 50$ ,  $-50 \le y \le 50$ ) assuming  $k = 0.254 \ mm^{-1}$ .

## 14. 3D plotting

The van der Waals equation gives a relationship between the pressure P(atm), volume V(L), and temperature T(K) for a real gas:

$$P = \frac{nRT}{V - nb} - \frac{n^2a}{V^2},$$

where n is the number of moles,  $R = 0.08206 \ (L \ atm)/(mole \ K)$  is the gas constant, and  $a(L^2atm/mole^2)$  and b(L/mole) are material constants. Consider 1.5 moles of nitrogen  $(a = 1.39 \ L^2atm/mole^2)$ , and  $b = 0.0391 \ L/mole)$ . Make a 3D surface plot that shows the variation of

pressure (dependent variable, z axis) with volume (independent variable, x axis) and temperature (independent variable, y axis). The domains for volume and temperature are:  $0.3 \le V \le 1.2 L$  and  $273 \le T \le 473 K$ .

## 15. 3D plotting

The normal stress  $\sigma_{xx}$  at point (y, z) in the cross section of a rectangular beam, due to the applied force F at point  $(y_F, z_F)$  is given by:

$$\sigma_{xx} = \frac{F}{A} + \frac{F \ z_F \ z}{I_{yy}} + \frac{F \ y_F \ y}{I_{zz}},$$

where  $I_{zz}$  and  $I_{yy}$  are the area moments of inertia defined by:

$$I_{zz} = \frac{1}{12}bh^3$$
 and  $I_{yy} = \frac{1}{12}hb^3$ 

Determine and make a 3D surface plot of the normal stress in the cross-

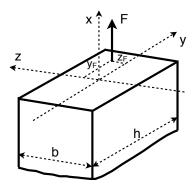


Figure 22: Cross section of a rectangular beam

sectional area shown in Figure 22, given that: h = 40 mm, b = 30 mm,  $y_F = -15 \text{ mm}$ ,  $z_F = -10 \text{ mm}$ , and F = -250000 N. Plot the coordinates y and z in the horizontal plane, and the normal stress in the vertical direction.

## 16. 3D plotting

A defect in a crystal lattice where a row of atoms is missing is called an edge dislocation, as shown in Figure 23. The stress field around an edge dislocation is given by:

$$\sigma_{xx} = \frac{-Gby(3x^2 + y^2)}{2\pi(1 - \nu)(x^2 + y^2)^2},$$

$$\sigma_{yy} = \frac{Gby(x^2 - y^2)}{2\pi(1 - \nu)(x^2 + y^2)^2},$$

$$\tau_{xy} = \frac{Gbx(x^2 - y^2)}{2\pi(1 - \nu)(x^2 + y^2)^2},$$

where G is the shear modulus, b is the Burgers vector, and  $\nu$  is Poisson's ratio. Make 3D surface plots of the stress components (each in a separate

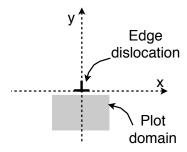


Figure 23: Edge dislocation

figure window) due to an edge dislocation in aluminium for which  $G=27.7\times 10^9~Pa, b=0.286\times 10^{-9}~m,$  and  $\nu=0.334.$  Plot the stresses in the domain  $-5\times 10^{-9}\leq x\leq 5\times 10^{-9}~m$  and  $-5\times 10^{-9}\leq y\leq -1\times 10^{-9}~m.$  Plot the coordinates x and y in the horizontal plane, and the stresses in the vertical direction.

## 17. 3D plotting

Molecules of a gas in a container are moving around at different speeds. Maxwell's speed distribution law gives the probability distribution P(v) as a function of temperature and speed:

$$P(\nu) = 4\pi \left(\frac{M}{2\pi RT}\right)^{\frac{3}{2}} v^2 e^{\frac{-Mv^2}{2RT}},$$

where M is the molar mass of the gas in kg/mol, R=8.31 J/mol K is the gas constant, T is the temperature in K, and v is the molecules speed in m/s. Make a 3D surface plot of P(v) as a function of v and T for  $0 \le v \le 1000$  m/s and  $70 \le T \le 320$  K for oxygen (molar mass M=0.032 kg/mol).

## 18. 3D plotting

An RLC circuit with an alternating voltage source is shown in Figure 24. The source voltage  $v_s$  is given by  $v_s = v_m \sin(\omega_d t)$  where  $\omega_d = 2\pi f_d$  is which  $f_d$  is the driving frequency. The amplitude of the current I in the circuit is given by:

$$I = \frac{v_m}{\sqrt{R^2 + \left(\omega_d L - \frac{1}{\omega_d C}\right)^2}},$$

where R and C are the resistance of the resistor and the capacitance of the capacitor, respectively. For the circuit in Figure 24  $C=15\times 10^{-6}~F$ ,  $L=240\times 10^{-3}~H$ , and  $v_m=24~V$ .

a) Make a 3D surface plot of the current I (z axis) as a function of  $\omega_d$  (x axis) for  $60 \le f \le 110~Hz$ , and a function of R (y axis) for  $10 \le R \le 40~\Omega$ .

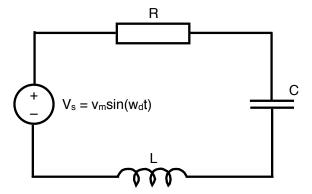


Figure 24: An RLC circuit with an alternating voltage source

b) Rotate the plot into the x-z plane. Estimate the natural frequency of the circuit (the frequency at which I is maximum). Compare the estimate with the calculated value of  $1/(2\pi\sqrt{LC})$ .

#### C.3 SCRIPTS AND FUNCTIONS

## 19. Scripts

A cylindrical silo with radius r has a spherical cap roof with radius R, as shown in Figure 25. The height of the cylindrical portion is H. Write a script file that determines the height H for given values or r, R, and the volume V. In addition the script should also calculate the surface area of the silo. The volume of the cylinder is given by:

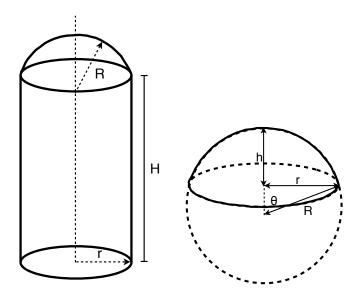


Figure 25: Silo

$$V_{\rm cyl} = \pi r^2 H,$$

and the volume of the spherical cap is given by:

$$V_{\rm cap} = \frac{1}{3}\pi h^2 (3R - h),$$

where  $h = R - R\cos(\theta)$  and  $\theta$  is calculated from  $\sin(\theta) = \frac{r}{R}$ . The height H of the cylindrical part can be expressed by:

$$H = \frac{V - V_{\text{cap}}}{\pi r^2}$$

The surface area of the silo is obtained by adding the surface areas of the cylindrical part and the spherical cap:

$$S = S_{\text{cyl}} + S_{\text{cap}} = 2\pi r H + 2\pi R h$$

Calculate the height and surface area of a silo with  $r=10\ m,\,R=15\ m$  and  $V=3500\ m^3.$ 

## 20. Scripts

Radioactive decay of radioactive materials can be modelled by the equation  $A = A_0 e^{kt}$ , where A is the population at time t,  $A_0$  is the amount at

t=0, and k is the decay constant  $(k \le 0)$ . Technetium-99 is a radioisotope that is used in the imaging of the brain. Its half-life time is 6 hours. Write a script to calculate the relative amount of Technetium-99  $(A/A_0)$  in a patient body for 24 hours after receiving a dose. After determining the value of k, define a vector t=0:2:24 and then calculate and plot the corresponding values of  $A/A_0$ .

## 21. Scripts

The variation of vapour pressure P (in units of mm Hg) of benzene with temperatures (in  ${}^{\circ}C$ ) in the range  $8{}^{\circ}C \leq T \leq 80{}^{\circ}C$  can be modelled with the Antoine equation:

$$log_{10}P = A - \frac{B}{C + T}$$

For benzene, the values of the constants are as follows: A=6.87987, B=1196.76, C=219.161. Write a script that calculates the vapour pressure for various temperatures. The script should create a vector of temperatures from  $T=8^{\circ}C$  to  $T=42^{\circ}C$  in increments of 2 degrees, and display a two-column table of P and T where the first column is temperatures in  ${}^{\circ}C$ , and the second column is the corresponding pressures in mm Hg. The script should also plot P against T and use a logarithmic axis for P.

## 22. Scripts

The temperature dependance of the heat capacity  $C_p$  of many gases can be described in terms of a cubic equation:

$$C_p = a + bT + cT^2 + dT^3$$

The following table gives the coefficients of the cubic equation for four gases.  $C_p$  is in  $Joules/(g \ mol)(^{\circ}C)$  and T is in  $^{\circ}C$ . Write a script to cal-

Table 10: Coefficients for the cubic equation for the heat capacity of gases

GAS	a	b	c	d
$SO_2$	38.91	$3.904 \times 10^{-2}$	$-3.205 \times 10^{-5}$	$8.606 \times 10^{-9}$
$SO_3$	48.50	$9.188 \times 10^{-2}$	$-8.540 \times 10^{-5}$	$32.40 \times 10^{-9}$
$O_2$	29.10	$1.158\times10^{-2}$	$-0.6076 \times 10^{-5}$	$1.311\times10^{-9}$
$N_2$	29.00	$0.220\times10^{-2}$	$-0.5723 \times 10^{-5}$	$-2.871 \times 10^{-9}$

culate the heat capacity for each gas at temperatures ranging between 200 and  $400^{\circ}C$  at  $20^{\circ}C$  increments. To present the results, create an  $11\times5$  matrix where the first column is the temperature, and second to fifth columns are the heat capacities of  $SO_2$ ,  $SO_3$ ,  $O_2$ , and  $N_2$ , respectively.

Constants for the Antoine equation taken from the Dortmund Data Base.

## 23. Functions

Create a function file that calculates the trajectory of a projectile. The inputs to the function should be the initial velocity and the angle at which the projectile is fired. The outputs from the function should be the maximum height and distance. In addition, the function should generate a plot of the trajectory. Use the function to calculate the trajectory of a projectile that is fired at a velocity of 230 m/s at an angle of 39 °.

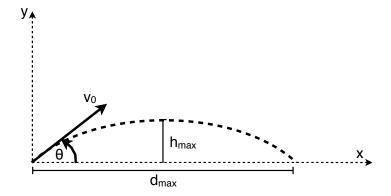


Figure 26: Motion of a projectile

# (?) The motion of a projectile

The motion of a projectile can be analysed by considering the horizontal and vertical components. The initial velocity  $v_0$  can be resolved into horizontal and vertical components:

$$v_{0x} = v_0 \cos(\theta)$$
 and  $v_{0y} = v_0 \sin(\theta)$ 

In the vertical direction the velocity and position of the projectile are given by:

$$v_y = v_{0y} - gt$$
 and  $y = v_{0y}t - \frac{1}{2}gt^2$ 

The time it takes the projectile to reach the highest point  $(v_y = 0)$  and the corresponding height are given by:

$$t_{h \max} = \frac{v_{0y}}{g}$$
 and  $h_{\max} = \frac{v_{0y}^2}{2g}$ 

The total flying time is twice the time it takes the projectile to reach the highest point,  $t_{\text{tot}} = 2t_{h_{\text{max}}}$ . In the horizontal direction the velocity is constant, and the position of the projectile is given by:

$$x = v_{0x}t$$

#### 24. Functions

Write a user-defined function, with two input and output arguments, that determines the height in metres and mass kilograms of person from their height in inches and mass in pounds. For the function name and arguments use [m,kg] = STtoSI(in,1b). Use the function in the Command Window to determine in SI units the height and mass of a 5 ft. 11 in. person who weighs 181 lb.

## 25. Functions

When n resistors are connected in parallel, their equivalent resistance  $R_{eq}$  can be determined from:

$$\frac{1}{R_{eq}} = \frac{1}{R_1} + \frac{1}{R_2} + \ldots + \frac{1}{R_n}$$

Write a user-defined function that calculates  $R_{eq}$ . For the function name and arguments use REQ = req(R). The input to function should be a vector in which each element is a resistor value, and the output from the function is  $R_{eq}$ . Use the function to calculate the equivalent resistance when the following resistors are connected in parallel: 50  $\Omega$ , 75  $\Omega$ , 300  $\Omega$ , 60  $\Omega$ , 500  $\Omega$ , 180  $\Omega$ , and 200  $\Omega$ .

#### 26. Functions

A 2D state of stress at a point in a loaded material is defined by three components of stresses  $\sigma_{xx}$ ,  $\sigma_{yy}$ , and  $\tau_{xy}$ . The maximum and minimum

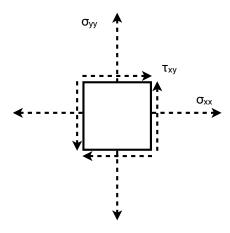


Figure 27: A 2D state of stress at a point in a loaded material

normal stresses (principal stresses) at the point,  $\sigma_{\text{max}}$  and  $\sigma_{\text{min}}$ , are calculated from the stress components by:

$$\sigma_{\text{min}}^{\text{max}} = \frac{\sigma_{xx} + \sigma_{yy}}{2} \pm \sqrt{\left(\frac{\sigma_{xx} - \sigma_{yy}}{2}\right)^2 + \tau_{xy}^2}$$

Write a user-defined function that determines the principal stresses from the stress components. For the function name and arguments use [Smax ,Smin] = princstress(Sxx,Syy,Sxy). Use the function to determine the principal stresses for the following states of stress:  $\sigma_{xx} = -190~MPa$ ,  $\sigma_{yy} = 145~MPa$ ,  $\tau_{xy} = 110~MPa$ .

## 27. Functions

In a low-pass RC filter (a filter that passes signals with low frequencies), the ratio of the magnitude of the voltages is given by:

$$RV = \left| \frac{V_o}{V_i} \right| = \frac{1}{\sqrt{1 + (\omega RC)^2}},$$

where  $\omega$  is the frequency of the input signal. Write a user-defined func-

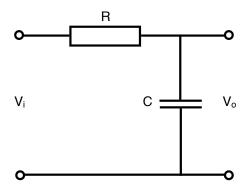


Figure 28: A low-pass filter

tion that calculates the magnitude ratio. For the function name and arguments use RV = lowpass(R,C,w). The input arguments are: R the size of the resistor in  $\Omega$  (ohms), C the size of the capacitor in F (Farads), and w the frequency of the input signal in rad/s. Write the function such that w can be a vector.

Write a script file that uses your lowpass function to generate a plot of RV as a function of  $\omega$  for  $10^{-2} \le \omega \le 10^6 \ rad/s$ . The plot should have a logarithmic scale on the x-axis  $(\omega)$ . When the script file is executed it should prompt the user to enter values of R and C. Run the script file with  $R=1200\ \Omega$  and  $C=8\ \mu F$ .

#### C.4 DECISION MAKING

## 28. Relational & Logical operators

The following were the daily maximum temperatures ( ${}^{\circ}C$ ) in Washington DC during the month of April 2002: 14, 23, 23, 12, 10, 9, 13, 23, 23, 19, 21, 17, 23, 28, 29, 33, 34, 32, 33, 27, 15, 21, 13, 18, 17, 19, 18, 23, 17, 21. Write a script, and use relational and logical operators to determine the following:

- a) The number of days the temperature was above 24  $^{\circ}C$ .
- b) The number of days the temperature was between 18  $^{\circ}C$  and 27  $^{\circ}C$ .

### 29. while loops

The flight of a model rocket of mass 0.05 kg can be modelled as follows. During the first 0.15 s the rocket is propelled up by the rocket engine with a force of 16 N. The rocket then flies up slowing down under the force of gravity. After it reaches the apex, the rocket starts to fall back down. When its down velocity reaches 20 m/s a parachute opens (assumed to open instantly) and the rocket continues to move down at a constant speed of 20 m/s until it hits the ground. Write a script that calculates and plots the speed and altitude of the rocket as a function of time during the flight.

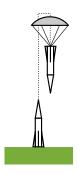


Figure 29: Flight of a model rocket

## Flight of a model rocket

The rocket is assumed to be a particle that moves along a straight line in the vertical plane. For motion with constant acceleration along a straight line, the velocity and position as a function of time are given by:

$$v(t) = v_0 + at$$
 and  $s(t) = s_0 + v_0 t + \frac{1}{2}at$ ,

where  $v_0$  and  $s_0$  are the initial velocity and position, respectively. The flight of the rocket can be divided into three segments and you should calculate each segment using a separate while loop in your script.

Segment 1 - The first 0.15 s when the rocket engine is on

During this period, the rocket moves up with a constant acceleration. The acceleration is determined by drawing a free body and a mass acceleration diagram. From Newton's second law, summing the forces in the vertical direction gives an equation for the acceleration:

$$a = \frac{F_E - mg}{m}$$

The velocity and height as a function of time are:

$$v(t) = 0 + at$$
 and  $h(t) = 0 + 0 + \frac{1}{2}at^2$ ,

where the initial velocity and initial position are both zero. In your script this while loop starts at t=0 and continues looping as long as  $t \leq 0.15$  s. The time, velocity and height at the end of this segment are  $t_1, v_1, \text{ and } h_1.$ 

Segment 2 - The motion from when the engines stops until the parachute

In this segment the rocket moves with a constant deceleration q. The speed and height of the rocket as a function of time are given by:

$$v(t) = v_1 - g(t - t_1)$$
 and  $h(t) = h_1 + v_1(t - t_1) - \frac{1}{2}g(t - t_1)^2$ 

In your script this while loop should continue looping until the velocity of the rocket is -20 m/s (negative since the rocket is falling). The time and height at the end of this segment are  $t_2$  and  $h_2$ .

(?) Flight of a model rocket (continued)

Segment 3 - The motion from when the parachute opens until the rocket hits the ground

In this segment the rocket moves with constant velocity (zero acceleration). The height as a function of time is given by:

$$h(t) = h_2 + v_{\text{chute}}(t - t_2),$$

where  $v_{\rm chute}$  is the constant velocity after the parachute opens. In your script this while loop should continue looping as long as the height is greater than zero.

## BIBLIOGRAPHY

- [1] E.B. Magrab, S. Azarm, B. Balachandran, J.H. Duncan, K.E. Herold, and G.C. Walsh. *An Engineer's guide to MATLAB: With applications from mechanical, aerospace, electrical, and civil engi.* 2nd. Pearson Prentice Hall, 2005.
- [2] H. Moore. MATLAB for Engineers. Prentice Hall, 2009.
- [3] J.F. Patzer. Introduction to Engineering Computing. http://www.pitt.edu/~patzer/index.htm [Last accessed: July 2010]. 2003. URL: http://www.pitt.edu/~patzer/index.htm.
- [4] R. Pratap. Getting Started with MATLAB 7 A Quick Introduction for Scientists and Engineers. Oxford University Press, 2006.