MATLAB: First Steps

iT Centre Learning

iT services

UNIVERSITY OF OXFORD

The small print

Prerequisites

Time in the workshop is precious – it is an opportunity for you to interact with the workshop leader and other participants through questions and discussions and to share your experiences and concerns. To make the most of this time we sometimes ask you to carry out learning activities ahead of the workshop so that everyone comes into the class with the same basic knowledge. We keep this prior learning to a minimum and often make use of online videos. Online videos provided through LinkedIn Learning can be accessed free of charge by University members anytime, anywhere, through a browser or app.

Your course booking will tell you if any prior learning activity is required. If you don't have an environment where you can do this learning, you can come along to one of our LinkedIn Learning sessions. These are a quiet space where you can work through videos or other workshop resources.

Copyright

About the workshop designer

Isaac's roots are originally in mathematics, then he branched out into engineering for his PhD where he used MATLAB extensively for mathematically modelling fluid flow. He left the academic world briefly to focus on science communication where he enjoyed a combination of running coding clubs and training teachers on the Computer Science curriculum.

He has been using MATLAB for 10 years and teaching MATLAB at the University of Oxford for 6 years. He has a passion for teaching, completing a PGCert in Higher Education with a focus on teaching programming to undergraduates. He is a Fellow of the Higher Education Academy and a certified Software Carpentries instructor.

For MATLAB quick links and Isaac's other interests, see linktr.ee/isaacmear

Revision history

Version	Date	Author	Comments
1.0	September 2022	Isaac Mear	Created

About this workshop

This workshop helps you to get started using MATLAB for research.

What you will learn

We will introduce some basic computing concepts, to give you the confidence to use MATLAB to complete a range of scientific tasks. This is an interactive workshop, so you will be trying out hands-on activities with the tutor on hand for guidance. We work through the short exercises so you can start learning how to use MATLAB for basic tasks.

We will include pointers to other workshops and further resources that will help you keep improving your MATLAB skills. If you have a specific idea of what you want to use MATLAB to achieve as part of your research, then there will also be the opportunity to discuss this. The tutor will then give specific advice for the next steps in your MATLAB learning journey.

What you need to know

We will assume that you are reasonably confident using a computer for daily activities.
You will need already to be able to:

- Understand what a file is on the computer (e.g. a Word document, a .jpg image)
- Understand how a file system words (with files stored in folders)
- Be able to move a file from one folder to another

To make this course as accessible as possible, we are assuming no prior knowledge of computer programming. This means that if you are experienced at programming, you may find parts of the course a little slow. In this case, if you are waiting for others to finish working through a section we recommend you keep yourself entertained with some coding challenges in MATLAB via mathworks.com/matlabcentral/cody/.

The resources you need

The teaching space will contain computers with MATLAB already installed, so you do **not** need to bring a laptop.

However, if you would like to work on your own device (either with MATLAB already installed, or to get help installing MATLAB), then please do feel free to bring this along. MATLAB can be used fully on Windows, Mac and Linux. There is a MATLAB App available for iOS devices.

The resources for most workshops, including any pre-course activity, are in the IT Learning Portfolio: visit skills.it.ox.ac.uk/it-learning-portfolio and search for “MATLAB”.

Learning Objectives

This workshop has the following learning objectives:

Learning Objective One : Using basic programming constructs

Learning Objective Two : Create and Run MATLAB Scripts

Learning Objective Three : Importing and Plotting Data

Learning Objective Four : Understand tasks MATLAB can help with

Learning Objective Five : Know how to install MATLAB and access further resources

Learning Objective One : Using basic programming constructs

In this section you will try using the MATLAB Command Window as a basic calculator, as well as defining some variables.

☐ Practice **naming variables**. You can experiment your self or use the following:

- Two questions from MATLAB OnRamp Chapter 2 (Lesson 2, Section 1)
- Feel free to move onto the next part, but if you feel you still need more practice with naming variables, follow along with this Software Carpentries page:

  http://swcarpentry.github.io/matlab-novice-inflammation/01-intro/index.html

☐ Practice using variabes on some **built-in functions**, such as those listed in the **MATLAB Quick Reference** under **Built in Functions**.

☐ Practice **creating arrays** and matrices:

- Take a look at the **MATLAB Quick Reference** and try **Generating Vectors**
- Questions from Chapter 4 "Vectors and Matrices" on MATLAB Onramp

☐ Practice **accessing and modifying arrays**:

- Take a look at the **MATLAB Quick Reference** and try the examples in the **Indexing section** on matrices `A = rand(1,15)` for Vector Indexing and `M = magic(15,15)`
- Take a look at the questions from Chapter 5 of MATLAB Onramp (Lession 1, Section 2)
- Follow along with this Software Carpentries page:

  http://swcarpentry.github.io/matlab-novice-inflammation/02-arrays/index.html

☐ Practice completing tasks on **whole arrays**

- Questions from Chapter 6 "Array Calculations" on MATLAB Onramp (Lesson 1, Section 1)

Click or tap here to enter text.

Learning Objective Two : Create and Run MATLAB Scripts

Scripts group lots of lines of code together. They allow you to repeat tasks quickly and easily.

☐ Work through the **Example: Introduction to Scripts using Shapes**
(see this document see after page 12, or the PDF in the Examples Folder.)

☐ To see a Live Script, complete the MATLAB Onramp section on running scripts:

https://matlabacademy.mathworks.com/R2022a/portal.html?course=gettingstarted#chapter=3&lesson=3&section=1
Task 2 on debugging is useful too.

☐ Take a look at the Live Script Gallery, and try running some examples:

https://uk.mathworks.com/products/matlab/live-script-gallery.html

Download your favourite, and open it in MATLAB on the local computer.
Edit it and check it still runs.

☐ If you need more practice understanding the MATLAB search path,
see MATLAB Fundamentals Chapter 14 (Lesson 4, Section 1)

☐ Functions are similar to scripts, but far more powerful. You can learn about them here:

http://swcarpentry.github.io/matlab-novice-inflammation/07-func/index.html

or by completing MATLAB Fundamentals Chapter 15:
matlabacademy.mathworks.com/R2021b/portal.html?course=mlbe#chapter=15&lesson=1&section=1

Click or tap here to enter text.

Learning Objective Three : Importing and Plotting Data

MATLAB has many plotting capabilities in both 2D and 3D. There are a range of plotting functions.

☐ Try out the Examples in the MATLAB Plot Documentation:

https://uk.mathworks.com/help/matlab/ref/plot.html

☐ Take a look at the page on Plotting in the **MATLAB Quick Reference**,

the summary of line colours and symbols may be useful when you are working.

☐ Work through the Exercises on Plotting from the Software Carpentries course:

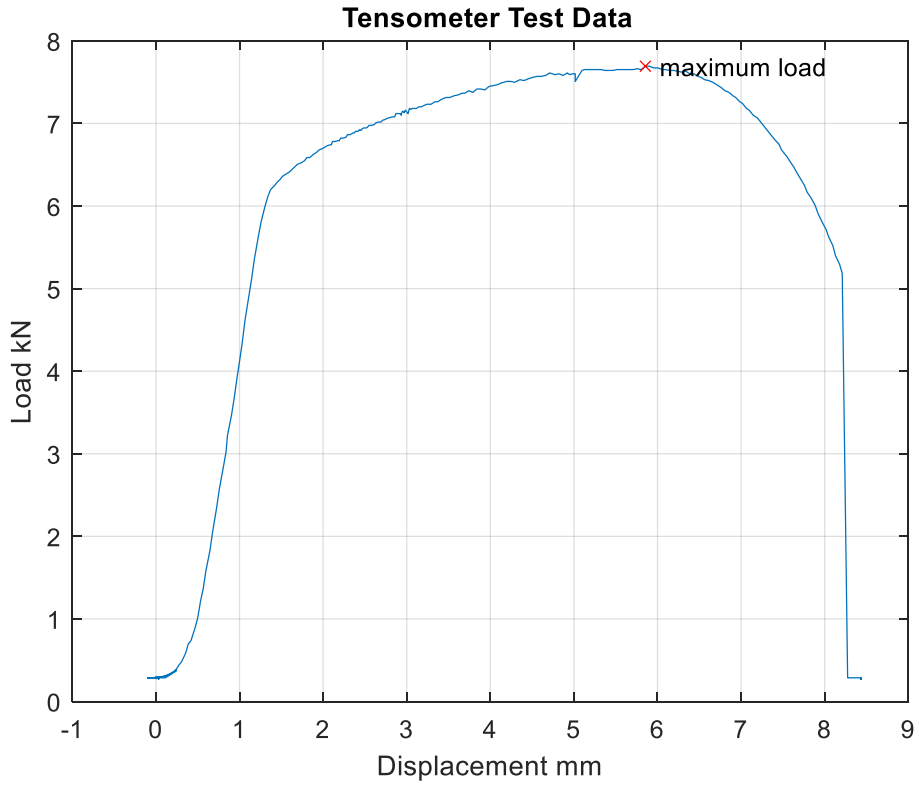
http://swcarpentry.github.io/matlab-novice-inflammation/03-plotting/index.html

You will need to load the data first, using the command:

```
patient_data = readmatrix('CarpentriesInflammationData/inflammation-01.csv')
```

☐ Look at the Example of using the Import Tool:
https://uk.mathworks.com/help/matlab/import_export/import-data-interactively.html

Use this to try and import the example TestData given. Can you recreate the figure below? The first column is the displacement, and the second is the load in kN.



Learning Objective Four : Understand tasks MATLAB can help with

4.1 Reading Reading documentation to find examples/get help

MATLAB has extensive documentation: that is information on its different functions complete with multiple examples. There are so many different things MATLAB can do that no one can memorise them all! That is why you need to practice reading the MATLAB documentation, which is full of MATLAB Exampes. Even when you've been using MATLAB for years, you'll still use the examples.

☐ Use the **help** function and **doc** function to read about the following functions:

`mean, append, rand, contains`

☐ Read the **documentation page** on different kinds of plots:

https://uk.mathworks.com/help/matlab/creating_plots/types-of-matlab-plots.html

Find a plot that might be useful in your own work, and open the page for that plot. Run one of the examples in the MATLAB Command Window.

☐ Try out the **Example Research Task: Identify a Maximum** after page 12.

☐ Read and run some of the **examples** from the **image processing toolbox:**

https://uk.mathworks.com/help/images/examples.html?s_tid=CRUX_topnav

4.2 Speeding up research using computational techniques (loops + checks)

Most research tasks can be sped up by using MATLAB to repeat proceses automatically. In order to make use of these, you need to understand some basic programming techniques. These are for loops, while loops and if statements. There is not enough time to cover each of these in depth, but it is important you start to practice using these so you can implement them in your own work flow.

☐ Use **MATLAB Quick Reference** to try some examples of if statements

☐ Complete the Software Carpentary Exercises on If statements:

http://swcarpentry.github.io/matlab-novice-inflammation/06-cond/index.html

☐ Use **MATLAB Quick Reference** to try some examples of for loops

☐ Complete the Software Carpentary Exercises on for loops statements:

http://swcarpentry.github.io/matlab-novice-inflammation/05-loops/index.html

☐ If you need to automatically move/save files or folders in your research, you may want to learn about the Command Line:

https://swcarpentry.github.io/shell-novice/

Learning Objective Five : Know how to install MATLAB and access further resources

MATLAB and all its associated toolboxes and products can be accessed free of charge by University members anytime, anywhere through:

- a full installation on any computer
- through a browser using MATLAB Online
- via the MATLAB App on Andriod or iOS

If you do not already have one, **create a MathWorks account** and validate it through the University's Single Sign-On portal. Ensure you sign up with a valid university e-mail address. You can do this via bit.ly/matlabportal

If you brought your own device, **install MATLAB** using the appropriate license following the details on www.eng.ox.ac.uk/matlab

Access MATLAB Online (matlab.mathworks.com).
In the Command Window, use the following command to explore the example on plotting and Live Scripts: `openExample('matlab/LiveEditorIntroduction')`

Visit the MATLAB Academy
Read through the different courses to find ones useful for you. Chapter headings give good insight into course topics.

Discuss what you want to use MATLAB for with the tutor
Based on their recommendations, or what you can find reading the MATLAB documentation, fill in your personal learning action plan on the next page.

Try out some of the **Cody Challenges** (mathworks.com/matlabcentral/cody/)

Learning Action Plan

Three hours is not enough to become a MATLAB expert, but we hope it will give you the confidence to move forwards with self-study. Below is a list of common functionality used for research. Tick those you feel apply to your situation, or write others down. Discuss this with the course leader if you want specific next steps, particularly if you have an idea of what you will use MATLAB for.

MATLAB Academy Courses to complete:

☐ MATLAB On Ramp ☐ MATLAB Fundamentals ☐ Data Processing+Visualization

☐ Intro to Statistical Methods ☐ Image Processing Onramp ☐ Programming Techniques

Other courses:

Read about relevant toolboxes documentation, and explore introductory examples:

☐ Image Processing ☐ Curve Fitting ☐ Data Acquisition

Other relevant toolboxes:

Programming Concepts to Learn more about:

☐ If Statements – Check if a condition is true or false (MATLAB Fundamentals Chapter 14 Lesson 3)

☐ For Loops: Repeat tasks a known amount of times (MATLAB Fundamentals Chapter 14 Lesson 4)

☐ While Loops: Repeat tasks unknown amount of times (MATLAB Fundamentals Chap 14 Lesson 5)

Where you might need to use these:

Learn about new datatypes:

☐ Cell Arrays ☐ Tables ☐ Strings

What you may use them for:

LinkedIn Learning:

☐ MATLAB for Data Calculation:
https://www.linkedin.com/learning/matlab-2018-essential-training/use-matlab-for-data-calculation

You may want to schedule some time in your calendar now, for when you are going to work on the next steps.

You may want to swap contact details with someone else on the course, so you have have an accountability buddy. You can check in on each other's progress with MATLAB.
You may also want to join the MATLAB User Group for Oxford: bit.ly/matlabusergroup

MATLAB Resources

- **MATLAB Portal** bit.ly/matlabportal

  This is the University of Oxford page on the MathWorks official website that allows you to link your MathWorks account with the University License.

- **MATLAB @ Oxford Webpage** www.eng.ox.ac.uk/matlab

  Website for all members of the University of Oxford to get information on MATLAB installation. It also lists your Local MATLAB Representatives, who can help with installations.

- **Join the University of Oxford MATLAB User Group** bit.ly/matlabusergroup

- **MATLAB Academy** matlabacademy.mathworks.com

  Free online training on a range of MATLAB topics. For example:
  MATLAB Fundamentals (16 hours), MATLAB for Data Processing and Visualisation (8 hours) or MATLAB Programming Techniques (16 hours)

- **LinkedIn Learning: MATLAB Essentials (3 hours)**
  www.linkedin.com/learning/matlab-2018-essential-training/use-matlab-for-data-calculation
  A good follow-on course which covers conditional logic and loops, useful datatypes strings and structures as well as recapping scripts and plotting

- **Cody Challenges** mathworks.com/matlabcentral/cody/

  Small coding challenges are a great way to improve your computational thinking. Problems are listed in 'Groups': read a challenge, use MATLAB to solve the problem and paste your code into the Cody website for automatic marking.

- **App Designer** www.mathworks.com/videos/app-designer-overview-1510748719083.html

  You can create complex user interfaces in MATLAB, that can speed up workflow.
  This 15 minutes video allows you to get started with creating apps in MATLAB.

- **Software Carpentries** https://software-carpentry.org/lessons
  Training on a wide variety of scientific computing techniques: from Unix, GitHub, R to MATLAB. Training is available in-person (for a cost) or all materials are for free online.

Further information

Getting extra help

The IT Learning Centre offers bookable clinics where you can get pre- or post-course advice. Contact us using courses@it.ox.ac.uk.

Study Videos from LinkedIn Learning

Our website contains a collection of self-service courses and resources. LinkedIn Learning video-based courses are free to all members of the University. Visit skills.it.ox.ac.uk/linkedin-learning and sign in with your Single Sign-On (SSO) credentials. You can watch the videos anywhere, anytime, and even download them onto a tablet/smartphone for offline viewing.

About the IT Learning Portfolio online

Many of the resources used in the IT Learning Centre courses and workshops are made available as Open Educational Resources (OER) via our Portfolio website at skills.it.ox.ac.uk/it-learning-portfolio.

Find resources for this course by visiting the IT Learning Portfolio and searching for “MATLAB”.

About the IT Learning Centre

The IT Learning Centre delivers over 100 IT-related teacher-led courses, which are provided in our teaching rooms and online, and we give you access to thousands of online self-service courses through LinkedIn Learning.

Our team of teachers have backgrounds in academia, research, business and education and are supported by other experts from around the University and beyond.

Our courses are open to all members of the University at a small charge. Where resources allow, we can deliver private courses to departments and colleges, which can be more cost-effective than signing up individually. We can also customize courses to suit your needs.

Our fully equipped suite of seven teaching and training rooms are usually available for hire for your own events and courses. For more information, contact us at courses@it.ox.ac.uk.

About IT Customer Services

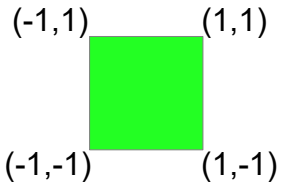
The IT Learning Centre is part of the Customer Services Group. The group provides the main user support services, assisting all staff and students within the University as well as other users of University IT services. It supports all the services offered by IT Services plus general IT support queries from any user, working in collaboration with local IT support units.

The Customer Services Group also offers a data backup service; an online shop; and a computer maintenance scheme. Customer Services is further responsible for desktop computing services – for staff and in public/shared areas – throughout UAS and the Bodleian Libraries.

Example: Introduction to Scripts using Shapes

This exercise gives you an idea of what it is like using MATLAB scripts.
You are provided with functions to draw and move shapes.

The shape we are going to produce is a simple square as shown on the right. This is represented by two rows of information:



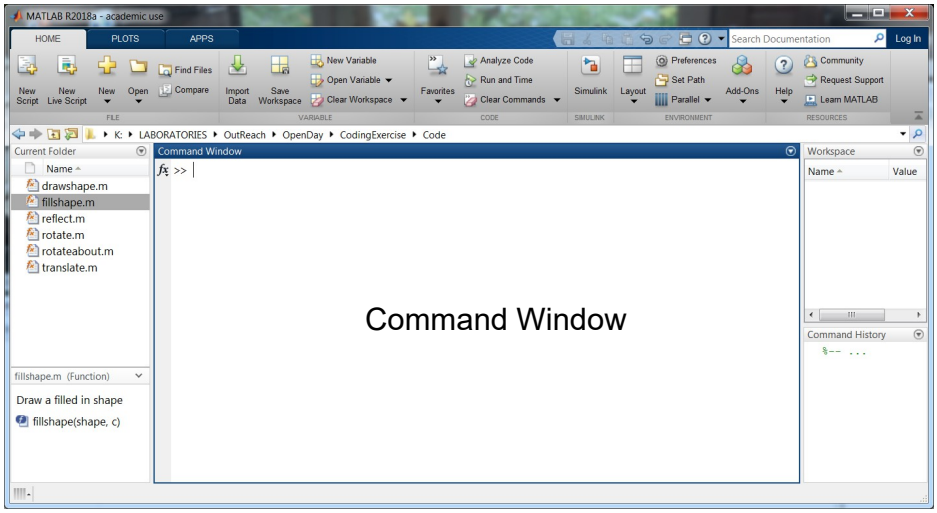
$$\begin{bmatrix} -1 & 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

- The *x* coordinates are on the top row.
- The *y* coordinates are on the bottom row.

The first column is the coordinates of the lower left hand corner of the square, (-1,1).
To draw the square, we go around in an anticlockwise direction. So the next coordinate is the lower right hand corner (1,-1) etc. To complete the shape, the last column returns to the lower left corner again. If we do not repeat this co-ordinate the shape will not close.

Opening the Editor in MATLAB

The large white window in the centre of MATLAB is the **Command Window**.



In the **Command Window** type: **edit MyProgram.m** then press Enter.

An **Editor** window opens so you can start writing your code.

Writing a Program to Plot a Green Square

In the Editor, enter the following code to define co-ordinates of a shape:

```
sqr = [-1  1  1 -1 -1
       -1 -1  1  1 -1];
```

To draw the shape, we use a function called **fillshape**. Insert the following line into your program after the above, to use the function to draw the square:

```
fillshape(sqr,'g')
```

The letter 'g' is used to indicate that we want the colour to be green.

To save and run the program, click on the green arrow icon in the top ribbon.

You will see the square fills the whole graph. We want the ***x*** and ***y*** coordinates of the graph to go from **-6** to **+6**. We also want the scale of the ***x*** and ***y*** axes to be the same (equal).

Close the figure window and insert the following at the bottom of the program:

```
axis('equal',[ -6 6 -6  6 ]);
```

Run the program again. The green square should be smaller in the middle of the graph.

Moving a Shape

The function **translate(a,b)** moves a shape to the left by ***a*** and up by ***b***.

After drawing the square in green, but before the axis command, add the following code:

```
shape1 = translate(sqr,3,4);
fillshape(shape1,'r')
```

Run the program again and you should see a red square that is 3 to the right and 4 up.

Below where shape1 is drawn, insert the following code:

```
shape2 = rotate(sqr,pi/3);
fillshape(shape2,'b')
```

Run the program again and you should see the square has been rotated by $\pi/3$ radians and drawn in blue.

Keeping all Shapes at Once (Hold On)

We can change the program so that we can see all three squares at the same time.

At the very top of the program, add the command to hold the shapes on the screen.

```
hold on
```

Then run the program again.

Using Loops to Draw Many Shapes

For something a bit more interesting, we will write code to draw 200 squares in a circle. First we need the coordinates of each of the squares. We can use trigonometry to generate these co-ordinates. At the very end of the program, add the following:

```
% Find 200 points between 0 and pi
n = 200;
T = linspace(0,2*pi,n);

% Use those points and some trigonometry,
% to find the co-ordinates around edge of a circle
X = 4*cos(T);
Y = 4*sin(T);

plot(X,Y,'r.')
```

What does this code do?

- The function **linspace** generates 200 numbers, evenly spaced between zero and 2π. To see the numbers, in the **Command Window**, type in **T** and press **Enter**.
- The variables **X** and **Y** are the coordinates of 200 points on a circle.
- The plot function, plots each of these points as a red dot.

  At each of the points on the circle, we are going to plot a square. Run the program to see this.

In the Command Window, enter **X(3)**. This is the third x-coordinate.

Likewise, **Y(3)** is the third y-coordinate. We are going to use this mechanism, to get each of the coordinates in turn, translate the square to the coordinates, then draw the shape in red.

Add the following to the bottom of the program:

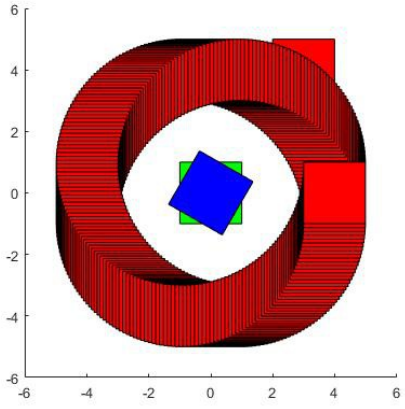
```
% For each co-ordinate found, draw a square at that point
for k = 1:n
    % find current co-ordinate
    x=X(k);
    y=Y(k);

    % Move the square to that location
    shape = translate(sqr,x,y);

    % Draw the square and sort out the axis
    fillshape(shape,'r')
    axis('equal',[ -6 6 -6  6 ]);
end
```

This is called a **for loop**. The top line says,

for each number from one to **n**,

set **k** equal to that number

then execute every line between the **for** and the **end**



What happens when the code is run?

So **k** is first set to one, and the first coordinates are obtained. The square is translated to those coordinates and drawn in red. Then **k** is set to two and the second square is drawn. The code within the loop is executed 200 times. Every time we go around the loop, the variable **k** is increase by one and we draw a square at a new location. Run the program.

Creating an Animation

Just before the **end** of the for loop, add the following.

```
pause(0.02);
```

There is now a 0.02 second pause after each square is drawn to slow things down.

At the top of the program, put a percent sign in front of hold.

```
%hold on
```

This stops the **hold on** command from running. Run the program again.

Rotating Many Shapes

Now we are going to rotate the square before it is translated.

Change the **for loop** so that it looks like this:

```
for k = 1:n
    x=X(k);
    y=Y(k);
    t=T(k);          %The current angle
    shape1 = rotate(sqr,t);
    shape2 = translate(shape1,x,y);
    fillshape(shape2,'r')
    axis('equal',[ -6 6 -6  6 ]);
    pause(0.02);
end
```

Run the program. The square should now also rotate as it moves in a circle.

Increase **t** by a factor of 4 to increase the rotation rate.

```
t = 4*T(k);
```

Then see what happened when you remove the percent sign from the front of hold on.

It looks better if we clear the graph first. Add the following just above the for loop.

```
cla
```

Example Research Task: Identify a Maximum

In this exercise, you will process data from a tensometer, a device for testing the strength of a material. The material specimen is slowly stretched and the applied force and the displacement (the amount the specimen has been stretched) are both recorded.

Ensure the file **TestData.csv** is on the MATLAB Path.

In the Current Folder Window in MATLAB, Right-click on **TestData.csv** and select **Open as Text**. You will see that on each line there are two numbers. First column is Displacement in mm. Second column is the applied load in kilonewtons.

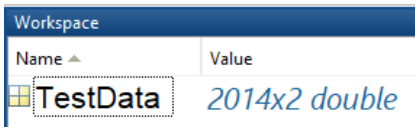
Write a script that will do the following:

- Load in the test data from the file, using the load function.

  e.g. To load the data into an array called **TestData** you would use

  ```
  >>   load TestData.csv
  ```

  This will add **TestData** to your Workspace

- Store the displacement and the load in two different variables.

  (You will need to access each column of TestData. Do not use the variable name load, this will overwrite the built-in load function!)

- Plot the load against displacement.
- Label the axes, add a title and a grid
- Plot a point on the graph where the load is maximum using a red cross.

  Use the max function to find the coordinates of the point on the graph where the load is maximum. You will need to use max with two output. Try the following example first. What do you think Index is?

  ```
  >>   A = [ 1 2 6 4 7 5 2]
  >>   [M,Index] = max(A)
  ```

- Use the text function to write “maximum load” next to the red cross.

Welcome to the IT Learning Centre

You are in the right place …

We'll be starting soon

1

MATLAB: First Steps

Dr Isaac "Izzi" Mear, FHEA
izzi.mear@eng.ox.ac.uk
Pronouns: He/Him
linktr.ee/isaacmear

Icons throughout from FreePik @ Flaticon.com

2

Ready To Learn?

The only assumed knowledge today is that you understand what a file/folder is on your computer.

Please ask questions at **anytime**.

Computing is full of technical words.
If I use a word you don't recognise – tell me!

3

Your safety and comfort are important

Where is the fire exit?

Please tell us if anything doesn't work

The welcome area has vending machines and water

Toilets are along the corridor outside the teaching rooms

4

Your safety and comfort are important

Let us know if you are too hot/cold/unwell

Please write your name on a sticker

Please silence your phone

5

Session Outline

Work through coursebook together

- Have **printed and digital** version of coursebook open
- Short talks to introduce concepts, then you have a go.
- Work at your own pace! You can always finish later.

Exercises

- Via MATLAB Onramp (online course)
- Software Carpentries
- https://software-carpentry.org/lessons

6

iT Centre Learning | iT services | University of Oxford

Session Outline

Work through coursebook together

- Have **printed and digital** version of coursebook open
- Short talks to introduce concepts, then you have a go.
- Work at your own pace! You can always finish later.

Follow-up work

- Continue with exercises after the session
- Explore the **Resources** introduced in the coursebook
- Videos with today's topics in LinkedIn Learning

7

Rough Schedule

Learning Objective		Talk Length	Time for Exercises	
LO 1	Basic Programming	15min	20mins	1 hour 10 mins
LO 2	Scripts	15min	20min	
Short Break 10mins				
LO 4.1	MATLAB Documentation	5 min	15 min	1 hour 10mins
LO 3	Plotting	10 min	20 min	
LO 4.2	Speeding up Research	10 min	0 mins	
LO 5	Further Resources	5 min	25min	30 mins

8

MathWorks Account

Some of the links will require you to have a MathWorks Account.

Let's set that up now.

bit.ly/matlabportal

9

1. MATLAB Basics

10

Launching MATLAB

11

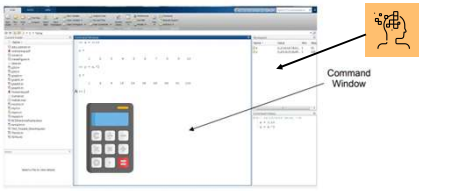
What is MATLAB?

- A software package for mathematical calculations.
- Powerful, but user friendly.
- Versatile: You can use it interactively or use it like a programming language.
- Can handle complex calculations on large data sets.
- Huge number of predefined functions to use.

12

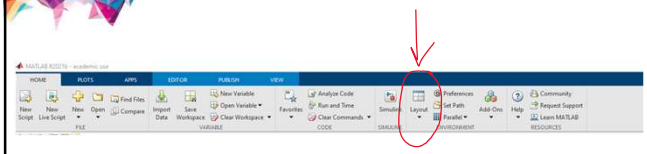
Using the Command Window

- **Command Window**: writing code 'on the fly' (i.e. without saving it)
- **Workspace**: Shows what is in the memory



13

Layout



14

Demo #1

- MATLAB as a calculator 1*2, 4*b, 4.*b
- Variables: Names must not start with numbers, and have no spaces
- Datatypes: Numbers and Arrays
- Errors: Element-wise
- MATLAB Quick Reference: Workspace and Command Window, Generating Vectors

15

Assignments

Most MATLAB statements take the form: **Name = value**

e.g. yield = 67

*We say: The variable called **yield** is assigned the value **67**.*

x = 5 ✓ *The variable called x is assigned the value 5.*

5 = x ✗ *You cannot have a variable called 5. Variable names cannot start with a number.*

16

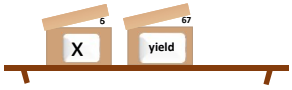
Ordering

y = x+ 4
x = 5 ✗

x = 5
y = x+ 4 ✓

17

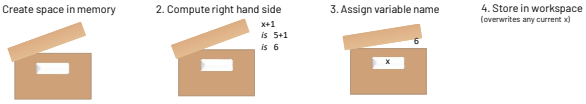
Workspace



Any variables assigned are placed in the 'workspace' storage area (imagine a shelf).

You can use recursive statements : **Name = name + value**

x = x + 1 ✓



18

Lists of Numbers (Matrices)

MATLAB stands for **MAT**rix **LAB**oratory.
It has been designed to use matrices.
It manipulates matrices very efficiently.

$$\begin{bmatrix} 1 & 5 & 7 & 8.8 & 7.42 \\ 4.6 & 72 & 5i & 92 & 44 \\ 7i & 65 & 77 & 63 & 4.2 \\ 3+5i & 22 & 4.11 & 54 & 54 \\ 17.3 & 111 & 32 & 21 & 21 \\ 83.2 & 230 & 12 & 41 & 16 \end{bmatrix}$$

Matrix

$$[12 \quad 15 \quad 2i \quad -17 \quad 3.89]$$

Row Vector

$$\begin{bmatrix} 12 \\ 18 \\ 8.8 \\ 17 \\ 5 \\ 1.01 \end{bmatrix}$$

Column Vector

34.2

Single Number

19

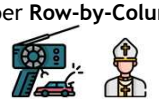
Matrices

M(2,4)=92

Row by **C**olumn

$$M = \begin{bmatrix} 1 & 5 & 7 & 8.8 & 7.42 \\ 4.6 & 72 & 5i & 92 & 44 \\ 7i & 65 & 77 & 63 & 4.2 \\ 3+5i & 22 & 4.11 & 54 & 54 \\ 17.3 & 111 & 32 & 21 & 21 \\ 83.2 & 230 & 12 & 41 & 16 \end{bmatrix}$$

You need to come up with a way to remember **Row-by-Column.**
You may want to use one of the following:
- **R** before **C** like in "**R**oman **C**atholic"
- **RC** like a remote **c**ontrol car



20

Matrix and Array Operators Multiplication

```
>> A * B
```

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 1\times5+2\times7 & 1\times6+2\times8 \\ 3\times5+4\times7 & 3\times6+4\times8 \end{bmatrix} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

```
>> A .* B
```

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} .* \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 1\times5 & 2\times6 \\ 3\times7 & 4\times8 \end{bmatrix} = \begin{bmatrix} 5 & 12 \\ 21 & 32 \end{bmatrix}$$

21

Using Built-in Functions

- zeros(2,3)
- sin(pi)
- mean([30 35 30])

22

Accessing Elements in Matrix

The format is always VariableName(row,column)

You can use : to access all the row or column
e.g. M(:,3) gets every row and just column 3 of matrix called M

You can use ranges to access parts of data
e.g. M(2:4,5) gets row 2 through 4 and just column 5

23

Programming Tips

- Coding is case sensitive
- Order is important
- There are many ways to do the same thing
- Computational thinking comes with practice
- Error messages are your friend

24

2. Using MATLAB Scripts

25

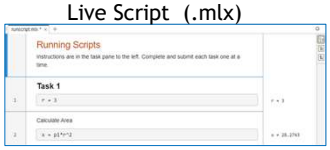
What are Scripts and why use them?

A file with multiple lines of code that can be reused.

Regular Script (.m)

```
%% Task 1

r = 3

% Calculate area
x = pi*r^2
```

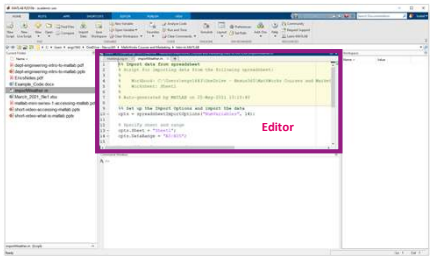
Live Script (.mlx)



26

Using the Editor

Editor: For saving bits of code to run again and again



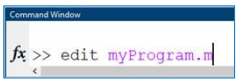
27

Creating New Scripts

Open the Editor with Command “edit”

Use the Ribbon Menu

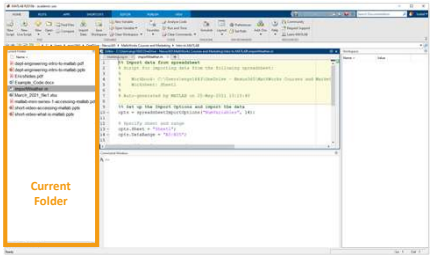
Open a specific script with Command “edit filename”



28

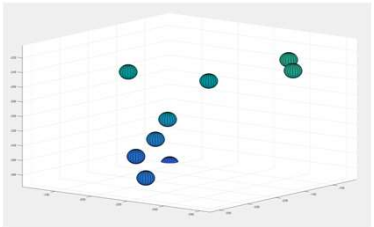
MATLAB Path

Current Folder: Shows files MATLAB can **see and use**



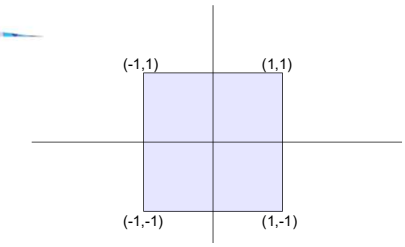
29

Demo



30

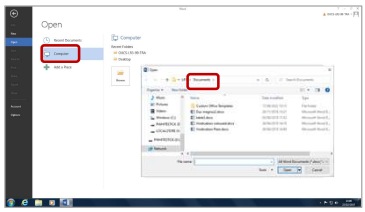
Have a Go (Moving Shapes)



$$\begin{matrix} x \\ y \end{matrix} \begin{bmatrix} -1 & 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

31

Today, the exercise files are in the **Documents** folder



MATLAB 2022a in teaching rooms

32

Functions

More powerful than scripts, as they keep variables "protected" inside.

Have inputs/outputs.

33

4. Understand the tasks that MATLAB can help with

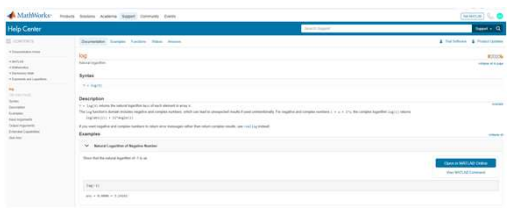
4.1 Reading documentation to find examples/get help

4.2 Speeding up research using computational techniques (loops + checks)

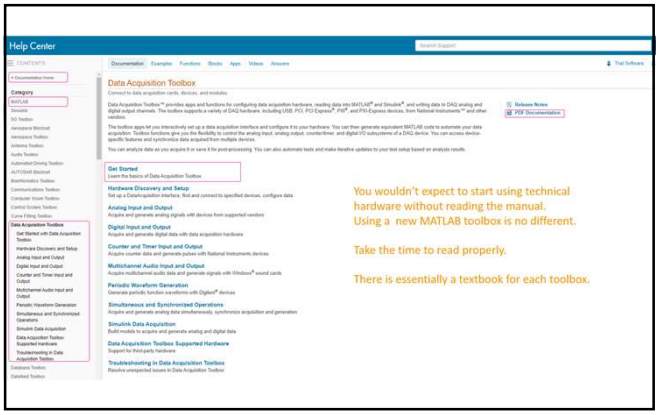
34

Using MATLAB Documentation

- `doc searchTerm`
- `help functionName`



35



36

iT Centre Learning iT services UNIVERSITY OF OXFORD

3. Importing and Plotting Data

37

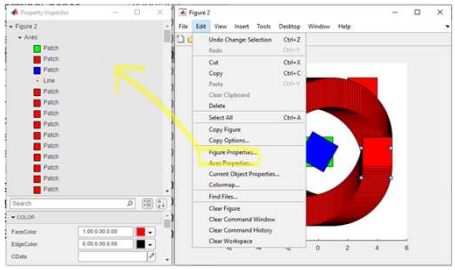
Plotting Demo

```
% Generate some (x,y) co-ordinates
x = -5:1:5
y = x.^2

% Plot the co-ordinates
plot(x,y,'rx-')
```

- Using MATLAB Documentation on https://uk.mathworks.com/help/matlab/ref/plot.html
- Name Valued Pairs

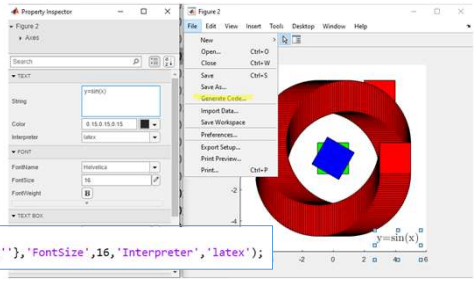
38

Plot Properties



39

Plot Properties



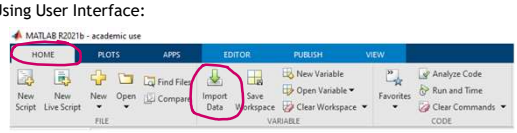
40

Importing Data

- Using Commands:

```
load TestData.csv

patient_data = readmatrix('CarpentriesInflammationData/inflammation-01.csv')
```

- Using User Interface:



41

4. Understand the tasks that MATLAB can help with

4.1 Reading documentation to find examples/get help	4.2 Speeding up research using computational techniques (loops + checks)

42

Speeding up Research: Loops

For Loop: For repeating something a **known** number of times

While Loop: For repeating something an **unknown** number of times

43

Speeding up Research: Checks

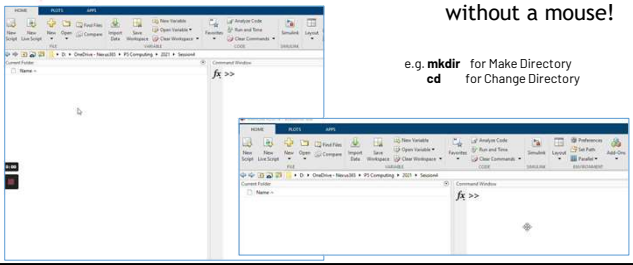
If statement: Checking if something is true/false

44

Command Line

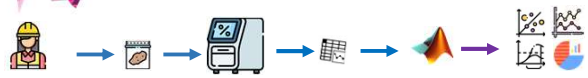
Text based commands that allow you to control the operating system without a mouse!

e.g. **mkdir** for Make Directory
cd for Change Directory



45

Example Research Workflow 1/3



Details of sample site (PDF/doc)
Photographs of sample + sample site

Data from testing (spreadsheets)

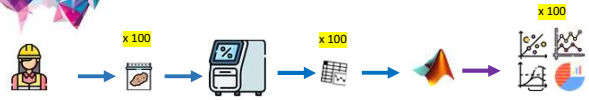
MATLAB code
Plots from analysis

File Storage: One folder per sample

- Details of sample site (PDF/doc)
- Photographs of sample + sample site
- Data from testing (spreadsheets)
- MATLAB code
- Plots from analysis

46

Example Research Workflow 2/3



Details of sample site (PDF/doc)
Photographs of sample + sample site

Data from testing (spreadsheets)

MATLAB code
Plots from analysis

File Storage: One folder per sample

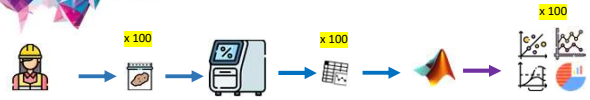
- Details of sample site (PDF/doc)
- Photographs of sample + sample site
- Data from testing (spreadsheets)
- MATLAB code
- Plots from analysis

Open folder, run MATLAB code to generate plots for analysis

100 clicks

47

Example Research Workflow 3/3



Details of sample site (PDF/doc)
Photographs of sample + sample site

Data from testing (spreadsheets)

MATLAB code
Plots from analysis

Fix! Create Script to Automate Process:

- Use **mkdir** to make new folder to save spreadsheets
- Create a 'loop' to repeat task over and over for different samples
- Use **cd** to open correct folder, then automatically run code and save images

Open folder, run MATLAB code to generate plots for analysis

100 clicks

48

Command Line Practice

Not enough time to cover Command Line in depth during this course, but...

In MATLAB practice simple commands like:
cd (changing directory)
and **mkdir** (make directory)

5 Hour Course:
https://swcarpentry.github.io/shell-novice/

49

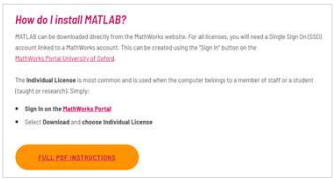
5. Installing MATLAB + Further Resources

iT Centre Learning

50

Everyone in the University can access MATLAB + all toolboxes for free!

www.eng.ox.ac.uk/matlab for install links & FAQ



Need Single Sign On

&

MathWorks Account

51

Three License Types

There are three different licenses:

- **Network** - for University computers on the University Network (or VPN)* — Uni network required to use
- **Designated Computer** - for University owned computers* — no network required to use
- **Individual** - for personally owned computers — no network required to use

* Requires file installation key from MATLAB Local Representatives

52

MATLAB via Browser

- Can also use MATLAB/Simulink Online
  Matlab.mathworks.com
  which has every toolbox!
- Via mobile/tablet App
- Can use MATLAB Drive (cloud storage)

53

MATLAB Academy: Free Online Training

MATLAB Academy -
matlabacademy.mathworks.com

- Free high-quality online courses
- Self paced and completed in browser
- Direct coding practice, quizzes, videos

- Beginner course MATLAB "Onramp"
  ~2 hours of content

54

iT Centre Learning

UNIVERSITY OF OXFORD

MATLAB Academy Courses

Foundation Skills

MATLAB Fundamentals (LINK)
Comprehensive introduction to data analysis, visualization, modeling, and programming in MATLAB.

MATLAB for Data Processing and Visualization (LINK)
Learn to import data from mixed files, manipulate and group data, and create custom visualizations.

MATLAB Programming Techniques (LINK)
Learn about managing data efficiently, testing & debugging code.

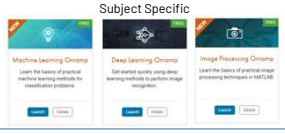
Computational Mathematics

Solving Nonlinear Equations with MATLAB

Solving Ordinary Differential Equations with MATLAB

Introduction to Statistical Methods with MATLAB

Subject Specific

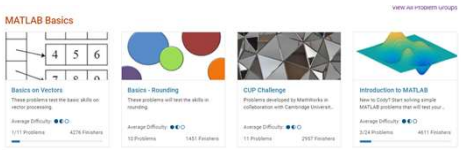


55

Cody Challenges

mathworks.com/matlabcentral/cody/

- Short coding exercises created by MathWorks
- They are not easy, but help you learn how to think computationally!

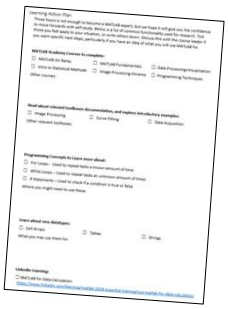


56

Learning Action Plan

Let me know if you want a chat to discuss your specific use of MATLAB

bit.ly/matlabusergroup



57

Find the resources for the workshop in our IT Learning Portfolio

Download the files (and more) from the IT Learning Portfolio at skills.it.ox.ac.uk/it-learning-portfolio

58

izzi.mear@eng.ox.ac.uk

59

iT Centre Learning iT services University of Oxford

MATLAB Quick Reference

Version 1.1

File Management

cd *name*	Change the current directory to folder called *name.*
mkdir *name*	Make a new directory with the given *name.*
dir	List contents of current directory (aka folder).
ls	List contents of current directory (aka folder).

Saving, Exporting and Importing Data

save *filename*	Save all variables to the file *filename*.mat
save *filename variable*	Save only the variable *variable* to the file *filename*.mat
load *filename*	Load in variables from the file *filename*.mat
load *filename.ext*	Load from the file *filename*, to a variable called *filename*

Working with Text

chr = 'hi there'	A character uses single quotes (each letter is an element)
chr = strcat('Good','Day')	Combine all elements in one character array.
str = "hi there"	A string uses double quotes (all one element)
str = join(array)	Combine all elements in string array into one string.
str = num2str(x)	Convert the number in x to text.
str = num2str(pi,'%0.5f')	Gives a string variable containing pi to 5 decimal places.

Display Format

display(a)	Suppose a = 5. This would print "a = 5"
disp(a)	Same as above, but with out the "a = "

format short	4 decimal places (dp)	**format long**	15 decimal places (dp)
format shortE	Scientific notation, 4dp	**format longE**	Scientific notation, 15dp
format shortEng	Engineering notation, 4dp	**format longEng**	Engineering notation, 15 dp

User Interaction

[x,y] = ginput(1)	Graph coordinates of a clicked on point.
x = input('What is x ? ')	Ask the user for a number

Help

help *function*	Help on a particular function called *function*.
doc *function*	Full documentation on function.

Workspace and Command Window

clear variables	Remove all variables from memory.
clear *var1 var2*	Remove specified variables *var1* and *var2* from memory.
clc	Clears the Command Window

Stop MATLAB code from running by clicking the Command Window and then use Ctrl + C.

Generating Vectors

Specific spacing	**start : end**	**x = 1 : 5** generates x = [1 2 3 4 5]
	start : separation : end	**y = 0:5:20** generates y = [0 5 10 15 20]
Specific no. of points	**linspace(start,end,n)** n = number of elements	**linspace(0,10,5)** generates [0 2.5 5 7.5 10]
	logspace(d1,d2,n) logarithmically spaced between $\mathbf{10}^{d1}$ and $\mathbf{10}^{d2}$.	**logspace**(-1,2,4) generates [0.1 1 10 100]

Generating Matrices

x = [1 2 3; 4 5 6; 7 8 9]	Comma or space between elements. Semicolon or return for new row. Enclosed in square brackets.
x = [1,2,3 4,5,6 7,8,9]	
a = [exp(0) sqrt(4) 1+2]	Each element can be an expression.
C = [A , B]	A and B are matrices with the same number of rows.
C = [A ; B]	A and B are matrices with the same number of columns.

zeros(n)	n by n matrix where each element is zero.
zeros(m,n)	m by n matrix where each element is zero.
eye(n)	n by n identity matrix.

Indexing

An **index** is the **position** of an element in a vector. MATLAB indexing starts at 1.

Vector Indexing	
Examples	
A(3)	The third element in vector A.
A([3 8 11])	The third, eighth and eleventh elements of A.
A(3:11)	The third to eleventh elements of A.
A(11:end)	All elements from the eleventh to the last element

Matrix Indexing	
variable(R,C)	**R is a vector of rows and C a vector of Columns**
Examples	
M(2,3)	The element in the second row and third column of M.
M(2, [3 8 11])	Second row, third, eighth and eleventh column.
M(4,3:11)	Forth row, third to eleventh column.
M(11:end,2)	Eleventh to the last row, second column.
M(5,:)	Fifth row, all columns.
M(11:end,:)	Eleventh to the last row, all columns.

Built in Matrix Functions

det(A)	Determinant
norm(A)	Norm
inv(A)	Inverse
[v,d]=eig(A)	d = Eigenvalues, v = Eigenvectors
sqrtm(A)	The matrix square root.
expm(A)	The matrix exponential base e.
logm(A)	The matrix natural logarithm.

Built in Functions

Hit ***fx*** icon next to prompt for function browser. Only selected functions shown here:

abs(x)	The absolute value (or modulus)	**round(x)**	Round to the nearest integer.
sqrt(x)	The square root.	**ceil(x)**	Round up.
exp(x)	The exponential base e. e^x	**floor(x)**	Round down.
log(x)	The base e $\log_e(x)$	**fix(x)**	Round towards zero.
log10(x)	The log base 10. $\log_{10}(x)$	**rem(x,b)**	Remainder of x divided by b.

Trigonometry

sin(x)	Sine x in radians.	All these functions are also available for : cosine `cos(x)`, tangent `tan(x)` secant `sec(x)` cotangent `cot(x)` cosecant `csc(x)`
sind(x)	Sine x in degrees.	
asin(x)	Arcsine, the inverse of sine, in radians.	
asind(x)	Arcsine in degrees	
sinh(x)	Hyperbolic Sine	
asinh(x)	Inverse hyperbolic sine.	

Complex Numbers

real(z)	The real part of z.	**imag(z)**	The imaginary part of z.
abs(z)	The modulus of z.	**angle(z)**	The phase angle of z.
conj(z)	The complex conjugate of z.		

Statistics

max(x)	Maximum	**median(x)**	Median	**var(x)**	Variance
min(x)	Minimum	**mean(x)**	Average	**std(x)**	Standard Deviation

Polynomials

p = [1 2 3 4 5] can represent the polynomial $x^4+2x^3+3x^2+4x+5$	
y = polyval(p,x)	Evaluate polynomial for each value in x.
roots(p)	Roots of polynomial.
p = poly(*roots*)	Polynomial with given roots.
p = polyfit(x,y,n)	Best fit of **x,y** data points to n^{th} order polynomial.

Operators

Adding a **dot** means element by element (one at a time). Think dot = single element.

Matrix Operations		Array or Element by Element	
*	**Matrix Multiplication**	.*	**Element by Element Multiplication**
^	**Raise Matrix to a power**	.^	**Raise each Element to a power**
'	**Transpose matrix (conjugate if complex)** *A'*	.'	**Transpose matrix (no complex conjugation)** *A.'*

For matrix division, see the slash \ as an arrow pointing up, it points to the matrix that will be inverted: See A\B as A↖B, meaning $A^{-1}B$ and see B/A as B↖A, meaning BA^{-1}.

/	**Right Matrix Division** $b/A = bA^{-1}$	./	**Element by Element Right Division** A./B=B.\A
\	**Left Matrix Division** $A\backslash b = A^{-1}b$	.\	**Element by Element Left Division** A.\B=B./A

Relational Operators		Logical Operators	
<	**Less Than**	&	**And**
<=	**Less Than or Equal**	|	**Or**
>	**Greater Than**	~	**Not**
>=	**Greater Than or Equal**		
==	**Equal**		
~=	**Not Equal**		

Special Numbers

pi	π
inf	∞

i	$\sqrt{-1}$
j	$\sqrt{-1}$

NaN means **N**ot **a** **N**umber (e.g. 0/0)

Corrections or suggestions to izzi.mear@eng.ox.ac.uk

Plotting Commands

plot(y)	Plot y against index number (1,2,3 etc)
plot(x,y)	Plot y against x
plot(x,y,'r+')	Plot y against x using red plus signs.

Symbol	Line/Mark Type	Symbol	Colour
.	Point	r	Red
o	Circle	g	Green
x	X mark	b	Blue
+	Plus sign	y	Yellow
*	Stars	m	Magenta
-	Solid line	c	Cyan
:	Dotted line	w	White
-.	Dash dot line	k	Black
--	Dash Line		

Other Types of Plot	
fill(x,y,'r')	Red filled graph
bar(x,y)	Bar graph
loglog(x,y)	x & y log scale
semilogx(x,y)	x log, y linear
semilogy(x,y)	x linear, y log
polar(theta,r)	Polar plot
surf(x,y,z)	3D surface
mesh(x,y,z)	3D mesh
plot3(x,y,z)	3D line plot

title('Title')	Graph title
xlabel('X axis') ylabel('Y axis')	Label the x-axis or y-axis
text(x,y,'My Text')	Place text at coordinates x,y.
grid	Place a grid on the graph.
hold on	Add any new plot to the current graph.
hold off	Replace current plot with any new plot.
tiledlayout(r,c)	Split figure into **r** rows by **c** columns of tiles.
nexttile	Move to the next tile
axis([minX maxX minY maxY])	Set the limits of graph in X and Y (4 element vector)

Animation

h = figure	Open a new window for plotting
figure(h)	Change to plotting in figure h.
delete(h)	Delete figure h.
clf	Clear current figure.
drawnow	Force the graph to update now.
pause	Wait for the user to press a key.
pause(5)	Wait for 5 seconds

Programming

For Loops: Repeat code a set number of times

General form

```
for variable = vector
    statement
    statement
      etc
end
```

Examples:

```
% Loop directly over vector
for k = [1 7 3 pi i]
   disp(k)
end
```

```
% Loop through vector using index
A = [1 7 3 pi i];
for k = 1:5
   disp(A(k))
end
```

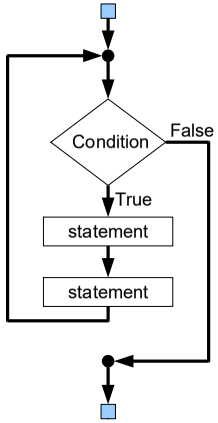
While Loops: Repeat code an unknown number of times

General form

```
while condition
   statement
   statement
     etc
end
```

Example

```
A = 7;      %Find the square root of A.
x = 1;      %First guess
err = 1;    %Set error to get started

% Newton Raphson Iteration
while err>0.0001
    x = (x.^2+A)./(2*x);
    % calculate the error
    err = abs(A-x.^2);
end

disp(x);
```



Functions: Custom code with separate variables

Function Definition	Using the Function

`function [out1,out2] = myfunc(a,b)`
`% The first block of comments`
`% defines what is printed out when`
`% you type  help myfunc`
 	
`% Other comments not part of help`
`out1 = a + b;`
`out2 = 2*out1 + sin(0.5*out1);`
	`[s1,s2] = myfunc(6,9)  % Get both outputs`
	
	`s = myfunc(5,4)  % Just get first output`
	
	`% Note! When using (aka 'calling') a function we do not need to use same variable names as in definition.`

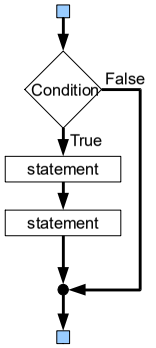
If Statement: Checking if something is true / false

General form

```
if condition
   statement
   statement
     etc
end
```

Example:

```
% Generate random num between 1-10
x = ceil(rand(1)*10)
if rem(x,3)==0
    disp('x is divisible by 3')
end
```

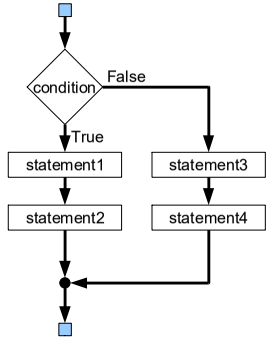


If - else to switch between **two** states:

```
if  condition
   statement1
     etc
else
   statement2
     etc
end
```

Example:

```
% Generate random num between 1-10
x = ceil(rand(1)*10)
if rem(x,3)==0
    disp('x is divisible by 3')
else
    disp('x is not divisible by 3 ')
end
```



elseif to switch between **multiple** states:

```
if condition1
    statement1
elseif condition2
   statement2
else
   statement3
end
```

Example:

```
% Generate random num between 1-10
x = ceil(rand(1)*10)
if rem(x,3)==0
  disp('x is divisible by 3')
elseif rem(x,2)==0
  disp('x is divisible by 2 ')
else
  disp('x is not divisible by 2 or 3')
end
```