MATLAB:

A Comprehensive Introduction

October 2018

Catherine Paverd

catherine.paverd@eng.ox.ac.uk

Operators & Control Statements

- Operators and control statements allow you to write programs and make a computer perform tasks in a similar manner to how humans think

Operators

Perform arithmetic, numerical and logical operations

Conditionals and Control Statements

Control the flow of the program and the number of times a section of code is run

Add eggs

Chocolate **or** caramel

As much milk as water

If the mixture has risen, remove the cake

Let the cake cool **for** ten minutes

Operators - Arithmetic

- General form of arithmetic operators is:

Operand1 **Operator** **Operand2**

Operator	Description
+ and -	Addition and subtraction. A+B adds A and B. A-B subtracts B from A.
.*	Array multiplication. A.*B is the element-by-element product of the arrays A and B.
.^	Array power. A.^B is the matrix with elements A(i,j)to the B(i,j) power.
./	Array right division. A./B is the matrix with elements A(i,j)/B(i,j).
.\	Array left division. A.\B is the matrix with elements B(i,j)/A(i,j).
*	Matrix multiplication. A*B is the linear algebraic product of A and B. For non-scalar A and B, the number of columns of A must equal the number of rows of B.
^	Matrix power. In the non-scalar power case, the calculation here involves eigenvalues and eigenvectors.
/	Matrix right division. B/A = (A'\B')'
\	Matrix left division. X= A\B is the solution to the equation AX=B.

Operators - Arithmetic

- Two types of arithmetic operations
  - **Array Operations:** Carried out element by element
  - **Matrix Operations:** Defined by the rules of linear algebra
- The full stop character (.) distinguishes the two types of operations
- E.g. **Matrix A .* Matrix B** = Elementwise multiplication of the elements in A by the same position elements in B
- E.g. Matrix A * Matrix B = Matrix multiplication

$$AB = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{bmatrix}$$

Operators - Relational

- General form of relational operators is:

Operand1 **Operator** **Operand2**

Operator	Description
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to (NB: this is EQUIVALENCE, NOT ASSIGNMENT)
~=	Not equal too

Operators - Relational

- Same form as arithmetic operators
- Compare operands **quantitatively**
- Always operate element by element
- If one operand is a scalar and the other is a matrix, the scalar is compared to each element of the matrix and a matrix is returned
- The outcome of the relational operator is binary – i.e. MATLAB will return a 1 (true) or a 0 (false) for each comparison it makes

Operators - Logical

- General form of logical operators is:

Operand1 **Operator** **Operand2**

	Operator	Description
Matrices	&	Logical AND. A&B returns true (1) for every element location that is true (non-zero) in both arrays, and false (0) for all other elements.
Matrices	|	Logical OR. A|B returns true (1) for every element location that is true (non-zero) in either one or the other, or both arrays, and false (0) for all other elements.
Matrices	~	Logical NOT. Complements each element of the input array. Non-zeros become false while zeros become true.
Scalar/ Logicals	&&	Short-circuit AND. If the outcome of the first operation is false, then the overall answer is false and the second expression is not evaluated.
Scalar/ Logicals	||	Short-circuit OR. If the outcome of the first operation is true, then the overall answer is true and the second expression is not evaluated.

Operators - Logical

- Perform logical calculations and return logical values
- For non-logical numerical data types:
  - **Zeros = False**
  - **Ones = True**
- Two types of operators:
  - Element-wise (bitwise) operators for arrays
  - Short circuits operators for scalars

Conditionals

- Control what code a program executes based on a test condition

IF Test Expression

Do Some code

END

IF Test Expression

DO Some Code

ELSEIF Another test expression

DO Some code

⋮

END

SWITCH Variable

CASE Some value
DO Something

CASE Some other value
DO Something Else

OTHERWISE
DO something else

END

Conditionals – if, elseif, else

- Simple test of an expression – code will execute if the ‘test expression’ is true
- Can combine multiple test expressions using logical operators
  - E.g. if *condition1* && *condition2*
    Do something
- Can use multiple ‘nested’ if statements
  - E.g. if *condition1*
    if *condition2*
    Do something

IF Test Expression

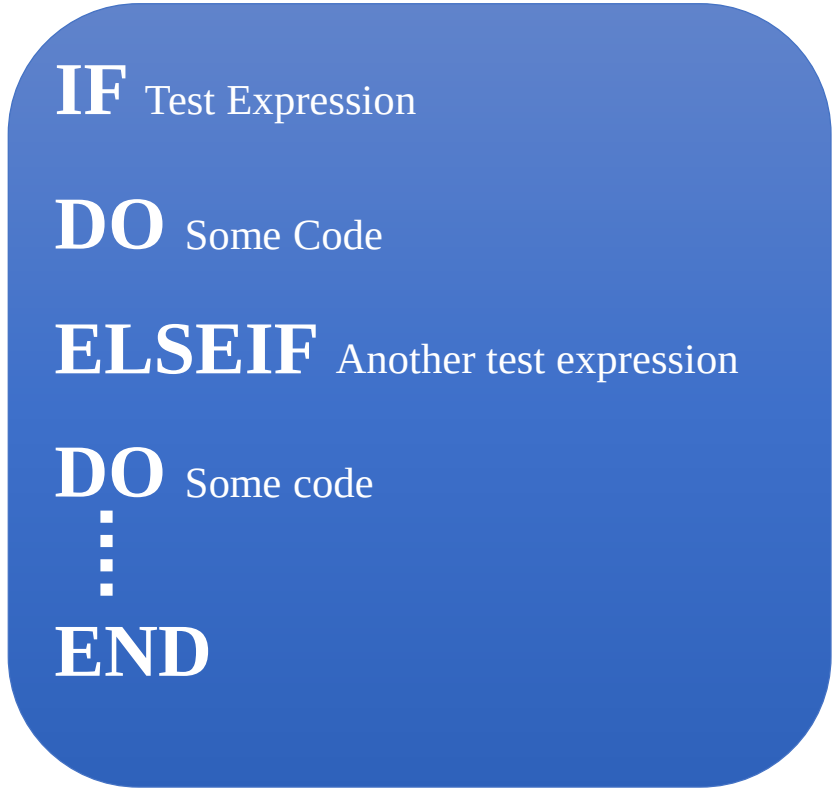
Do Some code

END

Conditionals – if, elseif, else

- Can test for multiple conditions using if and ifelse statements. The final ‘else’ statement is optional.

  - **if *condition1***
    Do something
    elseif *condition2*
    Do something else
    elseif *condition3*
    Do something else
    else
    Do something else
    end



Conditionals – switch

- Similar to if, elseif, else statements, however switch statements are usually used when a limited number of cases are expected
- Useful for checking a user input against an exact match
- The cases cannot be used in conjunction with relational operators, such as < or >
- If the first case is true, MATLAB will stop execution of the statement

SWITCH Variable

CASE Some value
DO Something

CASE Some other value
DO Something Else

OTHERWISE
DO something else

END

Loops

- Enable a block of code to be repeatedly executed
- *For* and *while* loops are the two loops used in MALAB

FOR index = start: increment: end
DO Some Code
END

WHILE something is true
DO some code
END

Loops - for

- Execute a block of code a set number of times based on the index and conditions defined in the first line
- **The code in the loop should not modify the index variable**
  - Note: if you do modify it, later versions of MATLAB may still execute the code correctly, but it is very bad practice

```
FOR index = start: increment: end
        DO Some Code
END
```

Loops - while

- Execute a block of code a as long as the test expression is true
- Should define the test variable before the loop
- **Code inside the loop must change the value used in the test expression at some point**
  - Note: if you do not modify it, you are likely to end up with an **INFINITE LOOP**, where MATLAB will never exit the loop
  - To exit an infinite loop, type CTRL+c in the command window

WHILE something is true

DO some code

END

Loops – Continue and Break

- **Continue** and **Break** statements can be used to interrupt the ongoing repeats in a loop
- ***Continue*** immediately passes control to the next repeat of the loop in which it appears, skipping any remaining code in the body of the loop
- ***Break*** terminate the execution of a loop and passes control to the point following the end line of the loop
- In nested loops, continue and break statements only apply to the innermost loop containing that statement

End of Part 1
Please start on Exercises 6, 7 and 8

Part 2
Programming

MATLAB Files

- We have already introduced the **MATLAB editor** and the simplest kind of MATLAB program file (M-file): the **script**.

- There are actually two kinds of M-files:

  - **Scripts**: they do not accept inputs and they do not return output arguments. They operate on data in the workspace.

  - **Functions**: they accept inputs and they return output arguments. They operate in private workspaces: internal variables are local to the function.

Scripts

- Scripts contain a **list of commands**
- When you run a script, MATLAB executes the commands in the M-file
- The result is the same as running the commands one-by-one in the command window
- Scripts share the base workspace with your interactive MATLAB session
- They operate on **existing data** in the workspace, or they can create **new data** on which to operate
- Any variables that scripts create remain in the **workspace** after the script finishes so you can use them for further work.

Scripts – Comments & Housekeeping

- **% symbol preceding text**: MATLAB comment
  - Comment text is not interpreted as code.
  - The first comments at the top of the M-file explain the usage of the script/function.
  - Additional comments should be used in the body of the M-file to explain particular sections of the code.
  - Comments are also used for temporarily disabling part of the code. This is referred as ‘commenting out’ code.

- Housekeeping code is used to keep MATLAB workspace and display area tidy.
  - **clear/ clear all**: to clean up the workspace
  - **close/ close all**: to close figures
  - **home/ clc** : to tidy up the command window

- Housekeeping code is useful at the start of scripts

Scripts - Body

- It is the **main subsection of executable code**.

- There are two main ways to run a script in MATLAB.
  - From the Command Window, you can enter the script name. Note, this only works if the script is either in the current folder or the path.
  - From the Editor, you can press F5, or click the Run button

Functions

- They are program files that:
  - Can **accept input** arguments
  - Can **return output** arguments

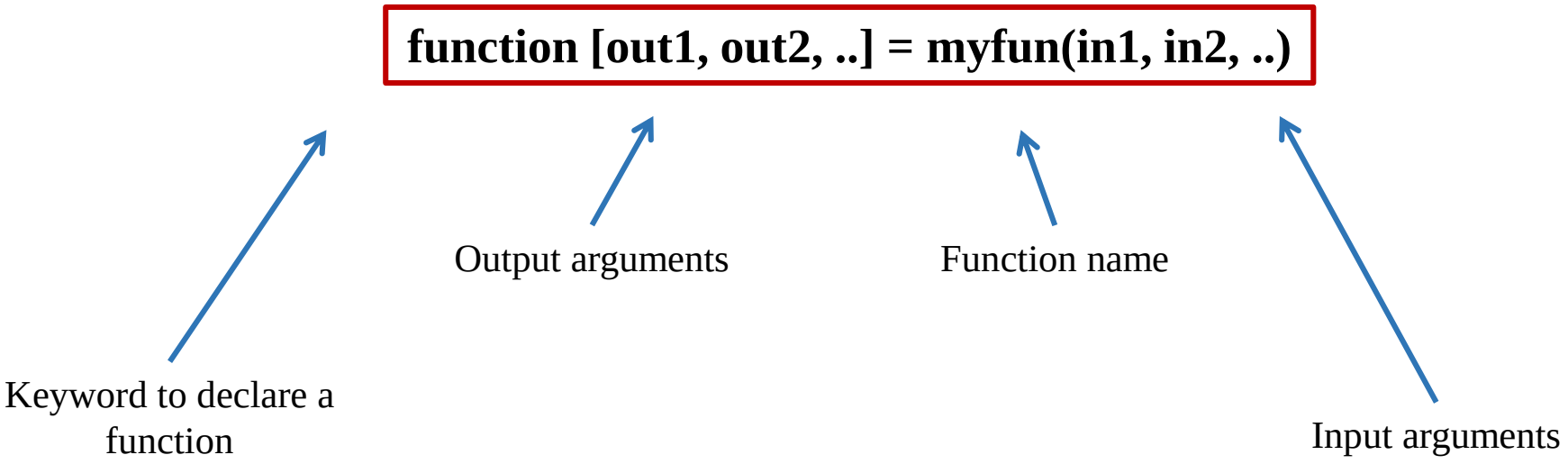
- The **M-file name** must be the **same as the function name**

- Each function operates on variables within its own **private workspace**, which is separate from the workspace you access at the MATLAB command prompt

- The input arguments are the initial contents of the private function workspace, and the output arguments are what is returned to the workspace from which the function was called.

Functions

- ***function*** is a special keyword used to declare functions.
- The function declaration must be the first executable line of any MATLAB function.



Functions – Body

- As with scripts, the function body is the main subsection of executable code

- Attention should be given to:
  - The use of the function inputs in the function body
  - The correct assignment of the output of the functions to the output arguments declared at the beginning of the function.

- As functions generally require inputs, there are two ways of running them:
  - Inputs are manually entered in the Command Window and the function name along with input and output arguments is typed in the command window.
  - Inputs are created in a script/ function and the function is called through the script/function.

Programming

A MATLAB problem is usually quite complex…

~~Large scripts/functions~~

Modular programs from simple building blocks are more efficient!!

End of Part 2
Please start on Exercises 9 & 10