Advanced MATLAB

Dawn Walker[1] Avgoustinos Vouros[1] Marzieh Tehrani[1]

[1] Department of Computer Science
University of Sheffield

Acknowledgements

Original slides created by:

- Dawn Walker (Dr)
- Tiago V. Gehring (PhD)

Revised and modified slides created by:

- Avgoustinos Vouros

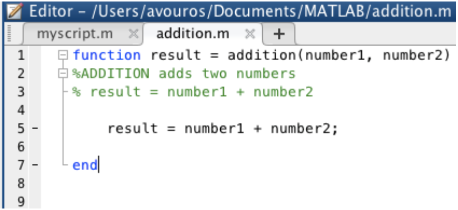
Creating and Running a Function

Creating and Running a Function

- Portions of code that can be reused or that have to be parametrised and run often should be stored into functions.
- Functions are files that can accept input argument(s) (parameters) and return output argument(s).
- To create a function to to the *HOME* tab of MATLAB's Toolstrip and either click on the arrow below the *New* icon and select Function.
- function [output_1, output_2,] = function_name (input_1, input_2,)

  function's body

  end



```
function result = addition(number1, number2)
%ADDITION adds two numbers
% result = number1 + number2

    result = number1 + number2;

end
```

Creating and Running a Function

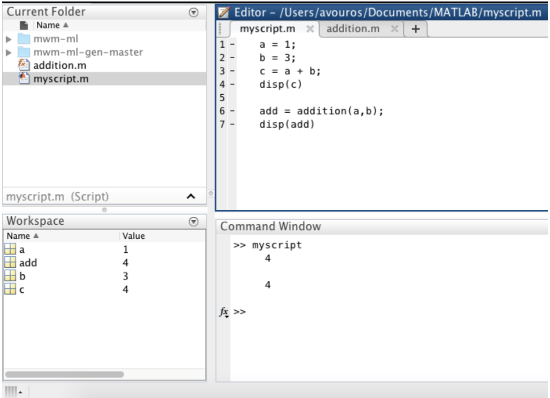
- To run a function type its name in the Command Window or inside a script followed by the input arguments.
- A function will always return its first output; in case of multiple output you have to specify a number of variable prior to the function you are calling, e.g.

$$[out1, out2] = myfunction(2, 5).$$

To skip outputs use $\sim$, e.g.

$$[\sim, out2] = myfunction(2, 5).$$

Creating and Running a Function



```
a = 1;
b = 3;
c = a + b;
disp(c)

add = addition(a,b);
disp(add)
```

Name	Value
a	1
add	4
b	3
c	4

```
>> myscript
     4

     4

>>
```

- Note: functions operate on variables within their own workspace, separate from the workspace you access at the MATLAB command prompt. the Workspace.

Global Variables

Global Variables

- Each MATLAB function has its own *local* variables, which are separate from those of other functions and from those of the base workspace.
- What if a variable is needed by a large number of functions? (e.g. environmental variable in agent model).

↓

Computational overhead for passing large collections of data between functions.

- Solution: declare a variable as **global**.

Global Variables

- global X, X=10; a=2; b=3;

- *In a function...*

  ```
  function demo_global(a,b)
          global X;
          X = a+b;
  ```

- *In the Command Window...*

  ```
  >> demo_global(a,b);
  >> X
        X = 5;
  ```

Global Variables

- Global variables can be seen and modified in any function in which they are declared.
- No need to pass the variable through input or output list.
- Declaring a variable as global and assigning a value to the variable are two separate steps.
- Any type of data can be defined as **global** if required.
- Convention to use CAPITALISED variable names for global variables to improve the readability of our code.

Cell arrays and Structured data

Cell arrays and Structured data

- **Cell arrays**
  - Cell arrays provide a way to store inhomogeneous data types e.g. matrices/vectors of different size or any other data type (including strings).
  - Cell arrays are created/accessed using braces '{ }' instead of square brackets '[ ]'.
  - Example:

```
X = eye(3);                  % matrix
Y = char('pink', 'floyd'); % string array
Z = 100;                  % integer
arr = {X,Y,Z};                  % create inhomogeneous array
m = arr{1};                  % access the first element
```

Cell arrays and Structured data

- **Structured data**
  - Multiple variables can be combined together inside a structure.
  - Structures are useful for code organisation and to pass multiple arguments with a single variable.
  - Syntax for creating a data structure in MATLAB:
    s = struct('field1',VALUES1,'field2',VALUES2,...)
  - Example:

```
earth = struct('name','earth', ...
               'mass', 5.97e24, 'radius', 6.371e6, ...
               'orbital_period', 365.2, ...
               'sun_distance', 149.6e6)
```

Cell arrays and Structured data

- **Extended example**

```
car = struct(’make’,’ford’,...
        ’model’,’focus’,...
        ’reg’,’MY CAR1’,...
        ’speed’,33),

van = struct(’make’,’robin’,...
        ’model’,’reliant’,...
        ’reg’,’MY VAN1’,...
        ’owner’,’cowboys co.’,...
        ’speed’,37)
```

Command Window:

```
car =
        make: ’ford’
        model: ’focus’
        reg: ’MY CAR1’
        speed: 33
van =
        make: ’robin’
        model: ’reliant’
        reg: ’MY VAN1’
        owner: ’cowboys co.’
        speed: 37
```

Cell arrays and Structured data

- Group the data together using a cell array:

```
>> Traffic_May1{1} = car;
>> Traffic_May1{2} = van

Traffic_May1 =
        [1x1 struct] [1x1 struct]
```

- Access data inside the cells using:

```
>> first_car = Traffic_May1{1}

first car =
        make: 'ford'
        model: 'focus'
        reg: 'MY CAR1'
        speed: 33
```

Cell arrays and Structured data

- Creating a data structure and putting it into a cell array can be done at once:

```
Traffic_May{3} = struct(’make’,’citroen’,...
        ’model’,’CV’,...
        ’reg’,’MY CAR6’,...
        ’speed’,’23’
```

Object Oriented Programming

Object Oriented Programming

- Consider an agent-based representation of a prey-predator system.

- Eating (at each iteration):
  1. Rabbits eat a certain amount of vegetation within a given radius of their current position;
  2. foxes don't eat vegetation, they eat rabbits.

Object Oriented Programming

Class Rabbit:

Attributes:

x, y, z, …

Functions:

eat, breed, …

Class Fox:

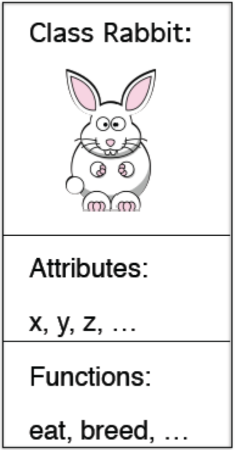
Attributes:

a, b, z, …

Functions:

eat, breed, …

Object Oriented Programming



Class Rabbit:

Attributes:

x, y, z, …

Functions:

eat, breed, …

objects rabbit1, rabbit2, ...

are instances of Class Rabbit

Object Oriented Programming

- Object Oriented Programming (OOP) in MATLAB.
  1. **classdef** command to define a class.
  2. **properties:** parameters that define the class.
  3. **methods:** functions to access and modify the class properties.
  4. Matlab allows organising classes in separate folders, the contents of which define a **class package**.
- OOP has features that could be useful for modelling agent-based systems. For example:
  - Inheritance allows the definition of child classes that can inherit selected methods from their parent class. It won't be used as part of this course.

Exercise

- Read the help for structures, cell arrays.
- Read MATLAB help for OOP: http://uk.mathworks.com/help/matlab/object-oriented-design-with-matlab.html.
- Work through the worksheet.

For Further Reading I

Mathworks
MATLAB Documentation.
https://uk.mathworks.com/help/matlab/.

MIT
A Matlab Cheat-sheet (MIT 18.06, Fall 2007).
http://web.mit.edu/18.06/www/MATLAB/matlab-cheatsheet.pdf

Thor Nielsen
Matlab Cheat Sheet.
http://www.econ.ku.dk/pajhede/Cheatsheet.pdf