SCS2211- Laboratory II

Introduction to Octave

By Dasun Bamunuarachchi

dtb@ucsc.cmb.ac.lk

University of Colombo School of Computing

Laboratory II

- Lecturers:

Dasun Bamunuarachchi – 8 Lectures

Kavinda Athapaththu, Piyumi Seneviratne – 7 Lectures

- Evaluation:

30% - Assignments, 70% - Examination

What is GNU Octave?

- A high-level scripting language, primarily intended for simplifying numerical computations.
- Provides a convenient command line interface for interactive usage.
- Octave was written by **John W. Eaton** and **many others**.
- The name Octave has nothing to do with music.
- Named after one of the author's former professor **Octave Levenspiel**, well known for his ability to do quick "*back of the envelope*" calculations.

What is GNU Octave?

- Not for symbolic computation but numerical computation.
- Compatible with **MATLAB** to a good extent.
- GNU Octave is distributed under GNU general public license.
- Because Octave is community driven software you can contribute to its development (If you love it!).

Features of Octave

- Tools for solving common numerical linear algebra problems
- Finding the roots of nonlinear equations
- Integrating ordinary functions
- Integrating differential equations
- Plotting and other graphical representations
- Extensible and customizable via user-defined functions
- Can dynamically load modules written in C++, C, Fortran, or other languages.

Who uses it?

"Centre for Advanced Computational Technologies / University of Lecce - Italy"

"German Aerospace Center (DLR)"

"University of Applied Sciences Bern"

…and many more

List of users have shared their opinions here:

https://wiki.octave.org/Who_Uses_Octave%3F

Why Octave(or MATLAB)?

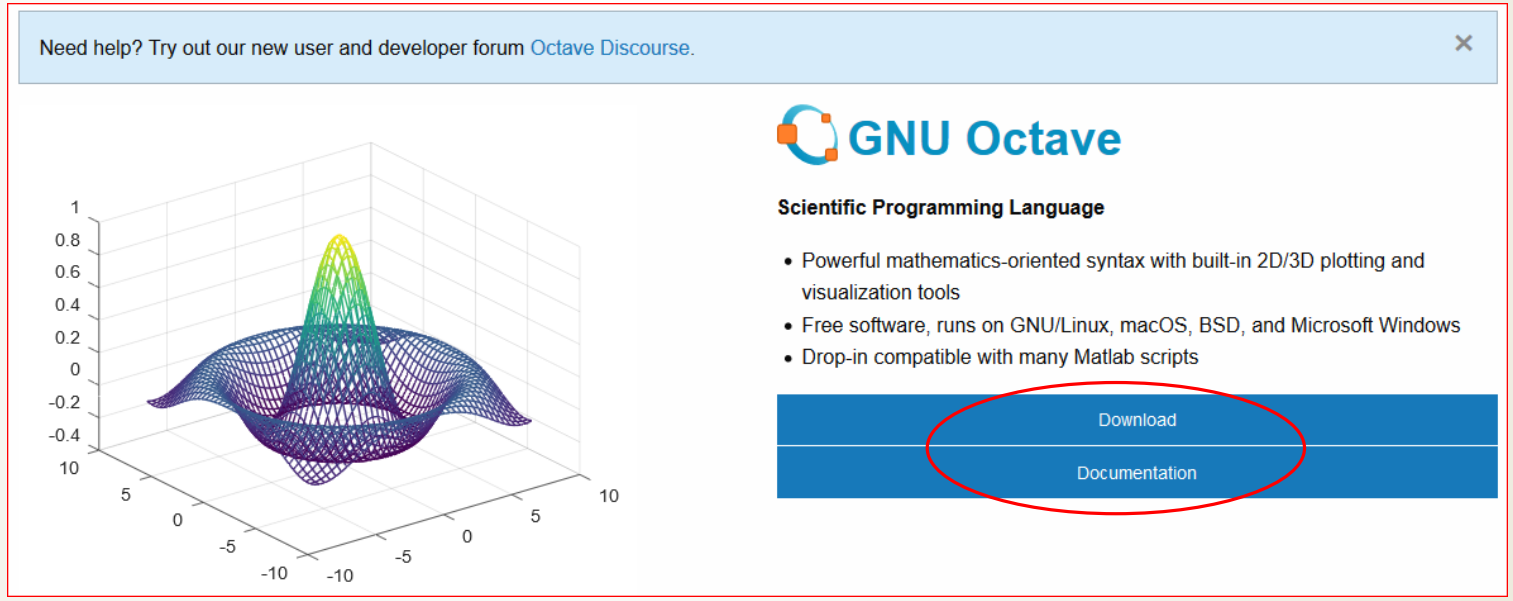
- Why not use a general purpose high-level language?
- Examples: C++, Java etc..?
- Solutions to mathematical problems take time to program
- No native support for mathematical concepts
  - Matrices
  - Graphics

Alternatives to MATLAB

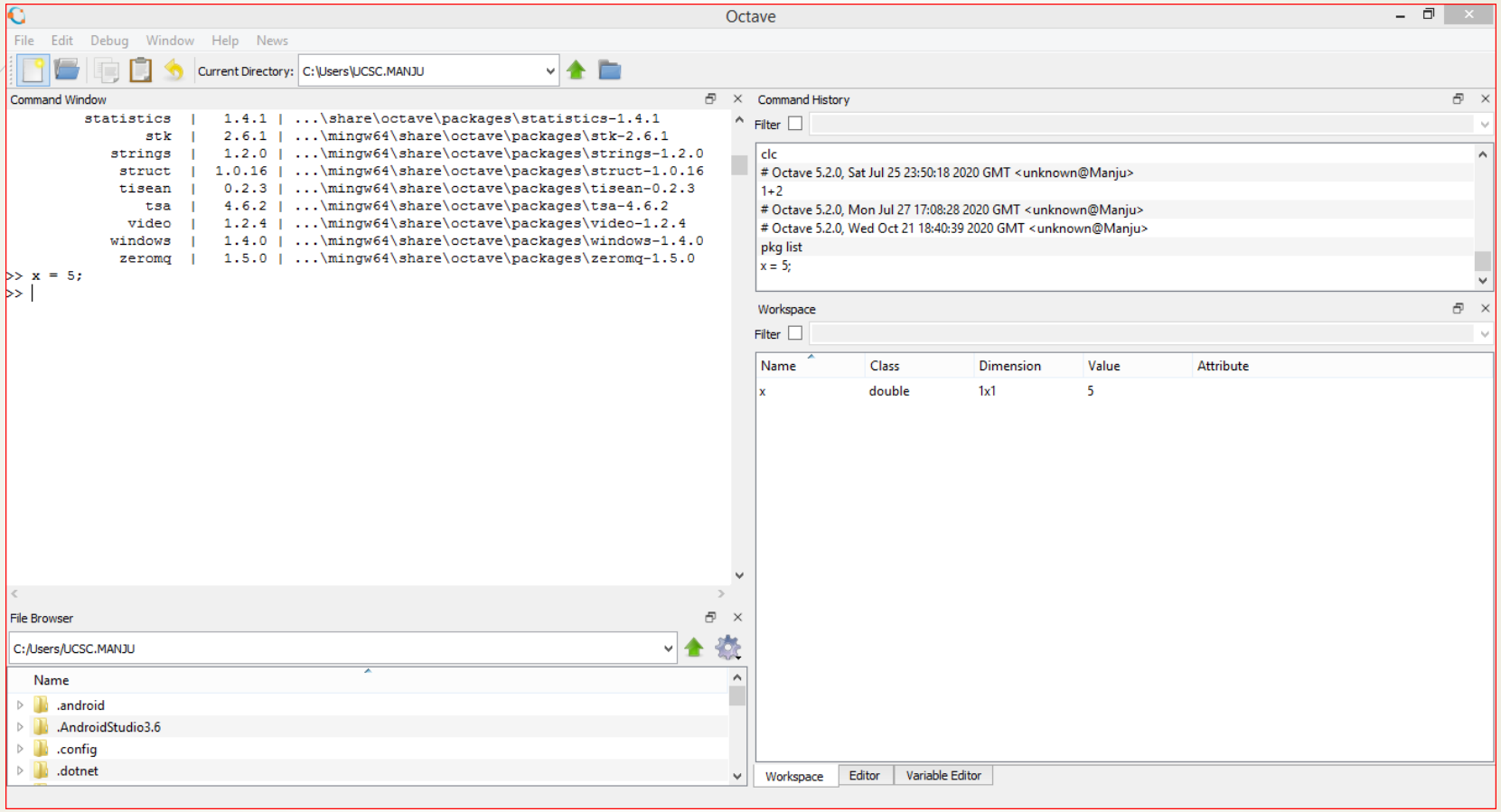
- Octave
- Scilab
- Maxima
- Julia
- Sage Math
- AnyLogic

Download & installation

- Go to Octave site follow the download option.
- Install or unzip the bundle => Installer is easier.



Octave Interface



Launching and basic operation

- In most systems **"octave"** command should load the interface.
- In Windows an icon may be present.
- Enter the command **"version"** in the command window to see the software version.
- Command **"quit"** or **"exit"** can close Octave.
- To stop an operation in the middle **Ctrl+c.**
- **ans** is a variable automatically created by Octave to assign the return value.
- Having a **semicolon(;)** at the end of the command suppresses the output.

Basic operation…

- **help <name-of-the-command>** shows information about the command

e.g.: **help quit**

- Using Octave as a calculator +,-,/,*

Command: **x = 415 * 37**

Result: **x = 15355**

- Multiplication is not implicit

>> 4(3 + 5)

error: index (8): out of bound 1

>> 4*(3 + 5)

ans = 32

Basic operation…

- Recalling last command

Up arrow key OR **Ctrl+p**

- Octave code can be commented.

Syntax is: **executable code % comment => my command**

- Alternative syntax

executable code # comment => Created: 2020-10-21

- Anything typed after % or # is ignored by the interpreter

Basic operation…

- Variable names allowed:

Legal names of variables consist of any combination of letters and digits, starting with a letter

RateVal, Phi2y, x1, X2, z1y2, Theta_1

- Variable names not allowed:

Conflicting with normal syntax of commands

Rate-Val, 2y, %x, @sign, Alpha+1

- Octave is **sensitive** to the **case**

```
>>Help
error: 'Help' undefined near line 1 column 1
```

Basic operation…

- Helpful commands: **clc**

Clears the command window

- Another one : **clear** <var_name>

Deletes a variable in workspace

- **clear**

Deletes all variables in workspace !

Basic operation - Directories

- Current Working Directory

pwd

- List directory contents

ls, ls -l

- Change the current working directory to dir

cd dir

- Create a directory named Lab_II

mkdir Lab_II

- Remove the directory named dir

rmdir dir

- Change current directory to another drive

cd E:

Scripts

- Collections of commands to be executed in a sequence
- Can be written and edited in the Octave editor window
- Save as .m files (**.m** is a MatLab extension which is borrowed by Octave)
- Can be saved and loaded from commands or GUI.
- For file to be loaded by name it must be in the working directory.
- Otherwise specify the fully qualified file name.

Example

- Create a script file with following commands,

```
b = 5;
h = 4;
a = 0.5*(b*h) + 3
```

Save it as activity1.m, and execute it.

What is the output?

- Commands:

edit <name-of-the-script.m>

run <name-of-the-script.m>

More about Scripts

- Comment your scripts thoroughly to avoid wasting time later
- Note that scripts are somewhat static, since there is no input and no explicit output
- All variables created and modified in a script exist in the workspace even after it has stopped running

a	double	1x1	13
ans	double	1x1	0
b	double	1x1	5
h	double	1x1	4

Strings

- Make a helloWorld script

Hint: use **disp** to display strings.

- Strings are written between single quotes.

'This is a string'

'My fancy text'

- Double quoted text is acceptable as well

mystring = "This is an extra quoted text"

- **disp(mystring)**

Built-in variables

Don't use these as variable names!

- *i* and *j* can be used to input complex numbers
- **pi** has the value **3.1416…**
- **e** has the value **2.7183…**
- **ans** stores the last unassigned value (like on a calculator)
- **Inf** and **-Inf** represent positive and negative infinity
- **NaN** represents 'Not a Number'

Variables - Scalars

- A variable can be given a value explicitly

>> **a = 10**

- Or as a function of explicit values and existing variables

>> **c = 1.3*45-2*a**

- Octave shows the dimensions of the variable in workspace

>> **cooldude = 13/3;**

>> **size(cooldude)** % returns the dimensions of variable (1 x 1 for scalars)

Variables - Arrays

- Like other programming languages, arrays are an important part of Octave

- Two types of arrays:

1. A Vector/Matrix of numbers (either double or complex)
2. An Array of objects (more advanced data structure)

Variables - Vectors

Row vector is a space or comma separated set of values between square brackets.

row1 = [1 2 3.2 4 6 5.4];

row2 = [1, 2, 4, 7, 4.3, 1.1];

Workspace:

row1	double	1x6	[1, 2, 3.2000, 4, 6, 5.4000]
row2	double	1x6	[1, 2, 4, 7, 4.3000, 1.1000]

Output:

```
>> row1,row2
row1 =

   1.0000    2.0000    3.2000    4.0000    6.0000    5.4000

row2 =

   1.0000    2.0000    4.0000    7.0000    4.3000    1.1000
```

Variables - Vectors

Column vector is a semicolon separated set of values between square brackets

col = [1; 2; 3.2; 4; 6; 5.4];

Workspace:

col	double	6x1	[1; 2; 3.2000; 4; 6; 5.4000]

Output:

```
>> col = [ 1; 2; 3.2; 4; 6; 5.4 ]
col =

   1.0000
   2.0000
   3.2000
   4.0000
   6.0000
   5.4000
```

Variables – Size and Length

To Identify a **Column vector** and a **Row vector** separately different methods can be used

- Checking it in the workspace
- Displaying it in the command window
- Using **size** and **length** functions

```
>>
>> size(col)
ans =

   6   1

>> size(row1)
ans =

   1   6

>> length(row2)
ans =  6
>> length(col)
ans =  6
>>
```

Variables – Matrices

- A matrix is a two dimensional array.
- They can be defined in the same way as vectors by combining the rows and columns.

>> **matrix = [1 2 ; 3 4]** **% 1 and 2 are first row 3 and 4 are second row**

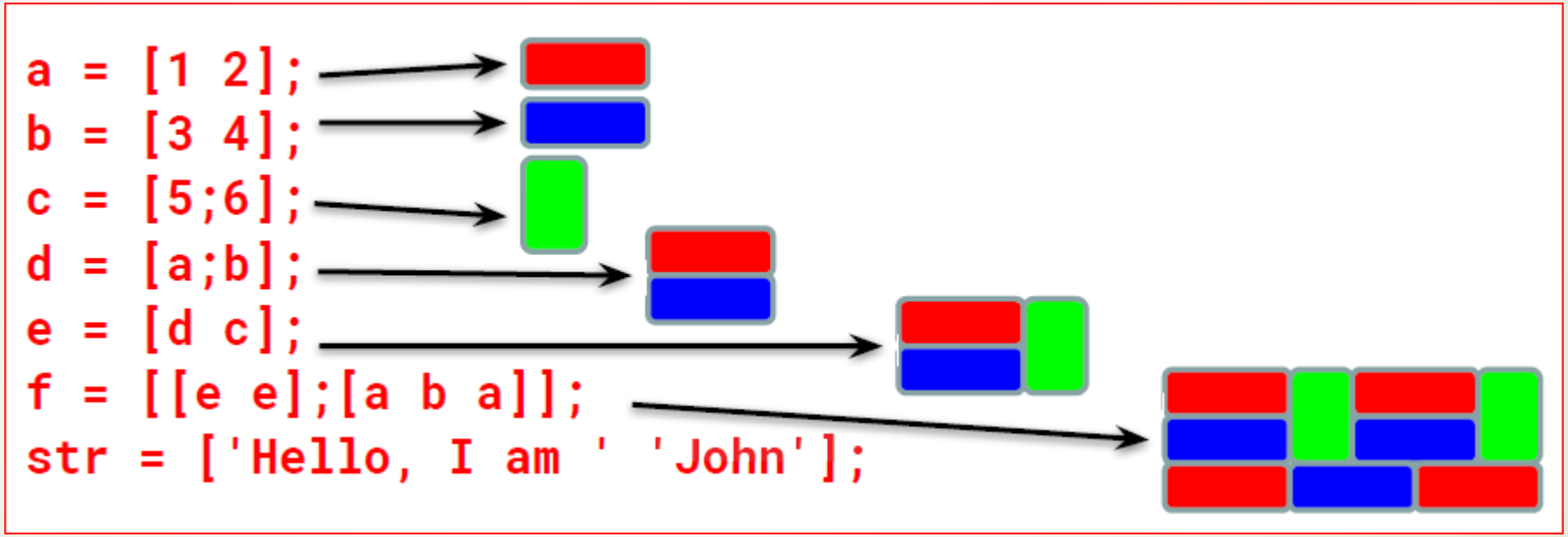
```
>> matrix = [ 1 2 ; 3 4]
matrix =

   1   2
   3   4

>>
```

Variables – Matrices

- Combining smaller matrices or vectors we can generate larger matrices.



Operators: Element-wise

- For element-wise operations use the **.<op>** notation. E.g.: **.*,./,.^**
- a = [1 2 3]; b = [4; 5; 6;] => (1x3) and (3x1)
- a*b

ans = 32

- a .* b

ans =

4	**8**	**12**
5	**10**	**15**
6	**12**	**18**

Operators: Element-wise…

- a’ = [1 2 3]’; b’ = [4; 5; 6;]’ => (3x1) and (1x3)
- a' * b‘ % a' .* b‘ gives the same answer

ans =

4	**5**	**6**
8	**10**	**12**
12	**15**	**18**

Automatic Initialization

Initialize a vector of **ones**, **zeros**, or **rand**om numbers

- » `o=ones(1,10)`
  - row vector with 10 elements, all 1
- » `z=zeros(23,1)`
  - column vector with 23 elements, all 0
- » `r=rand(1,45)`
  - row vector with 45 elements (uniform [0,1])
- » `n=nan(1,69)`
  - row vector of NaNs (useful for representing uninitialized variables)

The general function call is:

```
var=zeros(M,N);
```

Number of rows Number of columns

Built-in Functions

Octave has an enormous library of built-in functions

- Call using parentheses – passing parameters to the function
- natural and base 10 logarithms: log( ), log10( )
- **Try =>** $$\frac{\log_{10} 100}{\log_{10} 10}$$
- Square Root: sqrt( )
- **Try=>** $$\sqrt{3^2 + 4^2}$$

Built-in Functions

Octave trigonometric functions

- sin(pi/2)
- cos(pi/3)
- tan(pi/4)
- acos (<value>)
- asin (<value>)
- atan(<value>)

Built-in Functions

Octave trigonometric functions

- sind(in_degrees)
- cosd(in_degrees)
- tand(in_degrees)

Built-in Functions

- round(x) – Rounds the fractional value

  round(3.5), round(2.3)

- floor(x) – Return the largest integer not greater than x

  floor(3.3), floor(5.9)

- ceil(x) - Return the lowest integer not less than x

  ceil(5.7), ceil(5.21)

Built-in Functions

- Complex numbers:

Try square root of negative 1 : sqrt(-1)

- Defining complex numbers:

  a = 1 + 2*i

  b = 0 + 3*i

- Exercise:

  a + b = ?

  a – b = ?

Built-in Functions

Functions on complex numbers:

- **angle(1 + i)**
- **abs(1 - i)**

Exponential function:

- **exp(value)**

Thank You!

Download Octave - https://www.gnu.org/software/octave/