Introduction to Octave/Matlab

Deployment of Telecommunication Infrastructures

What is Octave?

- Software for **numerical computations** and **graphics**
- Particularly designed for **matrix computations**
  - Solving equations, computing eigenvalues, etc
  - Many engineering problems can be expressed as matrices and vectors and solved using matrix operations
- Has it's own programming language
  - Works like a very powerful, programmable, graphical calculator
- Is an **open source** software
  - It is syntax compatible with **MATLAB** (a proprietary software with similar functionalities)

Why and when to use Octave?

- Why?
  - Solutions to mathematical problems take time to program in a normal highlevel language like C++ or java
  - Octave has many built-in mathematical functions

- When?
  - When you want to solve mathematical problems **numerically**
  - Is **NOT** designed to find exact **symbolic** solutions to problems (unlike Mathematica or Maple)

The basics

- **Starting octave**
  - type **`octave`** on the command prompt. You are in octave environment when you see:
    - `octave:1>`
- **Input conventions**
  - Commands can be typed at the prompt or read from a script.
  - Scripts are plain text files with file suffix **.m**.
  - To run a script, type the script name without the suffix
  - “;” separates commands in a line and suppresses the output (use “,” to separate without suppressing the output)
  - Comments are preceded by **%**.
  - Octave is case-sensitive.

Basic operations

- Getting help
  - **`help`** lists all built-in functions and internal variables.
  - **`help name`** explains the variable or function “name”.
- Octave as a calculator
  - Just type mathematical commands at the prompt
    - use `+, -, *, /, ^, (),` etc. with normal precedence.
    - `octave:01> 5 + 3^2`
  - It also has mathematical built-in functions
    - examples: **`sin, cos, sqrt, log, log10, exp, abs, ...`**
    - **Practice:** Calculate $10\,\sin(90°) + \ln(e)$
    - **Note 1:** trigonometric functions work in **radians**
    - **Note 2:** **pi** and **e** (and **i**, **j**) are predefined variables, the **ans** variable is used to hold the result of the last operation

Dealing with Variables

- No need to declare the variable or its type
  - variables are either **floating point** numbers or **strings**
  - **`var=expression`** assigns the result of expression to var
    - **`octave:##> 6*2`**
    - **`octave:##> a = ans/3, b = "example string"`**
- To see the value of a variable just type its name and press return
- Some useful functions:
  - **`who`** Lists all functions and variables you created or used
  - **`clear`** removes all variables from the workspace
    - Recommendation: Use this at the beginning of all scripts
  - **`clear`** ***`VariableName`*** removes specified variable

Generating Vectors and Matrices

Vectors:

- Try the following:

```
aa = [0 1 2 3]
bb = 1:4
cc = 0.3:-0.1:0
dd = [1;2;3;4]
ee = cc'
ff = aa(2:3)
gg = [aa,cc,5]
hh = linspace(.1,1,10)
ii = logspace(2,6,5)
length(ii)
```

Matrices:

- Try the following

```
AA = [1 2; 3 4]
AA(2,1)
AA(1,:)
AA(:,2)
BB = [aa(1:2); bb(3:4)]
CC = ones(3,2)
DD = zeros(2,3)
AA = [AA,[0;0],AA']
EE = eye(size(AA))
FF = rand(2,2)
```

Notes on Matrices and Vectors

- Vectors are just a special case of matrices
- “,” or **space** add elements to the same row, while “**;**” starts a new row (when generating a matrix).
- “**:**” specifies a range of numbers “***min*:*step*:*max***”
  - step can be negative,
  - step is 1 if not specified
- The **indexes** of matrices and vectors should be **> 1**
- You can **delete or modify rows or columns** of a matrix by using a combination of what you learned:
  - `AA(1,:) = zeros(size(AA(1,:)))`
  - `AA(1,:) = []`

Matrix Operations

- Basic manipulations ⇨ Some useful functions:
  - **`find`**, **`sum`**, **`min`**, **`max`**, **`diag`**, **`size`**, **`rows`**, **`columns`**
  - see the handout for details (and try in octave to understand)
- Basic arithmetics
  - **+**, **-**, and * denote matrix addition, subtraction, and multiplication.
    - note that dimensions should match accordingly
  - **A'** transposes and conjugates **A**
- Element-wise operations
  - a "." (dot) before an operation, makes it work element-wise
    - try **`A = [1 2 ; 3 4], B = A^2, C = A.^2`**
  - **.*** , **./** , **.^** are element-wise multiplication, division, and power
  - **.+** and **.-** are the same as **+** and **-**
  - **A.'** transposes **A** but does not conjugate

Plotting graphs

- **`plot(X1, Y1, LineSpec, X2, Y2, LineSpec,...)`**
  - plots Y1 versus X1 and Y2 vs. X2 on the same plot, with the attributes specified in LineSpec (a string)
  - Example:

```
x  = linspace(0,2*pi,100);
y = sin(x);
plot(x,y,'ro')
```

  - Linespecs
    - **Colors:** w, **m**, **c**, **r**, **g**, **b**, **y**, **k**
    - **Line markers: .** , **o**, **x**, **+**, *, **s**, **d**, **v**, **^**, **<**, **>**, **p**, **h**
    - **Line format: -** , **:** , **-.** , **--**
    - LineSpecs of different types can be used together
    - LineSpecs should always be in string form (between quotes)
  - Other useful plotting commands
    - **`figure, figure(#), axis, hold on/off, grid on/off, title, xlabel, ylabel, legend`**

Control statements

- **if**

```
if expression
    statements
elseif expression
    statements
else
    statements
end
```

- **for**

```
for variable = vector
    statements
end
```

- **while**

```
while expression
    statements
end
```

- **switch**

```
switch x
   case x1
      statements
   case x2
      statements
   otherwise
      statements
end
```

Note: *expression* is a logical expression. Use the ***Comparison and Boolean Operators*** in the handout for making logical expressions.

Scripts

- **To create**
  - make a text file containing the *sequence of commands* which lead to your desired result
  - save the file with the extension **.m** (called M-files)
- **To run**
  - just type the filename without extension on the command line and press return
    - This has the same effect as entering the same *sequence of commands* line by line into octave
- **Exercise:**
  - Write a script that calculates and plots on the same figure exactly two periods (0 to 4π) of the following sine waves:

$$\sin(t) \qquad \sin(t+\pi)$$
$$\sin(t+\pi/2) \qquad \sin(t+3\pi/2)$$

Functions

- **To create:** make a file containing the following:

```
function [out1,out2,...] = function_name (in1, in2,...)
    function-body
end
```

  - Note: the **`end`** is optional (except for nested functions)
- **To run:** type the following at octave command line:
  - `octave:##> [o1,o2,...] = function_name[i1,i2,...]`
- **Exercise:**
  - Write a simple function that calculates the **angle between two vectors in degrees**. Note that for vectors v and u we have the following relationship between their dot product and the angle between them

$$v \cdot u = \mid v \mid \mid u \mid \cos(\measuredangle(v,u))$$