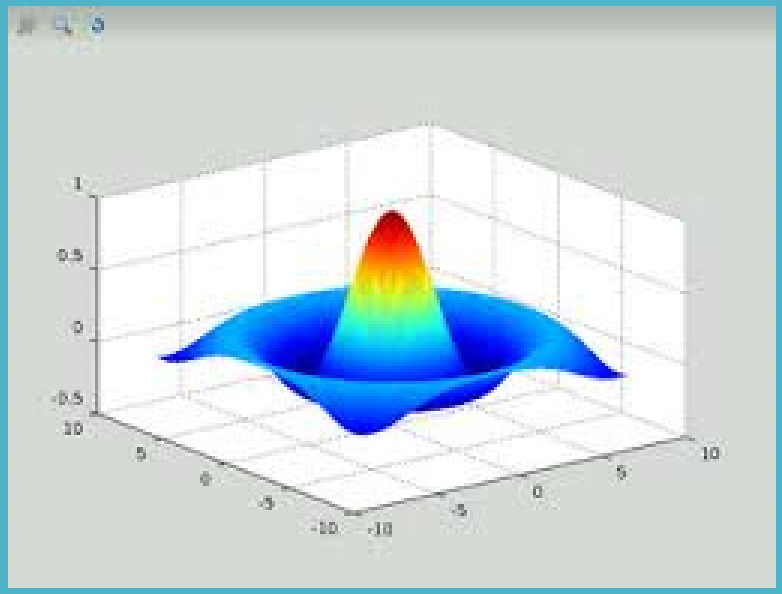
A quick introduction to Octave



23rd August 2017

Dr. (Mrs.) Anne Mindika Premachandra, PhD(ANU), BSc(Hons)(CS) (Col)

amp@ucsc.cmb.ac.lk, anne.mindika@gmail.com

0776315200

Outline

- GNU Octave
- Scripts
- Variables (Scalars, Vectors and Matrices)
- Manipulating variables
- Basic plotting
- Functions
- Flow control
- Help/Docs
- Neural Networks with Octave

Outline

- GNU Octave
- Scripts
- Variables (Scalars, Vectors and Matrices)
- Manipulating variables
- Basic plotting
- Functions
- Flow control
- Help/Docs
- Neural Networks with Octave

What is GNU Octave?

- GNU Octave is a high-level interpreted language, primarily intended for numerical computations.

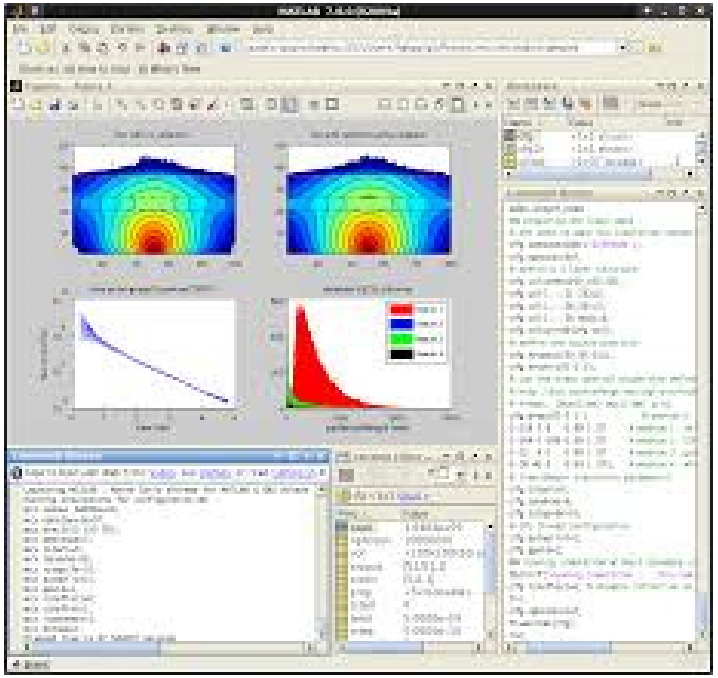
- In addition it is a programming language
  - Octave is an interpreted language, like Java
  - Commands executed line by line

- Octave includes
  - a Graphical User Interface (GUI) and
  - a Command Line Interface (CLI)

  - Octave is written in C++ using the C++ standard library.

- Get installers and sources from http://octave.org/download

Octave is great for:



- Data Analytics
- Machine Learning
- Statistical Computations
- Numerical Computations
- Advanced Graphing for data visualization and manipulation
- …

Octave and Matlab

- Matlab:
  - a commercial software
  - Name derived from MATrix LABoratory
  - Large toolbox of numeric/image library functions

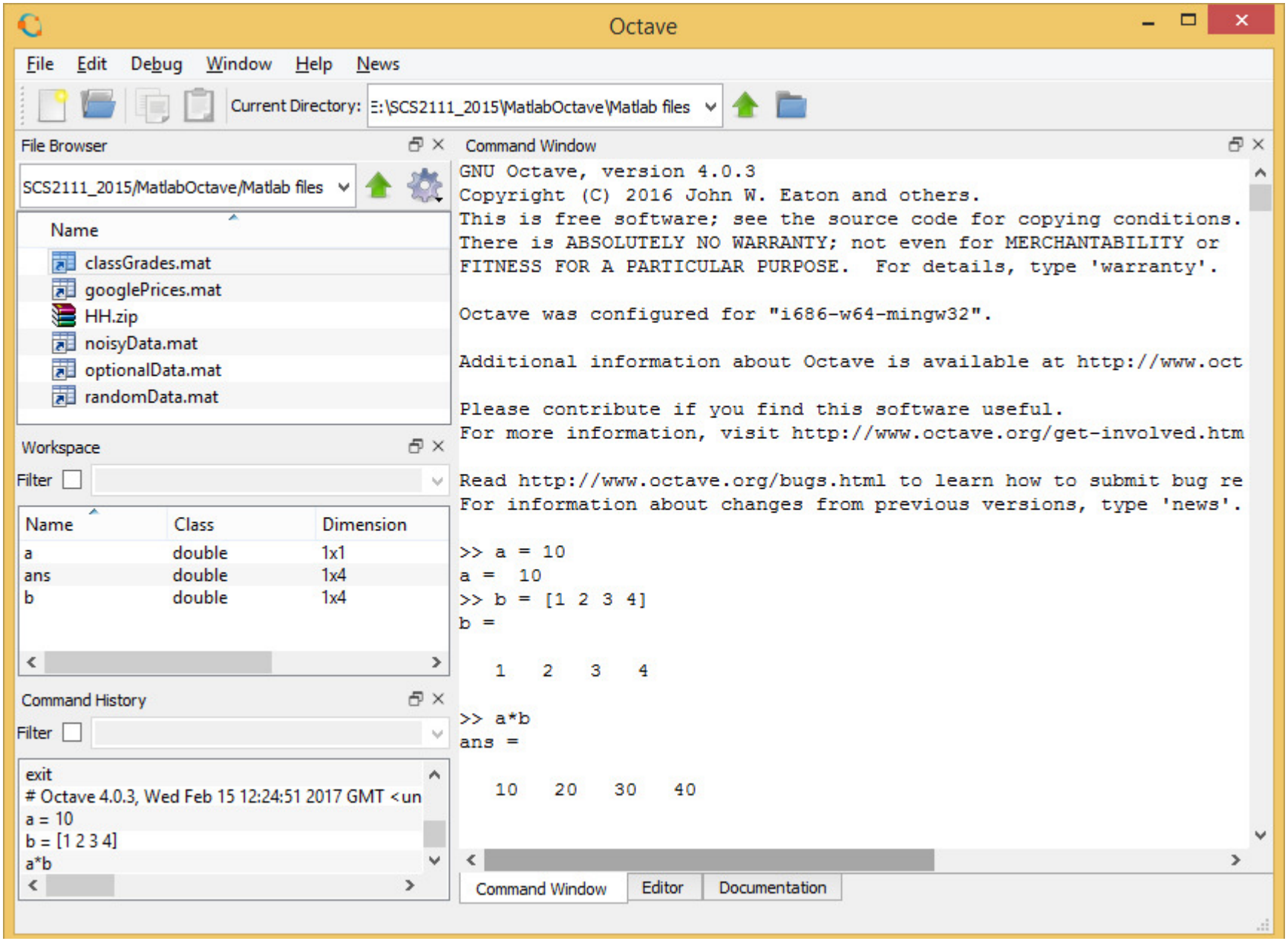
- Octave is one of the major free alternatives to Matlab
  - Octave is quite similar to Matlab
  - so that most programs are easily portable.

Other free alternatives to Matlab

- Freemat
- Sage
- Scilab
- R

[http://www.researchgate.net/post/Which_is_the_best_alternative_to_Matlab]

The GUI


Octave
File Edit Debug Window Help News
Current Directory: E:\SCS2111_2015\MatlabOctave\Matlab files
File Browser
SCS2111_2015/MatlabOctave/Matlab files
Name
classGrades.mat
googlePrices.mat
HH.zip
noisyData.mat
optionalData.mat
randomData.mat
Workspace
Filter
Name Class Dimension
a double 1x1
ans double 1x4
b double 1x4
Command History
Filter
exit
# Octave 4.0.3, Wed Feb 15 12:24:51 2017 GMT <un
a = 10
b = [1 2 3 4]
a*b
Command Window
GNU Octave, version 4.0.3
Copyright (C) 2016 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. For details, type 'warranty'.
Octave was configured for "i686-w64-mingw32".
Additional information about Octave is available at http://www.oct
Please contribute if you find this software useful.
For more information, visit http://www.octave.org/get-involved.htm
Read http://www.octave.org/bugs.html to learn how to submit bug re
For information about changes from previous versions, type 'news'.
>> a = 10
a = 10
>> b = [1 2 3 4]
b =
1 2 3 4
>> a*b
ans =
10 20 30 40
Command Window
Editor
Documentation

Outline

- GNU Octave
- Scripts
- Variables (Scalars, Vectors and Matrices)
- Manipulating variables
- Basic plotting
- Functions
- Flow control
- Help/Docs
- Neural Networks with Octave

Scripts : Overview

- Scripts are
  - collection of commands executed in sequence
  - written in the Octave editor
  - saved with an .m extension (The .m extension is used because MATLAB calls its script files M-files and Octave is based on MATLAB.)

- To create an Octave file from command-line

  ```
  >> edit helloWorld.m
  ```

  or click the new file icon

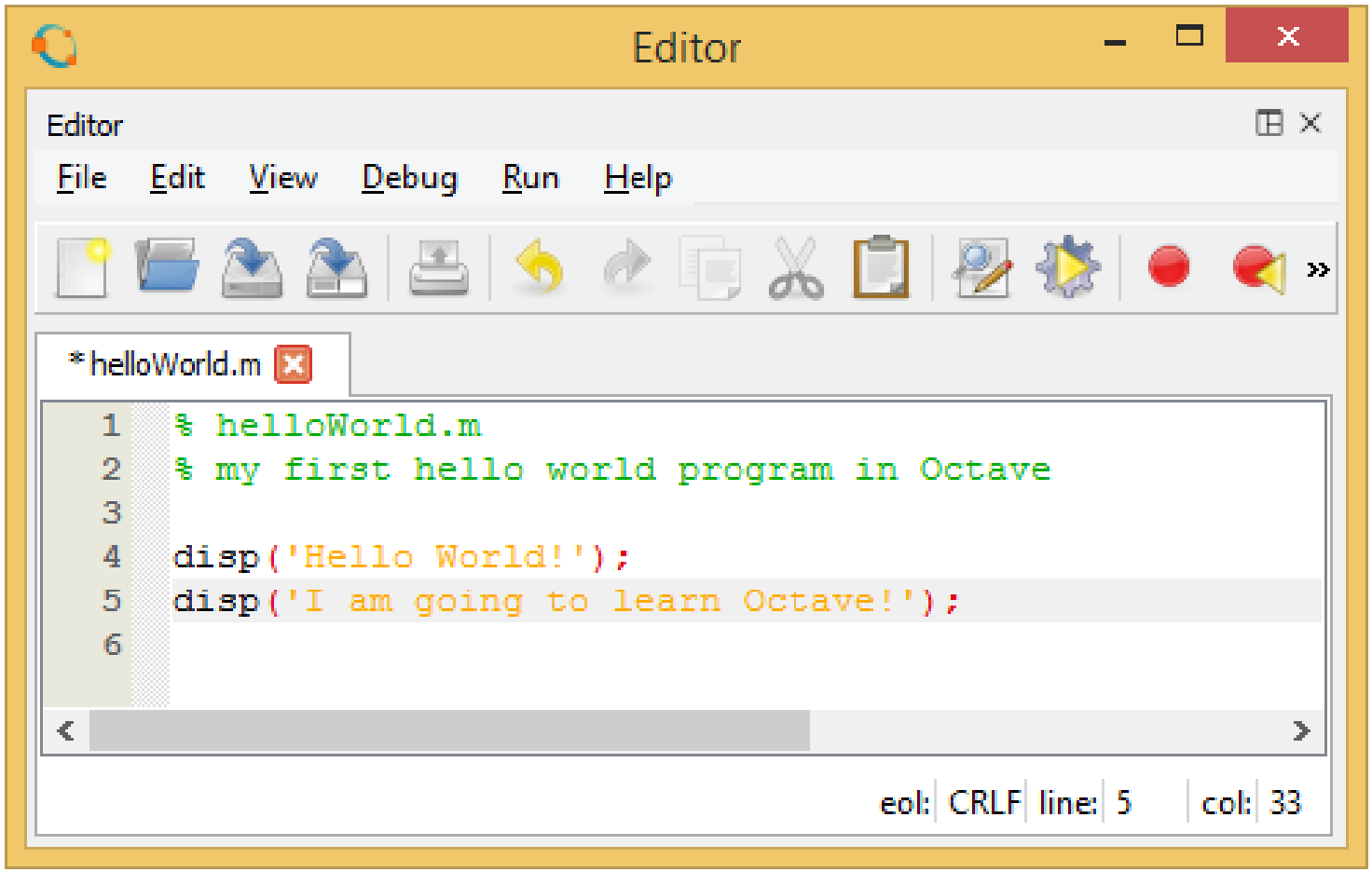
- To run the script:

  ```
  >> run helloWorld.m
  ```

Exercise: Scripts

Make a helloWorld script

- Hint: use **disp** to display strings. Strings are written between single quotes, like **'This is a string'**



Scripts: Some notes

- COMMENT!
  - Anything following a % is seen as a comment
  - The first contiguous comment becomes the script's help file
  - Comment 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

Outline

- GNU Octave
- Scripts
- Variables (Scalars, Vectors and Matrices)
- Manipulating variables
- Basic plotting
- Functions
- Flow control
- Help/Docs
- Neural Networks with Octave

Variable Types

- Octave is a weakly typed language
  - ➢ No need to initialize variables!

- Octave supports various types, the most often used are
  - >> 3.84 64-bit double (default)
  - >> a' 16-bit char

- Most variables you'll deal with will be **vectors** or **matrices** of doubles or chars

- **Other types are also supported:** complex, symbolic, 16-bit and 8 bit integers, etc.

Naming Variables

- To create a variable, simply assign a value to a name:

```
>> var1=3.14
>> myString='hello world'
```

- Variable names
  - first character must be a LETTER
  - after that, any combination of letters, numbers and _
  - CASE SENSITIVE! (var1 is different from Var1)

Built-in Variables

- Built-in variables. Don't use these names!

- i and j can be used to indicate complex numbers
- pi has the value 3.1415926…
- ans stores the last unassigned value (like on a calculator)
- Inf and -Inf are positive and negative infinity
- NaN represents 'Not a Number

Scalars

- A variable can be given a value explicitly

  >> a = 10 shows up in workspace!

- Or as a function of explicit values and existing variables

  >> c = 1.3*45-2*a

- To suppress output, end the line with a semicolon

  >> cooldude = 13/3;

Arrays

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

- Two types of arrays:

  (1) matrix of numbers (either double or complex)

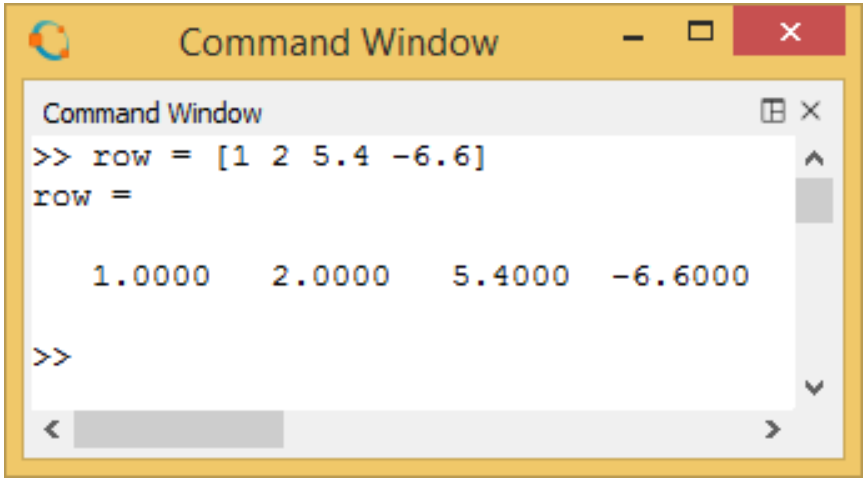
  (2) cell array of objects (more advanced data structure)

Row Vectors

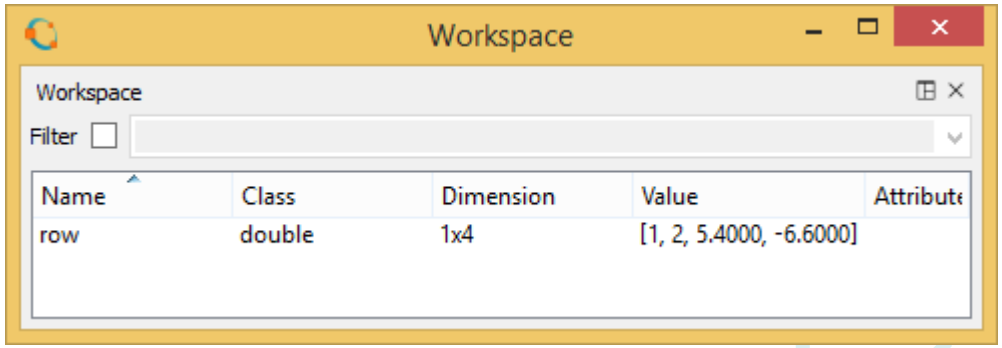
- Row vector: comma or space separated values between brackets

```
>> row = [1 2 5.4 -6.6]
>> row = [1, 2, 5.4, -6.6];
```

Command Window:



Workspace:

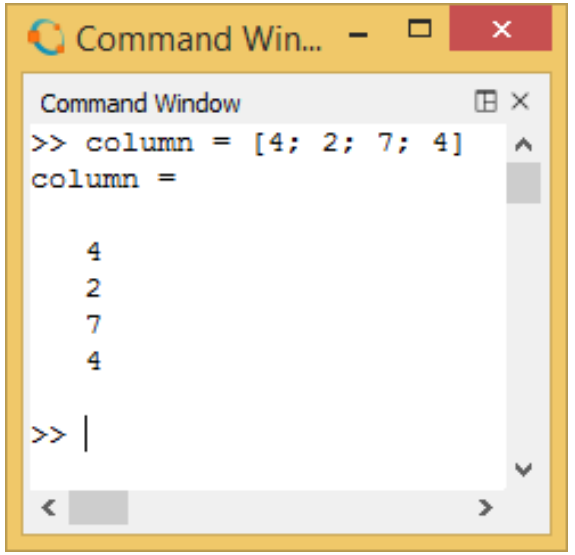


Column Vectors

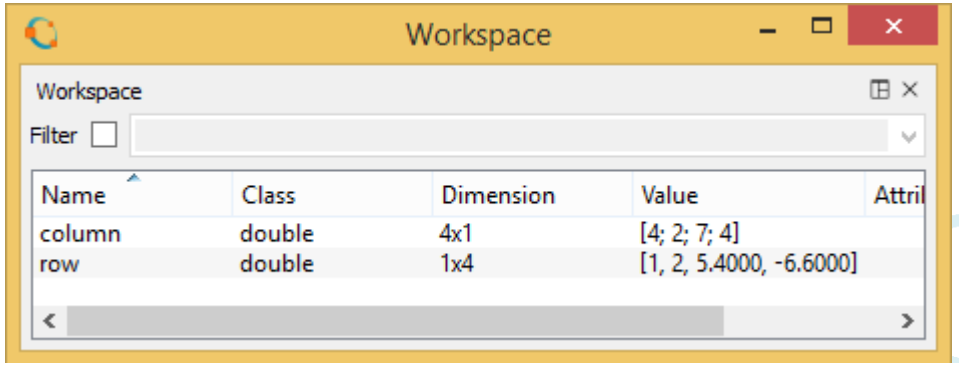
- Column vector: semicolon separated values between brackets

  >> column = [4;2;7;4]

  Command Window:



  Workspace:



Matrices

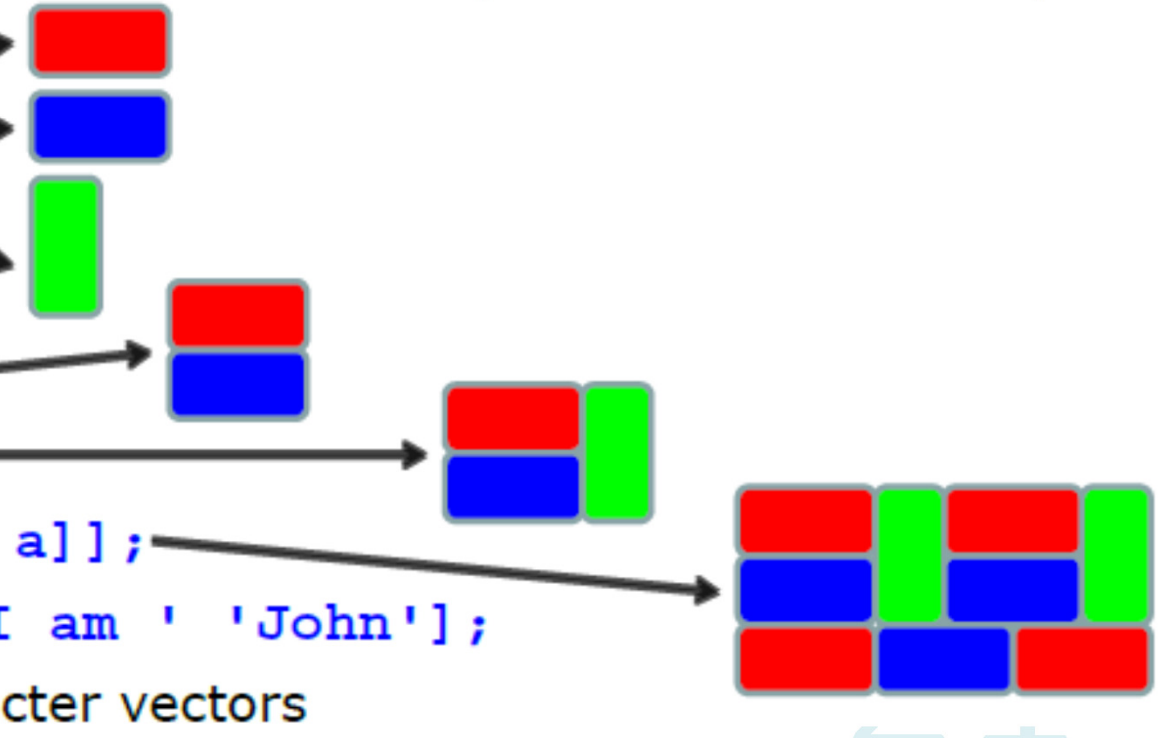
- Make matrices like vectors
- Element by element

```
» a= [1 2;3 4];
```

$$a = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

- By concatenating vectors or matrices (dimension matters)

```
» a = [1 2];
» b = [3 4];
» c = [5;6];
» d = [a;b];
» e = [d c];
» f = [[e e];[a b a]];
» str = ['Hello, I am ' 'John'];
```



  - Strings are character vectors

Save/clear/load

- Use **save** to save variables to a file
  - » `save myFile a b`
    - ➢ saves variables a and b to the file myfile.mat
    - ➢ myfile.mat file is saved in the current directory

- Use **clear** to remove variables from environment
  - » `clear a b`
    - ➢ look at workspace, the variables a and b are gone

- Use **load** to load variable bindings into the environment
  - » `load myFile`
    - ➢ look at workspace, the variables a and b are back

- Can do the same for entire environment
  - » `save myenv; clear all; load myenv;`

Exercise : Variables

Get and save the current date and time

- Create a variable `start` using the function `clock`
- What is the size of `start`? Is it a row or column?
- What does `start` contain? See `help clock`
- Convert the vector `start` to a string. Use the function `datestr` and name the new variable `startString`
- Save `start` and `startString` into a mat file named `startTime`

```
» help clock
» start=clock;
» size(start)
» help datestr
» startString=datestr(start);
» save startTime start startString
```

Exercise : Variables

Read in and display the current date and time

- In helloWorld.m, read in the variables you just saved using load
- Display the following text:

  I started learning Octave on *start date and time*

- Hint: use the disp command again, and remember that strings are just vectors of characters so you can join two strings by making a row vector with the two strings as sub-vectors.

```
>> load startTime
>> disp(['I started learning Octave on '  startString]);
```

Outline

- GNU Octave
- Scripts
- Variables (Scalars, Vectors and Matrices)
- Manipulating variables
- Basic plotting
- Functions
- Flow control
- Help/Docs
- Neural Networks with Octave

Basic scalar operations

- Arithmetic operations (+,-,*,/)
  - » 7/45
  - » (1+i)*(2+i)
  - » 1 / 0
  - » 0 / 0
- Exponentiation (^)
  - » 4^2
  - » (3+4*j)^2
- Complicated expressions, use parentheses
  - » ((2+3)*3)^0.1
- Multiplication is NOT implicit given parentheses
  - » 3(1+0.7) gives an error
- To clear command window
  - » clc

Built-in functions

- Octave has an **enormous** library of built-in functions
- Call using parentheses –passing parameter to function

```
» sqrt(2)
» log(2), log10(0.23)
» cos(1.2), atan(-.8)
» exp(2+4*i)
» round(1.4), floor(3.3), ceil(4.23)
» angle(i); abs(1+i);
```

Transpose

- The transpose operators turns a column vector into a row vector and vice versa

```
» a = [1 2 3 4+i]
» transpose(a)
» a'
» a.'
```

- The ' gives the Hermitian-transpose, i.e. transposes and conjugates all complex numbers

- For vectors of real numbers .' and ' give same result

Addition and Subtraction

- Addition and subtraction are element-wise; sizes must match (unless one is a scalar):

$$\begin{array}{r} [12 \quad 3 \quad 32 \quad -11] \\ +[2 \quad 11 \quad -30 \quad 32] \\ \hline =[14 \quad 14 \quad 2 \quad 21] \end{array}$$

$$\begin{bmatrix} 12 \\ 1 \\ -10 \\ 0 \end{bmatrix} - \begin{bmatrix} 3 \\ -1 \\ 13 \\ 33 \end{bmatrix} = \begin{bmatrix} 9 \\ 2 \\ -23 \\ -33 \end{bmatrix}$$

- The following would give an error

```
» c = row + column
```

- Use the transpose to make sizes compatible

```
» c = row' + column
» c = row + column'
```

- Can sum up or multiply elements of vector

```
» s=sum(row);
» p=prod(row);
```

Element-wise functions

- All the functions that work on scalars also work on vectors

```
» t = [1 2 3];
» f = exp(t);
```

  - is the same as

```
» f = [exp(1) exp(2) exp(3)];
```

- If in doubt, check a function's help file to see if it handles vectors elementwise

- Operators (* / ^) have two modes of operation
  - element-wise
  - standard

Operators: Element-wise

- To do element-wise operations, use the dot: `.` (`.*`, `./`, `.^`). BOTH dimensions must match (unless one is scalar)!
  - `» a=[1 2 3];b=[4;2;1];`
  - `» a.*b, a./b, a.^b → all errors`
  - `» a.*b', a./b', a.^(b') → all valid`

$$\begin{bmatrix}1 & 2 & 3\end{bmatrix}.*\begin{bmatrix}4\\2\\1\end{bmatrix} = ERROR$$

$$\begin{bmatrix}1\\2\\3\end{bmatrix}.*\begin{bmatrix}4\\2\\1\end{bmatrix}=\begin{bmatrix}4\\4\\3\end{bmatrix}$$

$$3\times1.*3\times1=3\times1$$

$$\begin{bmatrix}1 & 1 & 1\\2 & 2 & 2\\3 & 3 & 3\end{bmatrix}.*\begin{bmatrix}1 & 2 & 3\\1 & 2 & 3\\1 & 2 & 3\end{bmatrix}=\begin{bmatrix}1 & 2 & 3\\2 & 4 & 6\\3 & 6 & 9\end{bmatrix}$$

$$3\times3.*3\times3=3\times3$$

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}.^2=\begin{bmatrix}1^2 & 2^2\\3^2 & 4^2\end{bmatrix}$$

Can be any dimension

Operators: Standard

- Multiplication can be done in a standard way or element-wise
- Standard multiplication (*) is either a dot-product or an outer-product
  - Remember from linear algebra: inner dimensions must MATCH!!
- Standard exponentiation (^) can only be done on square matrices or scalars
- Left and right division (/ \) is same as multiplying by inverse
  - Our recommendation: just multiply by inverse (more on this later)

$$\begin{bmatrix}1 & 2 & 3\end{bmatrix} * \begin{bmatrix}4\\2\\1\end{bmatrix} = 11$$

$$1\times 3 * 3\times 1 = 1\times 1$$

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} \wedge 2 = \begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} * \begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}$$

Must be square to do powers

$$\begin{bmatrix}1 & 1 & 1\\2 & 2 & 2\\3 & 3 & 3\end{bmatrix} * \begin{bmatrix}1 & 2 & 3\\1 & 2 & 3\\1 & 2 & 3\end{bmatrix} = \begin{bmatrix}3 & 6 & 9\\6 & 12 & 18\\9 & 18 & 27\end{bmatrix}$$

$$3\times 3 * 3\times 3 = 3\times 3$$

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

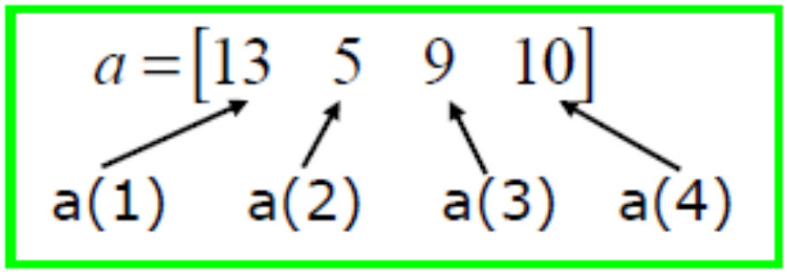
Automatic Initialization

- To initialize a linear vector of values use **linspace**
  - » `a=linspace(0,10,5)`
    - starts at 0, ends at 10 (inclusive), 5 values
- Can also use colon operator (**:**)
  - » `b=0:2:10`
    - starts at 0, increments by 2, and ends at or before 10
    - increment can be decimal or negative
  - » `c=1:5`
    - if increment isn't specified, default is 1
- To initialize logarithmically spaced values use **logspace**
    - similar to **linspace**, but see **help**

Vector Indexing

- Octave indexing starts with 1, not 0

- a(n) returns the n^{th} element

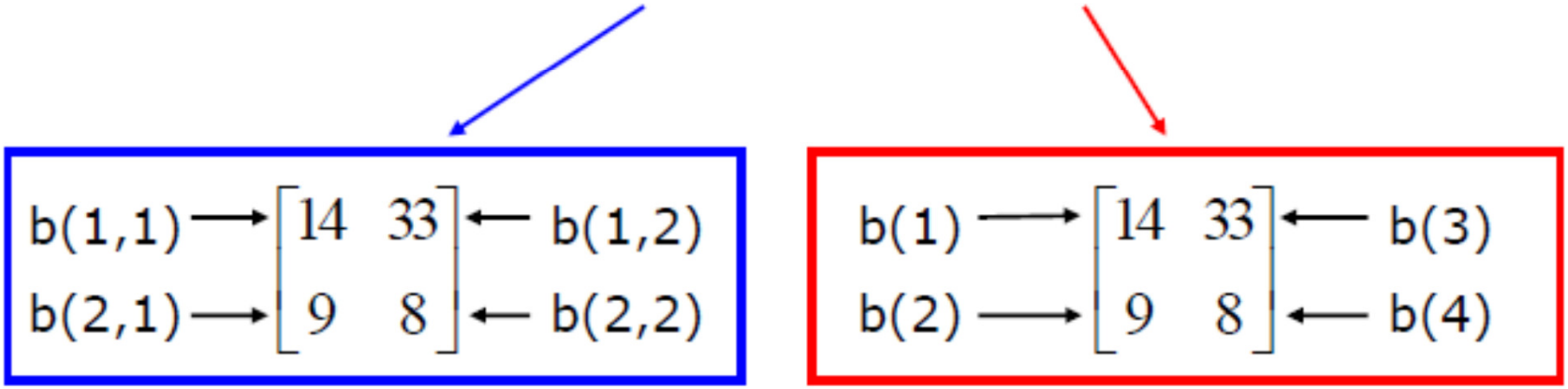


- The index argument can be a vector. In this case, each element is looked up individually, and returned as a vector of the same size as the index vector.

```
» x=[12 13 5 8];
» a=x(2:3);      ⟶  a=[13 5];
» b=x(1:end-1);  ⟶  b=[12 13 5];
```

Matrix Indexing

- Matrices can be indexed in two ways
  - using **subscripts** (row and column)
  - using linear **indices** (as if matrix is a vector)
- Matrix indexing: subscripts or linear indices



- Picking submatrices

```
» A = rand(5) % shorthand for 5x5 matrix
» A(1:3,1:2) % specify contiguous submatrix
» A([1 5 3], [1 4]) % specify rows and columns
```

Advanced Indexing 1

- To select rows or columns of a matrix, use the **:**

$$c = \begin{bmatrix} 12 & 5 \\ -2 & 13 \end{bmatrix}$$

```
» d=c(1,:);  ⟶  d=[12 5];
» e=c(:,2);  ⟶  e=[5;13];
» c(2,:)=[3 6];  %replaces second row of c
```

Advanced Indexing 2

```
» vec = [5 3 1 9 7]
```

- To get the minimum value and its index:
  - `» [minVal,minInd] = min(vec);`
    - `max` works the same way
- To find any the indices of specific values or ranges
  - `» ind = find(vec == 9);`
  - `» ind = find(vec > 2 & vec < 6);`
    - `find` expressions can be very complex,

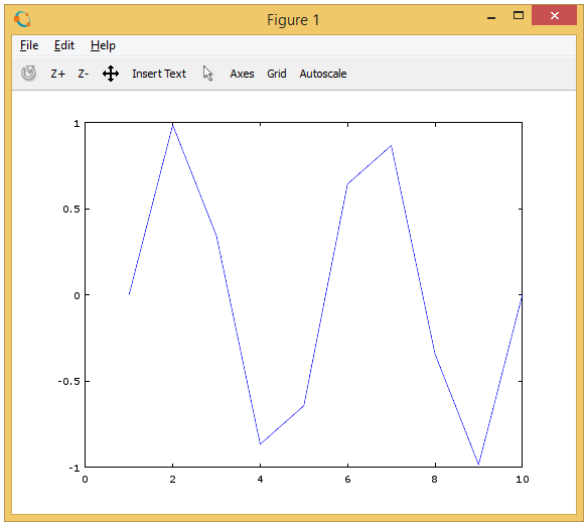
Outline

- GNU Octave
- Scripts
- Variables (Scalars, Vectors and Matrices)
- Manipulating variables
- Basic plotting
- Functions
- Flow control
- Help/Docs
- Neural Networks with Octave

Plotting

- Example

```
» x=linspace(0,4*pi,10);
» y=sin(x);
```

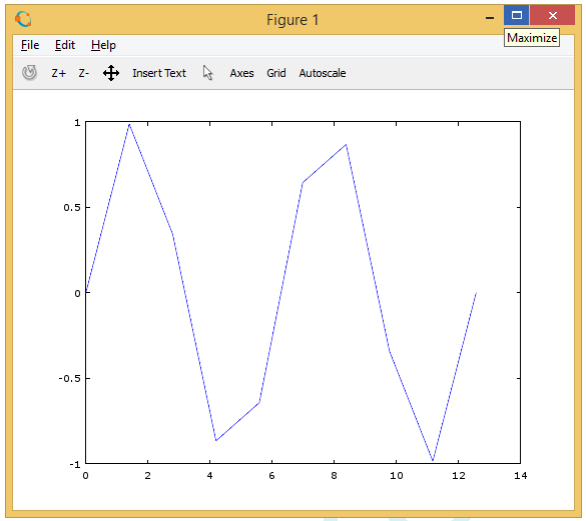


- Plot values against their index

```
» plot(y);
```

- Usually we want to plot y versus x

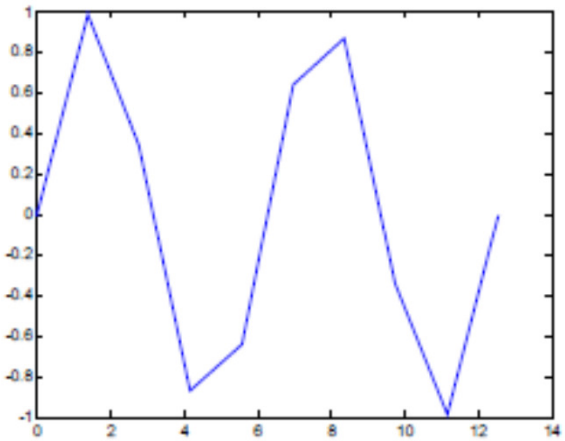
```
» plot(x,y);
```



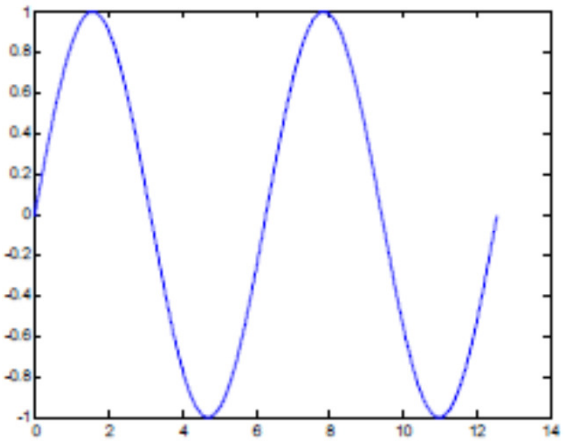
What does plot do?

- **plot** generates dots at each (x,y) pair and then connects the dots with a line
- To make plot of a function look smoother, evaluate at more points
  - » `x=linspace(0,4*pi,1000);`
  - » `plot(x,sin(x));`
- x and y vectors must be same size or else you'll get an error
  - » `plot([1 2], [1 2 3])`
    - ➢ error!!

10 x values:



1000 x values:



Plot axis/title naming

- There are commands to "annotate" a plot to put on axis labels, titles, and legends.

- For example:

➢ To put a label on the axes we would use:

```
>> xlabel ('X-axis label')
>> ylabel ('Y-axis label')
```

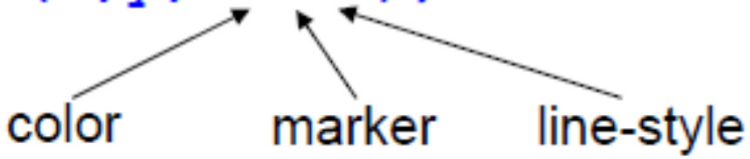
➢ To put a title on the plot, we would use:

```
>> title ('Title of my plot')
```

Plot options

- Can change the line color, marker style, and line style by adding a string argument

```
» plot(x,y,'k.-');
```



- Can plot without connecting the dots by omitting line style argument

```
» plot(x,y,'.')
```

- Look at **help plot** for a full list of colors, markers, and linestyles

Outline

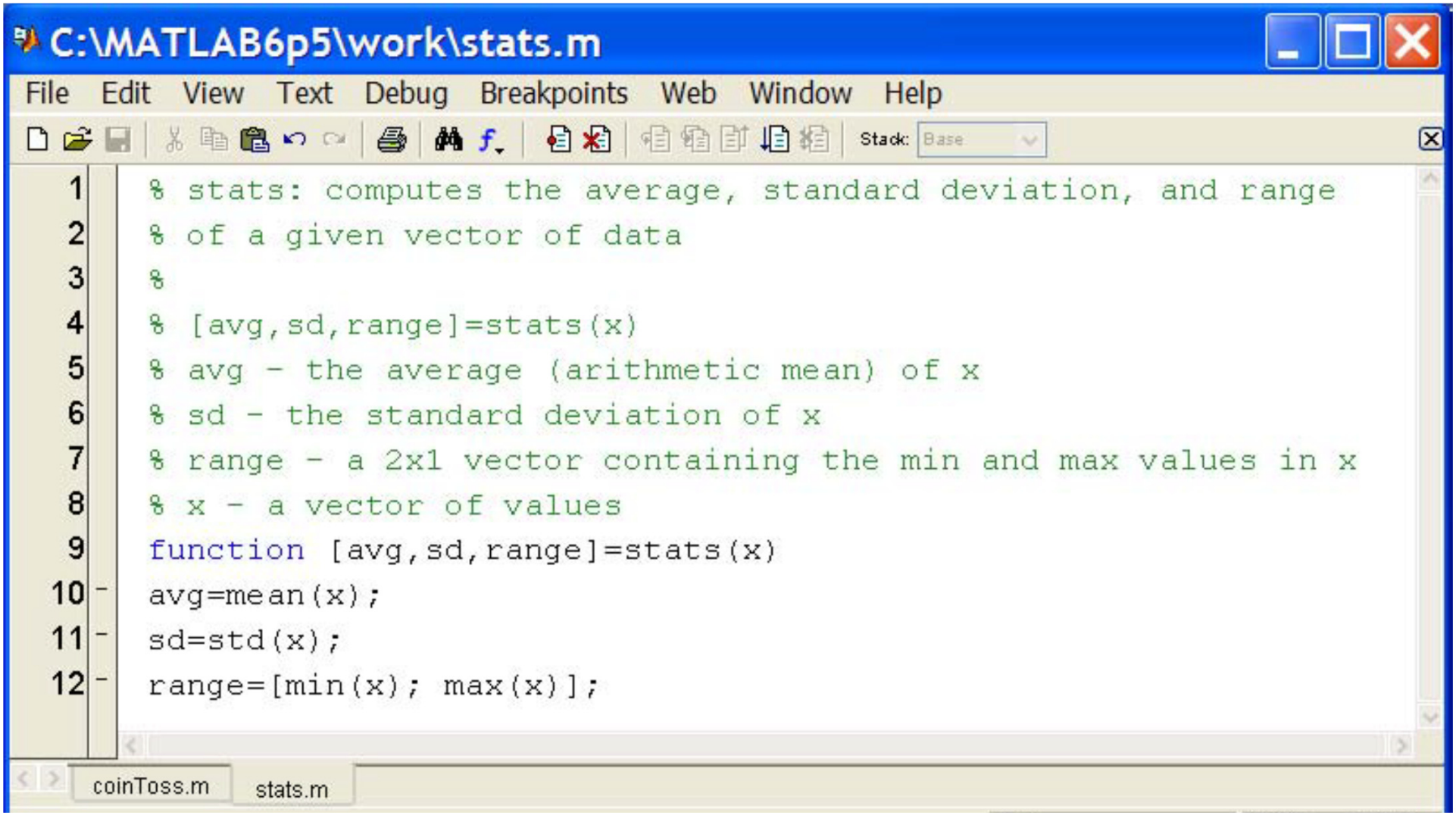
- GNU Octave
- Scripts
- Variables (Scalars, Vectors and Matrices)
- Manipulating variables
- Basic plotting
- Functions
- Flow control
- Help/Docs
- Neural Networks with Octave

User-defined functions

- Functions look exactly like scripts, but for ONE difference:

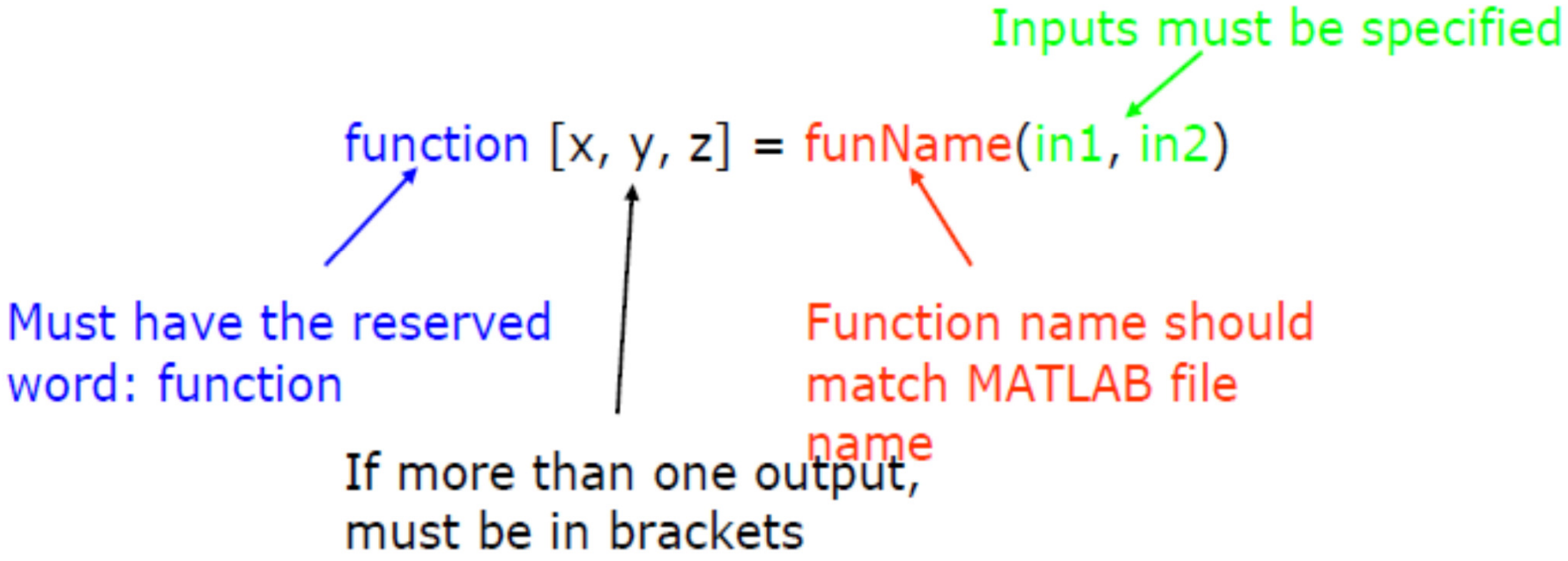
- Functions must have a function declaration

User-defined functions



User-defined functions

- Some comments about the function declaration



- No need for return: 'returns' the variables whose names match those in the function declaration
- Variable scope: Any variables created within the function but not returned disappear after the function stops running

Functions: overloading

- Octave functions are generally overloaded
  - Can take a variable number of inputs
  - Can return a variable number of outputs

- What would the following commands return?

```
» a=zeros(2,4,8);
» D=size(a)
» [m,n]=size(a)
» [x,y,z]=size(a)
» m2=size(a,2)
```

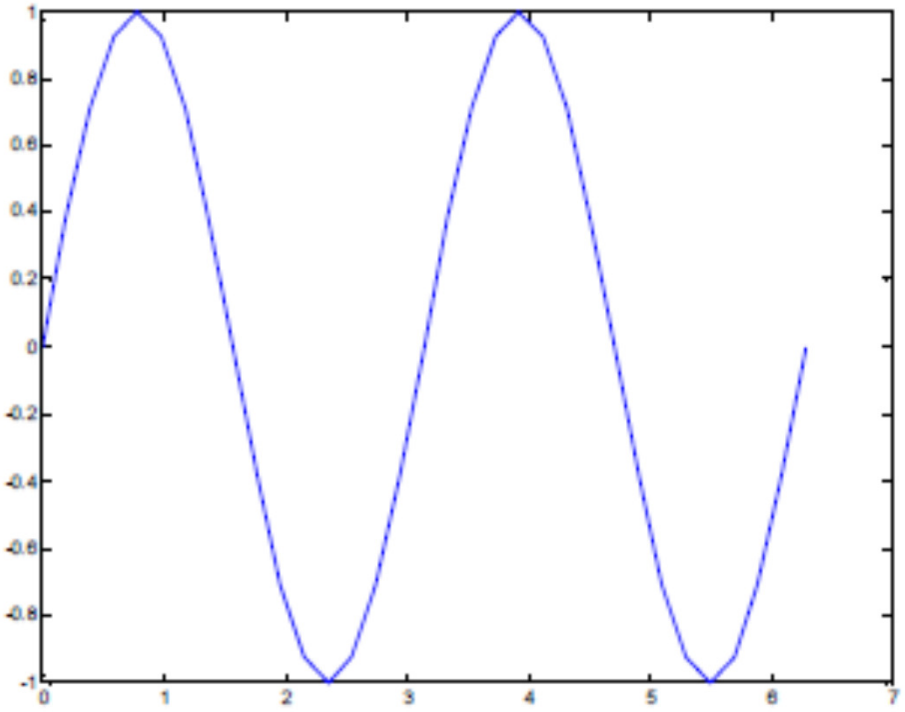
- You can overload your own functions by having variable input and output arguments (see varargin, nargin, varargout, nargout)

Functions: exercise

- Write a function with the following declaration:

  function plotSin(f1)

- In the function, plot a sin wave with frequency f_1, on the range [0,2π]: $\sin(f_1 x)$
- To get good sampling, use 16 points per period.

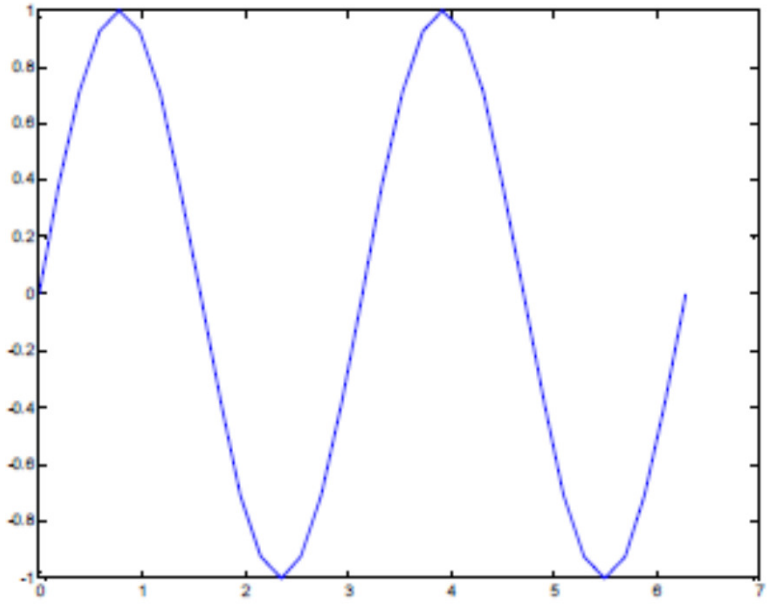


Functions: exercise

- In an MATLAB file saved as plotSin.m, write the following:

```
» function plotSin(f1)

  x=linspace(0,2*pi,f1*16+1);
  figure
  plot(x,sin(f1*x))
```



Outline

- GNU Octave
- Scripts
- Variables (Scalars, Vectors and Matrices)
- Manipulating variables
- Basic plotting
- Functions
- Flow control
- Help/Docs
- Neural Networks with Octave

Relational Operators

- Octave uses mostly standard relational operators
  - equal `==`
  - **not** equal `~=`
  - greater than `>`
  - less than `<`
  - greater or equal `>=`
  - less or equal `<=`
- Logical operators

	elementwise	short-circuit (scalars)
And	&	&&
Or	|	||
Not	~	
Xor	xor	
All true	all	
Any true	any	

- Boolean values: zero is false, nonzero is true
- See **help .** for a detailed list of operators

if/else/elseif

IF

```
if cond
    commands
end
```

Conditional statement: evaluates to true or false

ELSE

```
if cond
    commands1
else
    commands2
end
```

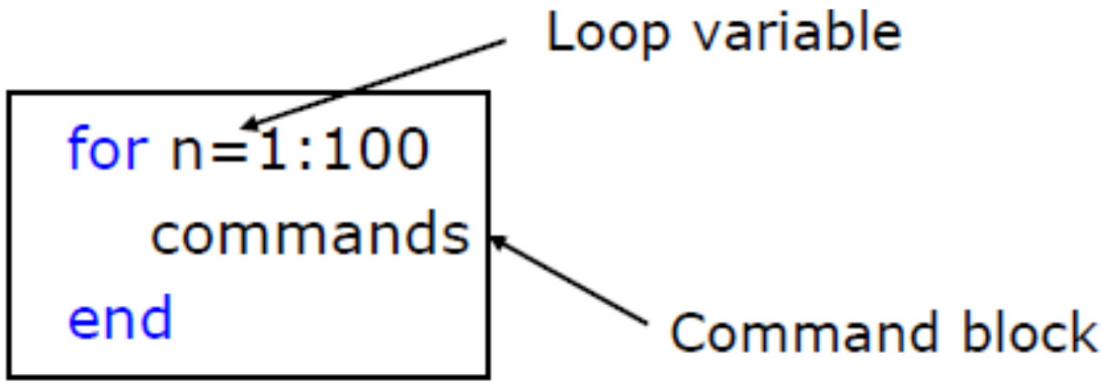
ELSEIF

```
if cond1
    commands1
elseif cond2
    commands2
else
    commands3
end
```

- No need for parentheses: command blocks are between reserved words

for

- for loops: use for a known number of iterations



- The loop variable
  - Is defined as a vector
  - Is a scalar within the command block
  - Does not have to have consecutive values (but it's usually cleaner if they're consecutive)
- The command block
  - Anything between the **for** line and the **end**

while

- The while is like a more general for loop:
  - Don't need to know number of iterations

```
WHILE

while cond
    commands
end
```

- The command block will execute while the conditional expression is true
- Beware of infinite loops!

Outline

- GNU Octave
- Scripts
- Variables (Scalars, Vectors and Matrices)
- Manipulating variables
- Basic plotting
- Functions
- Flow control
- Help/Docs
- Neural Networks with Octave

Help/Docs

- `help`
  - **The most** important function for learning MATLAB on your own
- To get info on how to use a function:

  `» help sin`
  - Help lists related functions at the bottom and links to the doc
- To get a nicer version of help with examples and easy-to-read descriptions:

  `» doc sin`
- To search for a function by specifying keywords:

  `» doc` + Search tab

Outline

- GNU Octave
- Scripts
- Variables (Scalars, Vectors and Matrices)
- Manipulating variables
- Basic plotting
- Functions
- Flow control
- Help/Docs
- Neural Networks with Octave

Support for Neural Networks in Octave

- Octave does not have a GUI tool for NN's (like NNtool in Matlab)

- Some efforts called plexso and octnnetpkg which are not GUI-based are available as packages:

➢ https://sourceforge.net/projects/octnnettb/files/octnnetpkg/
➢ https://sourceforge.net/projects/octnnettb/files/octnnetpkg/nnet-0.1.9/neuralNetworkPackageForOctaveUsersGuide0.1.9.1.pdf/download

- Installation :

➢ download **nnet-0.1.9.tar.gz** and save it to your current working directory.
➢ Then within octave, issue the following from the command line:

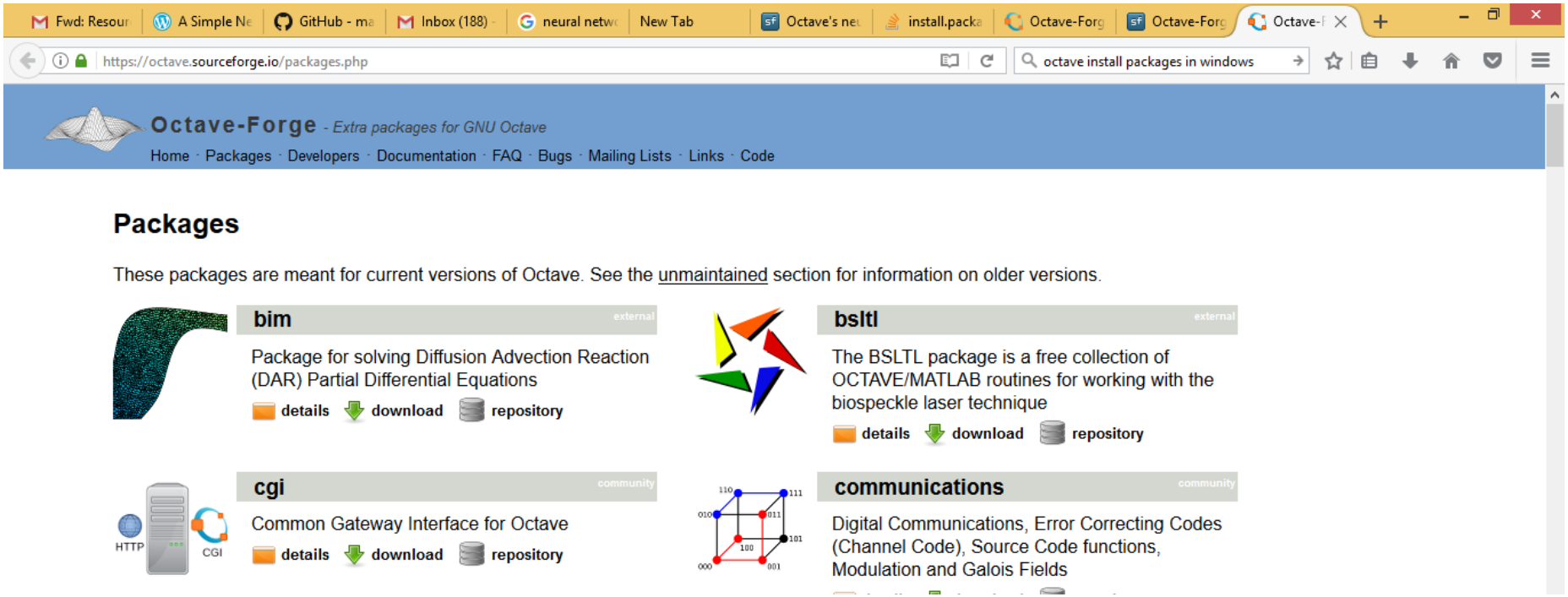
```
>> pkg install  nnet-0.1.9.tar.gz
```

- More code and resources:

➢ https://github.com/massie/octave-nn
➢ https://aimatters.wordpress.com/2015/12/19/a-simple-neural-network-in-octave-part-1/

Octave-Forge :
Extra packages for GNU Octave

- Octave-Forge is a community project for collaborative development of GNU Octave extensions, called “Octave packages” https://octave.sourceforge.io/packages.php
- Octave extensions are developed and released on SourceForge https://sourceforge.net/projects/octave/



Summary & Questions?

- GNU Octave
- Scripts
- Making variables (Scalars, Vectors and Matrices)
- Manipulating variables
- Basic plotting
- Functions
- Flow control
- Help/Docs
- Neural Networks with Octave

Thank You!!