Introduction to GNU-Octave

Dr. K.R. Chowdhary, Professor &
Campus Director, JIETCOE

JIET College of Engineering
Email: kr.chowdhary@jietjodhpur.ac.in
Web-Page: http://www.krchowdhary.com

July 11, 2016

What is Octave?

- Open source interactive software tool for numerical computations and graphics
- Solving Simultaneous equations, computing Eigen-vectors and Eigen-values
- Data can be expressed in matrix and vectors
- Has its own programming language
- More suited for Engineering problem solution, & equivalent to Matlab
- Most of the functionality of MATLAB already exists in GNU Octave and octave can run most MATLAB scripts
- More efficient than C++ or other HLLS
- Has GUI as well command-line interface
- Is interpreted language
- dynamically typed

MATLAb vs GNU Octave

- GNU Octave is mostly compatible with MATLAB. However, Octave's parser allows some (often very useful) syntax that MATLAB's does not, so programs written for Octave might not run in MATLAB.
- Octave supports the use of both single and double quotes. MATLAB only supports single quotes
- Octave supports C-style autoincrement and assignment operators, MATLAB does not: i++; ++i; i+=1; etc.
- Octave supports temporary expressions: columns = size(mtx)(2); tmp = size(mtx)
- Install on linux: $ sudo apt-get install octave
- Available on windows, Linux, mac

Installing octave with GUI for Ubuntu 16.04

```
Method 1 - Using PPA:

sudo apt-add-repository ppa:octave/stable
sudo apt-get update
sudo apt-get install octave

method 2 - Compiling the source yourself:

sudo apt-get build-dep octave
wget ftp://ftp.gnu.org/gnu/octave/octave-4.0.0.tar.gz
tar xf octave-4.0.0.tar.gz
cd octave-4.0.0/
./configure
make
sudo make install

More tools of octve can be installed by "$ synaptic"
command at pronmpt or through menu.
```

Simple Commands

```
>> 2+2
ans=4

>> exp(1)
ans=2.7183

>> 1.2*sin(40*pi/180+log(2.4^2))
ans=0.76618

>> who
ans

>> format long
>> ans
ans=0.766177651029692

>> format short
>> ans
=0.76618
```

functions

cos: Cosine of an angle (in radians)

sin: Sine of an angle (in radians)

tan: Tangent of an angle (in radians)

exp: Exponential function (ex)

log: Natural logarithm (NB this is loge , not log10)cos

log10: Logarithm to base 10

sinh: Hyperbolic sine

cosh: Hyperbolic cosine

tahh: Hyperbolic tangent

acos: Inverse cosine

acosh: Inverse hyperbolic cosine

functions

asin: Inverse sine

asinh: Inverse hyperbolic sine

atan: Inverse tangent

atan2: Two-argument form of inverse tangent

atanh: Inverse hyperbolic tangent

abs: Absolute value

sign: Sign of the number (-1 or +1)

round: Round to the nearest integer

floor: Round down (towards minus infinity)

ceil: Round up (towards plus infinity)

fix: Round towards zero

rem: Remainder after integer division

Simple Commands

Loading and saving data files:

```
>> save anyname
(saves to anyname.mat, in current directory)
Later on it can be retrieved by:

>> load anyname

>> help sqrt

>> a=[1 4 5]
a =
    1   4   5
s= "hello world!"
s= ’Hello world!’
```

Simple Commands

```
>> b=[2, 1, 0];   # vector
b= 2  1  0

>> c= [1 4; 3 10]; #  matrix
c = 1  4
    3  10
>> inv(c)
    -2.0    1.0
     1.5   -0.5
>> e = 2:6
e=2 3 4 5 6

>> e=2:0.3:4
e=2.000  2.300   ....    3.800
```

Simple Commands

```
>> a=[1:2:6 -1  0]
a=1 3 5 -1 0
>>a(3)
ans= 5
>> a(3:5)
5  -1  0
>> a*2
2 6 10 -2 0
>> b=[1 2 3 4 5 6]
b= 1 2 3 4 5 6
>> a.^2
ans=1   9   25   1  0
>> b.^2
ans=1  4   9  16   25  36
```

Control structure

```
#If statement: if.m
vector = [ 1 2 3 4 5];
if length(vector) < 4
    vector(4) = 0;
else
    vector(4)
end

#Loops: for.m
for i = 1:10
    i;
endfor
#loop: while.m
while i <= 10
   i++;
endwhile
```

The transpose operator

```
>> A
A = 5 7 -1
    3 9 -2
>> A'
ans = 5  3
      7   9
     -1  -2

>> I = eye(4)
I =  1  0  0  0
     0  1  0  0
     0  0  1  0
     0  0  0  1
```

Matrices and vectors

```
>> A=[5 7 9
   -1 3 -2]
A = 5 7  9
  -1 3  -2
>> B=[2 0; 0 -1; 1 0]
B= 2    0
   0    -1
   1    0
>> C = [1:3; 8:-2:4]
C = 1  2  3
   8   6  4
Matrix multiplication
>> A*B
ans = 19 -7
     -4  -3
>>B*C
ans = 2  4  6
     -8 -6 -4
      1   2 3
```

Solving simultaneous linear equations

```
let x1=1, x2=2, x3=-1, x4=-2, so
  x1 + 2x2 -  x3 +  x4 = 4
 2x1 + x2  + 3x3 -  x4 = 3
 3x1 - x2  + 2x3 + 2x4 = -5
 -x1 - x2  + 3x3 + x4 = -8

 A =[1 2 -1 1; 2 1 3 -1; 3 -1 2 2; -1 -1 3 1];
 b =[4; 3; -5; -8];
 x= A \ b
 x =
   1.0000
   2.0000
  -1.0000
  -2.0000
```

Plotting a graph

```
>> x=[0:pi/3:2*pi]
x=0.0000    ....         6.28319

>> y=sin(x)
y=0.0000     ....  0.86603 .....-0.0000
```

Plot the Graph by command:

```
>> plot(x, y)
```

The graph displayed can be saved though gui menu or by command:

```
>> print('graph1.eps','-deps')
```

This program is plotprog.m

Plotting graphs

Thse plots more accurate graph: accuplot.m

```
>> x=linspace(0, 2*pi, 1000);
>> y = sin(x);
>> plot(x, y);
```

Improving the presentation:

```
>> title('Graph of y = sin(x)')
>> xlabel('Angles')
>> ylabel('value')
>> grid on
```

Clear graph by

```
>> clf
```

Multiple graphs can be created by:

```
>> plot(x, y, ':', x, cos(x), '-')
```

A rectified sine wave

```
>> edit using  editor and save as rectsin.m
t = linspace(0, 10, 100);
y = abs(sin(t));
plot(t,y);
title('Rectified Sine Wave');
xlabel('t');

can also run by command
>> rectsin<cr>
```

Control structures

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

```
>> a=0; b=2;
>> if a > b
          c=3
      else
           c=4
      end
c = 4
>> 1 == 2
ans=0
```

Control structures

```
switch x
case x1,
    statements
case x2,
   statements
otherwise,
   statements
end
>> a=1;
>> switch a
  case 0
      disp(’a is zero’);
  case 1
     disp(’a is one’);
otherwise
     disp(’a is not a binary digit’);
end
a is one
```

Control structures

```
for variable = vector
    statements
end
>> for n=1:5
   nf(n) = factorial(n);
 end
>> disp(nf)
   1   2     6     24     120
while expression
  statements
end
For example,
>> x=1;
>> while 1+x > 1
    x = x/2;
end
>> x
x = 1.1102e-016 # smallest no.
```

Putting several graphs in one window

subplot(rows, columns, select):

```
>> x = linspace(-10, 10);
>> subplot(2,1,1) % Specify two rows, one column, and select
>>  % the top one as the current graph
>> plot(x, sin(x));
>> subplot(2,1,2);
>> plot(x, sin(x)./x);
```

3D plots:

```
1>> t = 0:pi/50:10*pi;
>> x = sin(t); y = cos(t); z = t;
>> plot3(x, y, z);
```

Putting surfaces

```
>> x = 2:0.2:4; % Define the x- and y- coordinates
>> y = 1:0.2:3; % of the grid lines
>> [X,Y] = meshgrid(x, y); %Make the grid
```

For example, to plot f(x, y)=(x-3)^2-(y-2)^2 over the grid calculated earlier, you would type:

```
>> Z=(X-3).^2 - (Y-2).^2;
>> surf(X,Y,Z)
```