

# 1 Getting Started

1. Go to a workstation in room 249 in Research I
2. Log in using your new account name and password
3. Open a terminal using the mouse following the instructions given in class.
4. Type the command `passwd`
5. Enter your new password - make sure it is at least 6 characters with one number and one punctuation mark. Make sure to remember this for the whole semester!
6. Type the command `octave`
7. Open a web browser, and go to the class web site.
8. Follow the rest of the instructions in the tutorial.

## 2 Basics

Type in the following

```
# a comment - not processed by octave
```

```
# simple math  
2+ 3
```

```
# a sine function and cosine  
sin(3)  
cos(3)
```

```
# defining variable  
x = 3
```

```
# using a variable in a function  
cos(x)
```

```
# evaluate a function into a variable  
y = cos(x)
```

```
# not printing the result out  
y = cos(x);
```

Use octave to answer the following questions

1. What is  $e^3$ ?
2. Set  $x= 2$ .
  - (a) what is  $\cos^2(x) + \sin^2(x)$ ?
  - (b) What is  $\tan(x)$ ?
  - (c) What is the natural log of  $x$ ?
  - (d) What is the log (base 10) of  $x$ ?

### 3 Basic Arrays

Type in the following

```
# setting up a one dimensional array of values - a vector
x = [0,0.3,0.8,1.5,2,2.4,3,4,5,5.8,7.1]
```

```
# function of a vector - creates a new vector
y = cos(x)
```

```
# plotting a single vector
plot(y)
```

```
# plotting two vectors
plot(x,y)
```

```
# plotting with a symbol
plot(x,y,"*")
```

```
# plotting with symbols and lines
plot(x,y,"*-")
```

```
#setting up a random 2d array
rand(3,4)
```

```
# 3d random array
rand(3,4,2)
```

Do the following exercises-

1. Plot  $\sin(x)$  using the x vector above.
2. Set a vector xx from 0 to 5 with spacing of 1.
3. Plot  $e^x$  using the vector you just created.
4. Do the following matrix manipulations. You will need to type these in rather than using copy and paste.
  - (a) Transpose the vector x using x' transposing a vector
  - (b) Calculate the inner product of x\*x'
  - (c) Calculate the outer product of x'\*x

## 4 Generating Arrays and Advanced Plotting

Type in the following commands into octave

```
#clear the memory
clear

#setting up a random 2d array
a = rand(3,4)

# 3d random array
aa = rand(3,4,2)

# set up a 2d array manually - ; are break for the rows
b = [1,2,3;4,5,6;7,8,9]

# access the second row, third column
b(2,3)

#multiply an array by a scale
b *5

# subtract a scale from an array
b -2

# create a one dimension vector
c = [3,4,5]
```

1. Multiply the matrix b times the vector c using the command b\*c'
2. Solve the equation

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \\ 5 \end{pmatrix} \quad (1)$$

by using the command a 'backslash' b, where backslash is the keyboard character

3. Verify the solution using by multiplying the original matrix times the solution vector, then subtracting the the right-hand side
- 4.

## 5 Generating Arrays and Advanced Plotting

Type in the following values into octave.

```
# creating an evenly space array from 0 to 3 with 100 pts
x = linspace(0,3,100);
```

```
# finding the length of a vector
size(x)
```

```
# finding the minimum and maximum value of a vector
min(x)
max(x)
```

```
# creating a function of this space using an exponential
y = exp(-0.3*x);
plot(x,y)
```

```
# two plots of two different functions
y1 = exp(-0.2*x);
plot(x,y,x,y1);
```

```
# taking the natural log of a function
y2 = log(y1);
plot(x,y2)
```

```
# fitting the equation  $y = ax + b$  to the values
cc = polyfit(x,y2, 1);
```

```
# evalating the polynomial with its new coefficients
y3 = polyval(cc,x);
plot(x,y2, x,y3,"*")
```

```
# transforming this back to the original exponentials
y4 = exp(y3);
plot(x,y1, x,y4,"*")
```

1. Find the maximum and minimum of  $y1$ .

## 6 Simple Loops and Fitting

```
# simple loops
for i = 1:10
    x = i
end

# steps in loops
for i= 1:2:8
    x= i
end

# assigning elements in loops
clear x,y
for i= 1:100;
    x(i) = i;
    y(i) = 3*x(i)^3 - 2*x(i)^2 + x(i)*5 + 3;
end
plot(x,y)

# fit the data to a cubic
cc = polyfit(x,y,3)

# using the wrong model - first clear the variable cc
clear cc;
cc = polyfit(x,y,2)
y1 = polyval(cc,x);
plot(x,y, x,y1,"*")

# another wrong model
clear cc;
cc = polyfit(x,y,1)
clear y1
y1 = polyval(cc,x);
plot(x,y, x,y1,"*")

# yet another wrong model
clear cc;
cc = polyfit(x,y,4)
clear y1
y1 = polyval(cc,x);
plot(x,y, x,y1,"*")
```

Do the following exercises-

1. Fit the data to a 4th order equation.
2. Plot the data using the new polynomial.
3. What is the equation you are plotting using the new fit?
4. Is this fit better, worse, or about the same as the fit to the cubic from above?

## 7 Defining a Function

Try the following code-

```
# creating a simple function
function output_values = myfunction(x)
    output_values = x*x + 3;
endfunction

# evaluating
myfunction(2)
myfunction(3)

# clearing a variable
clear x
clear y
x = linspace(0,5,100);
y=myfunction(x);

# since that doesn't work - note the additional period after each operation + and *
function output_values = myfunction(x)
    output_values = x.*x .+ 3;
endfunction

y = myfunction(x);

plot(x,y,"*")
```

Now answer the following questions-

1. Define a function that calculates  $e^{-x^2}$  for a vector of values.
2. Evaluate the function in the interval between -4 and 4
3. Plot the result
4. Using the *trapz* function, evaluate the area under the function.
5. Square the area.

## 8 Baseball Problem

```
function position=baseball(x0, v0, t)
    g = -9.8;
    x = x0(1) + v0(1)*t;
    y = x0(2) + v0(2)*t + 0.5*g*t^2;
    position=[x,y];
endfunction

x0(1) = 1;
x0(2) = 0;
v = 30;
angle = 45;
v0(1) = v * cos(angle*pi/180);
v0(2) = v * sin(angle*pi/180);

t0 = 0;
tf = 5;
nsteps = 101;
dt = (tf-t0)/nsteps;

y(1) = x0(1);
i = 1;
while (y(i) > 0)
    i = i + 1;
    t = t0 + dt*i;
    pos = baseball(x0,v0, t);
    x(i) = pos(1);
    y(i) = pos(2);
end
plot(x,y)

# simple random numbers and arrays of random numbers
rand
rand(3,3)

# adding noise to data
for i = 1:nsteps
    yy(i) = y(i) + (rand-0.5) * 7;
end
plot(x,y,x,yy,"*")

# fitting the data and plotting the result
cc = polyfit(x,yy,2);
ygen = polyval(cc,x);
plot(x,yy,"*", x,y,x,ygen)
```

## 9 Using Random Numbers to Calculate $\pi$

```
# generate a large number of random numbers and plot them
n = 1000;
val= rand(n,2);
plot(val(:,1),val(:,2),".")

# rescale the vector of random numbers
val = val * 2 -1;
plot(val(:,1),val(:,2),".")

# find the sum of the square of the random numbers
for i = 1:n
val(i,3) = sqrt(val(i,1)^2 + val(i,2)^2);
end

# Count the number of values with distances less than 1
ct = 0;
for i = 1:n
    if (val(i,3) < 1)
        ct = ct + 1;
    end
end
ct

newpi = ct/n*4;
newpi
printf("newpi = %g \n",newpi)
```

## 10 Labeling and Printing Plots

```
# radioactive half life of two different elements

# set the time values over 1000 years
ttime = linspace(0,1000,50);

# set the half life and initial concentration of element 1
thalf1 = 50;
initial1 = 100;

# set the half life and initial concentration of element 1
thalf2 = 75;
initial2 = 100;

# find the concentrations for both elements
amount1 = initial1 * (1/2).^(ttime/ thalf1);
amount2 = initial2 * (1/2).^(ttime/thalf2);

# plot them using colors with lines and points
plot(ttime, amount1,"+r-r",ttime, amount2, "*b-b")

# label the graph
xlabel("tme - yrs");
ylabel("concentration")
title("radioactive decay")
legend("75 year half life", "100 year half life");

# plot using a log scale on the axis
semilogy(ttime, amount1, "*-r",ttime, amount2,"+-b");

# now save this into a png file in your directory
print -dpng -landscape -color newimage.png
```

## 11 Creating a Data File

```
# set up parameters for a radioactive decay
npts = 101;
tstart = 0;
tfinal = 5;
time = linspace(tstart, tfinal, npts);
initial_concentration = 1000;
half_life = 2.1;

# calculate calculate the concentrations
y = initial_concentration * exp(-time/ half_life);

# add random values equal to the square root of the concentration
# to the values. Use a normalized random concentration with
# the random numbers using the randn function. Because
# the value can be negative, we need to make a correction if that
# happens.
for i = 1: npts
    c = y(i);
    dy = sqrt(y(i))* (2 * randn -1 );
    if (y + dy < 0)
        dy = abs( dy);
    end
    y(i) = y(i)+ dy;
end

# take the log of the concentrations
yl = log(y);
# write the data to a file
[FID,MSG] = fopen("newfile","w+");
for i = 1: npts
    fprintf(FID,"%8g, %10g, %10g\n",time(i),y(i),yl(i));
end
fclose(FID);
```

## 12 Reading an Ascii Data File

```
a =load ("newfile" );  
  
# fit the data with a line  
cc = polyfit(a(:,1),a(:,3),1)  
lfit = polyval(cc,a(:,1));  
  
plot(a(:,1),a(:,3),'*',a(:,1),lfit)
```

1. Try fitting a curve to the concentration, rather than the log of the concentration
2. Write the data out to a file, including your fit to a curve.

## 13 Fourier Transforms

Open an editor and enter the following text. Save this file as "fftplt.m".

```
function jj = fftplt(val)

    # set up the number of points and make a sin curve
    # with the frequency set by the input variable val
n= 2048;
x = linspace(0,n-1,n);
y =sin(val*2*pi*x/n);

    # take the fourier transform of the sine curve
yy = real(fft(y));

    # find the power of the spectrum by squaring each value
zz = yy.^2;

    # plot the sine curve and the first few terms of the Fourier transform
subplot(2,1,1)
plot(x,y)
subplot(2,1,2)
plot(x,zz,"*r")
mm = max(zz) * 1.2;
axis([0,16,0,mm])
jj = 0;
endfunction
```

1. Take the function "fftplt" of the values 1 through 10. Note the changes in the time plot on the top and the frequency plot on the bottom.
2. What is the relationship between the number of cycles and the top graph and the frequency on the bottom graph?
3. Type "subplot(1,1,1)" to move the graphics back to a single frame.

## 14 Three dimensional graphics

The following code generates a three dimensional set of data where  $z$  is a function of  $x$  and  $y$ . The value of  $z$  depends on the distance from the center of the domain at  $(0,0)$ , and is  $\cos(r/3) \exp(r/3)$ , where  $r$  is the distance.

```
ngrid = 64;
x = linspace(-6,6,ngrid);
y = linspace(-6,6,ngrid);
for i = 1: ngrid
    for j = 1:ngrid
        r = sqrt( x(i)^2 + y(j)^2);
        z(i,j) = exp( -r/3) * cos(r*3);
    end
end
end
```

1. Display the graph using the command "mesh(x,y,z)".
2. Redisplay the graph using the command "surf('x,y,z)".
3. Change the viewing angle using the command "view(10,80)".
4. Change it again using the command "view(30,45)".

## 15 A few more graphics tricks

```
#Histograms of random data with a normal distribution
hist (randn (10000, 1), 30);
```

```
#Histograms of random data with a uniform distribution
hist (rand(10000, 1), 30);
```

Pie charts

```
aa = [30,60,270];
ex = [0,1,0];
pie(aa,ex)
```

Three dimensional curves

```
t =linspace(0, 30, 100);
r = exp(t/10);
x = r.*sin(t);
y = r.*cos(t);
plot3 (x,y, z);
```

## 16 Images

Save the image from the class website to your local directory, then execute the following script.

```
hh = imread("M31_hallas-small-gray.jpg");
image(hh)

# find the dimensions and max and min pixel values for the image
(n,m) = size(hh)
zmax = max(max(hh));
zmin =min(min(hh));

# filtering all the values that are less than 80% of the peak value
aa= hh;
for i = 1:n
    for j = 1:m
        if (hh(i,j) < zmax * 0.8)
            aa(i,j) = 1 ;
        end
    end
end
# redisplay the image
image(aa)

# flip the image
cc =flipup(hh);
image(cc)
```