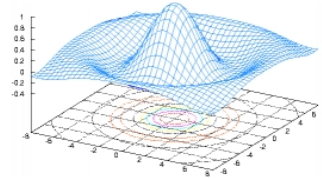


## Octave

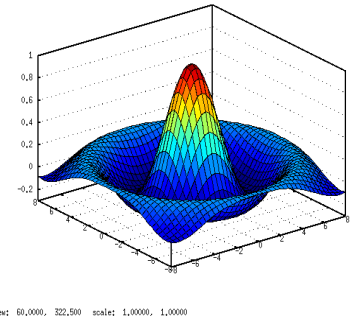
Home
About Octave
News Archive
Docs
Wiki
FAQ
Help
Bugs
License
Download
Related Projects
Mailing Lists
Funding
Contributors



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

convenient command line interface for solving linear and nonlinear problems numerically, and for performing other numerical experiments using a language that is mostly compatible with Matlab. It may also be used as a batch-oriented language.

For more information, see the page [about Octave](#).



# Hands on Matlab & Octave

Matlab provides a Data Analysis framework useful for multiple purposes not just for particle physics. GNU Octave is mostly compatible with Matlab and it also can be used as a batch-oriented language and it's free.

They are being used in several research groups.

Matlab/Octave help is very useful.

>help *whatever*

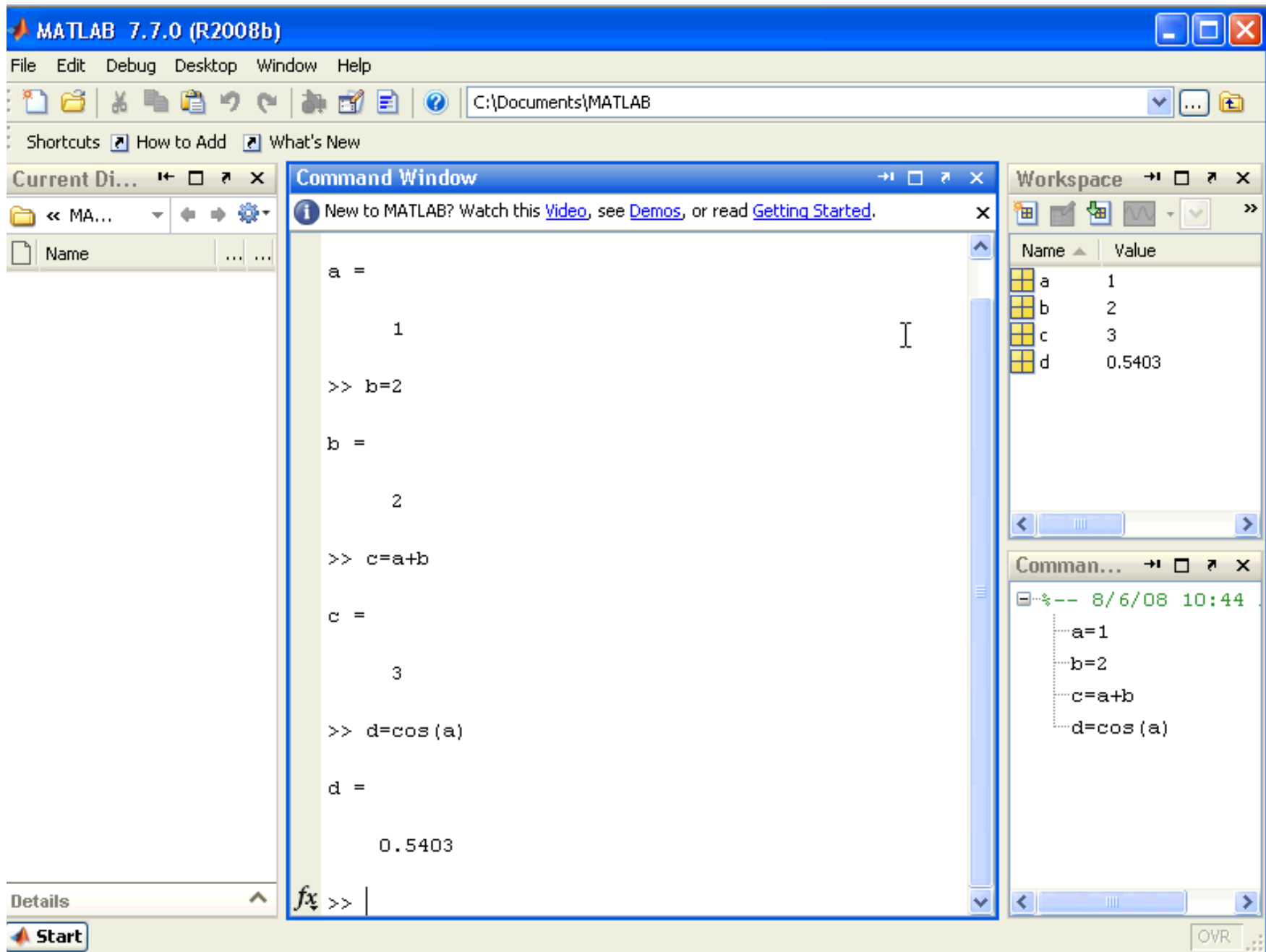
There are also several sources of help online. Matlab official website is:

<http://www.mathworks.com/>

An excellent source of statistics documentation

<http://www.jstatsoft.org/>

# Shutting a mosquito with a cannon



The image shows the MATLAB 7.7.0 (R2008b) interface. The Command Window displays the following code and output:

```
a =  
    1  
  
>> b=2  
  
b =  
    2  
  
>> c=a+b  
  
c =  
    3  
  
>> d=cos(a)  
  
d =  
    0.5403
```

The Workspace window shows the following variables and their values:

Name	Value
a	1
b	2
c	3
d	0.5403

The Command Window History shows the following commands:

```
8/6/08 10:44  
a=1  
b=2  
c=a+b  
d=cos(a)
```

# Vectors & Matrices

The image displays the MATLAB 7.7.0 (R2008b) software interface. The main window is titled "MATLAB 7.7.0 (R2008b)" and contains several panes:

- Command Window:** Shows the execution of MATLAB commands and their outputs.

```
c =  
    3  
  
>> d=cos(a)  
  
d =  
    0.5403  
  
>> t=[1 2 3 4 5]  
  
t =  
    1    2    3    4    5  
  
>> t=1:5  
  
t =  
    1    2    3    4    5  
  
fx >> t=0:0.01:1
```
- Workspace:** Displays the current workspace variables and their values.

Name	Value
a	1
b	2
c	3
d	0.5403
t	[1,2,3,4,5]
- Command History:** Shows a list of previously executed commands, including:

```
a=1  
b=2  
c=a+b  
d=cos(a)  
t=[1 2 3 4 5]  
t=1:5
```

# Vectors & Matrices

```
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.

>> t=0:.1:1

t =

Columns 1 through 8

    0    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000    0.7000

Columns 9 through 11

    0.8000    0.9000    1.0000

>> t=t'

t =

    0
    0.1000
    0.2000
    0.3000
    0.4000
    0.5000
    0.6000
    0.7000
    0.8000
    0.9000
    1.0000

>>
```

# Working with vectors

```
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.

t =

    0
 0.1000
 0.2000
 0.3000
 0.4000
 0.5000
 0.6000
 0.7000
 0.8000
 0.9000
 1.0000

>> u=sin(2*pi*t)

u =

    0
 0.5878
 0.9511
 0.9511
 0.5878
 0.0000
-0.5878
-0.9511
-0.9511
-0.5878
-0.0000

fx >> tu = [t u]
```

```
tu =  
  
      0      0  
  0.1000  0.5878  
  0.2000  0.9511  
  0.3000  0.9511  
  0.4000  0.5878  
  0.5000  0.0000  
  0.6000 -0.5878  
  0.7000 -0.9511  
  0.8000 -0.9511  
  0.9000 -0.5878  
  1.0000 -0.0000
```

```
>> tu(3,1)
```

```
ans =
```

```
0.2000
```

```
>> tu(:,1)
```

```
ans =
```

```
      0  
  0.1000  
  0.2000  
  0.3000  
  0.4000  
  0.5000  
  0.6000  
  0.7000  
  0.8000  
  0.9000  
  1.0000
```

Colon ":"?...  
>help :

: is equivalent to 1:end

```
>> tu = [tu cos(t)]
```

```
tu =
```

```
      0      0      1.0000  
  0.1000  0.5878  0.9950  
  0.2000  0.9511  0.9801  
  0.3000  0.9511  0.9553  
  0.4000  0.5878  0.9211  
  0.5000  0.0000  0.8776  
  0.6000 -0.5878  0.8253  
  0.7000 -0.9511  0.7648  
  0.8000 -0.9511  0.6967  
  0.9000 -0.5878  0.6216  
  1.0000 -0.0000  0.5403
```

```
>> tu(:,3)
```

```
ans =
```

```
1.0000  
0.9950  
0.9801  
0.9553  
0.9211  
0.8776  
0.8253  
0.7648  
0.6967  
0.6216  
0.5403
```

```
tu =  
  
    0         0    1.0000  
  0.1000    0.5878    0.9950  
  0.2000    0.9511    0.9801  
  0.3000    0.9511    0.9553  
  0.4000    0.5878    0.9211  
  0.5000    0.0000    0.8776  
  0.6000   -0.5878    0.8253  
  0.7000   -0.9511    0.7648  
  0.8000   -0.9511    0.6967  
  0.9000   -0.5878    0.6216  
  1.0000   -0.0000    0.5403
```

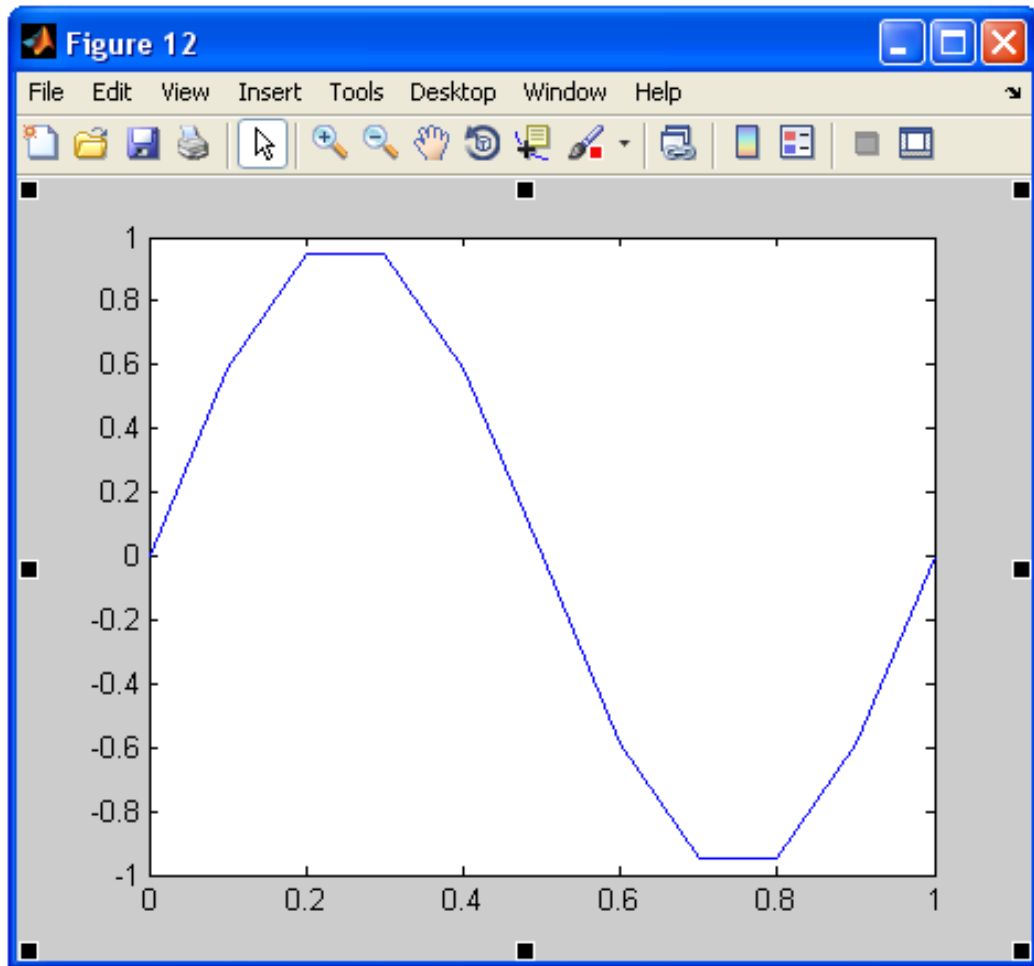
```
>> tu(:,3)=[]
```

```
tu = Delete a column
```

```
    0         0  
  0.1000    0.5878  
  0.2000    0.9511  
  0.3000    0.9511  
  0.4000    0.5878  
  0.5000    0.0000  
  0.6000   -0.5878  
  0.7000   -0.9511  
  0.8000   -0.9511  
  0.9000   -0.5878  
  1.0000   -0.0000
```

```
>> plot (t,u)
```

```
>>
```



# Boolean Operations

```
u =  
    0  
    0.5878  
    0.9511  
    0.9511  
    0.5878  
    0.0000  
   -0.5878  
   -0.9511  
   -0.9511  
   -0.5878  
   -0.0000
```

Numeric  
variable

```
>> u>0
```

```
ans =
```

```
    0  
    1  
    1  
    1  
    1  
    1  
    0  
    0  
    0  
    0
```

Boolean  
variable

```
>> u(u>0)
```

```
ans =
```

```
    0.5878  
    0.9511  
    0.9511  
    0.5878  
    0.0000
```

Applying  
boolean  
variable to  
a numeric  
vector

```
>> u(u>0)
```

```
ans =
```

```
    0.5878  
    0.9511  
    0.9511  
    0.5878  
    0.0000
```

```
>> t(u>0)
```

```
ans =
```

```
    0.1000  
    0.2000  
    0.3000  
    0.4000  
    0.5000
```

```
>> t
```

```
t =
```

```
    0  
    0.1000  
    0.2000  
    0.3000  
    0.4000  
    0.5000  
    0.6000  
    0.7000  
    0.8000  
    0.9000  
    1.0000
```

fx >>

```
u =
```

```
    0  
    0.5878  
    0.9511  
    0.9511  
    0.5878  
    0.0000  
   -0.5878  
   -0.9511  
   -0.9511  
   -0.5878  
   -0.0000
```

```
>> u(u==0)
```

```
ans =
```

```
    0
```

```
>> u(u==0|u>0.5)
```

```
ans =
```

```
    0  
    0.5878  
    0.9511  
    0.9511  
    0.5878
```

```
>> u(u==0|u>0.5&u>0.9)
```

```
ans =
```

```
    0  
    0.9511  
    0.9511
```

fx >>

AND → “&”

OR → “|”

NOT → “~” or “!”

Notice than & applies  
before |. If you don't  
want that you need ()

```
>> t(u==0|u>0.5&u>0.9)
```

```
ans =
```

```
    0  
    0.2000  
    0.3000
```

fx >>

# Bitwise logic

```
>> x=[1:16]
```

```
x =
```

```
     1     2     3     4     5     6     7     8     9    10    11    12    13    14    15    16
```

```
>> bitand(x,2)
```

```
ans =
```

```
     0     2     2     0     0     2     2     0     0     2     2     0     0     2     2     0
```

```
>> bitor(x,2)
```

```
ans =
```

```
     3     2     3     6     7     6     7    10    11    10    11    14    15    14    15    18
```

```
>> logical(bitand(x,2))
```

```
ans =
```

```
     0     1     1     0     0     1     1     0     0     1     1     0     0     1     1     0
```

```
>> x(logical(bitand(x,2)))
```

```
ans =
```

```
     2     3     6     7    10    11    14    15
```

```
fx >>
```

“Logical” converts a number variable into a boolean one. Where zero goes to false or boolean zero and any other number goes to true or boolean one.

# Operations with vectors

The dot operator:

(Dot) “.” before an operation tells that you want that operation element by element.

Examples:

```
> a=linspace(-2,3,7)
```

```
linspace?...  
>help linspace
```

```
a =
```

```
-2.00000 -1.16667 -0.33333  0.50000  1.33333  2.16667  3.00000
```

```
> b=a.^2
```

```
b =
```

```
4.00000  1.36111  0.11111  0.25000  1.77778  4.69444  9.00000
```

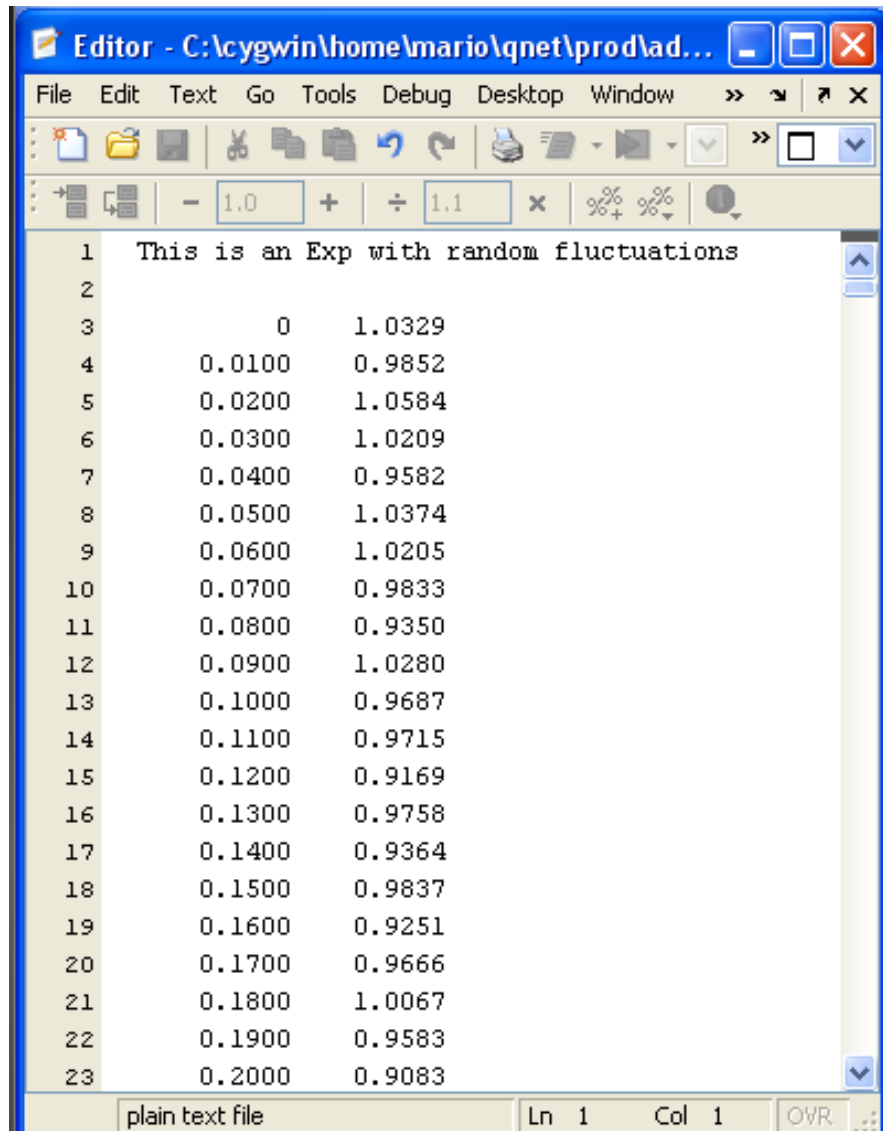
```
> a.*b
```

```
-8.000000 -1.587963 -0.037037  0.125000  2.370370 10.171296 27.000000
```

BUT THAT IS  $a.^3$

Notice that + and - do not need a dot because they naturally operate component by component

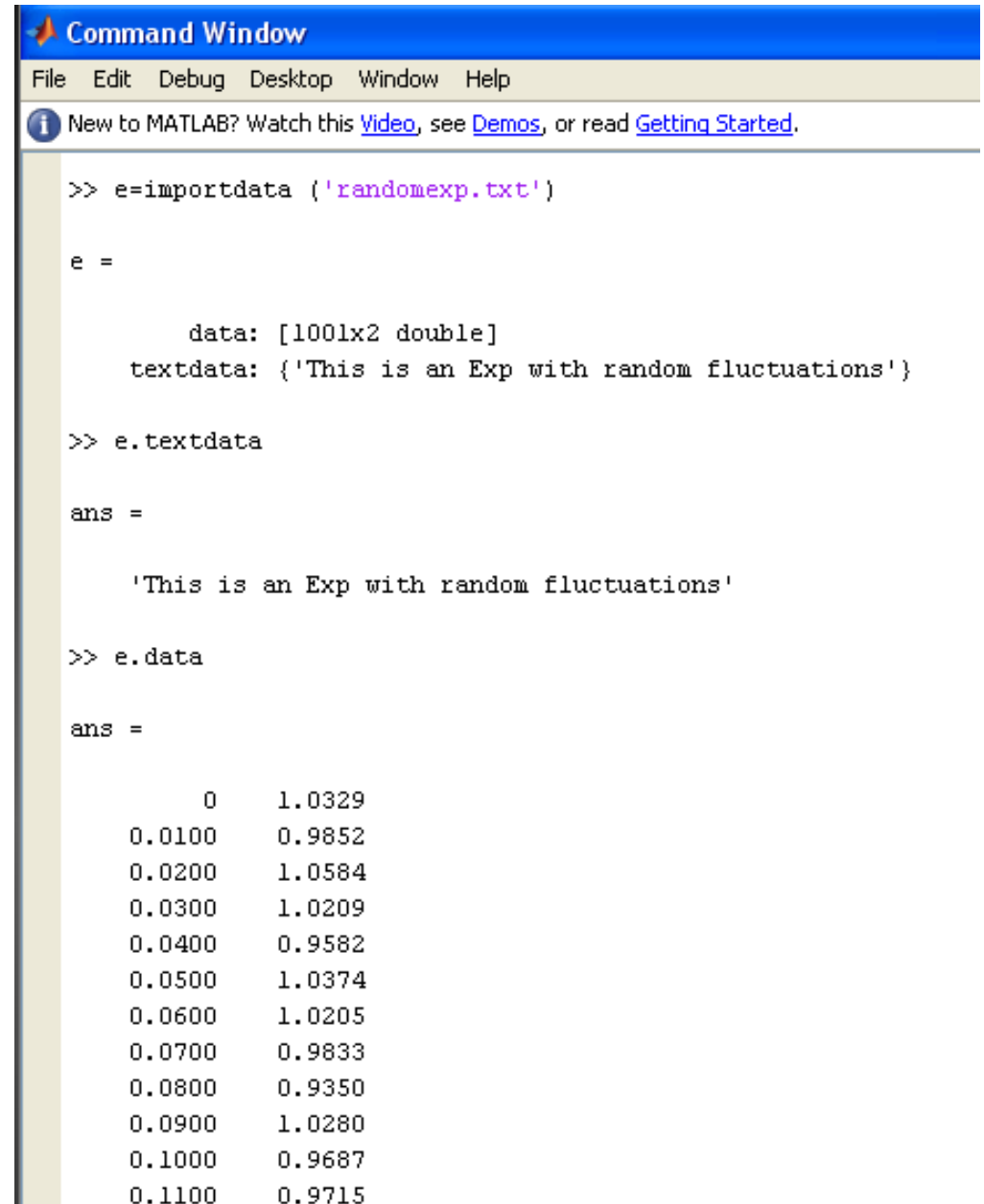
# Import Data



Editor - C:\cygwin\home\mario\qnet\prod\ad...

```
1 This is an Exp with random fluctuations
2
3      0      1.0329
4      0.0100    0.9852
5      0.0200    1.0584
6      0.0300    1.0209
7      0.0400    0.9582
8      0.0500    1.0374
9      0.0600    1.0205
10     0.0700    0.9833
11     0.0800    0.9350
12     0.0900    1.0280
13     0.1000    0.9687
14     0.1100    0.9715
15     0.1200    0.9169
16     0.1300    0.9758
17     0.1400    0.9364
18     0.1500    0.9837
19     0.1600    0.9251
20     0.1700    0.9666
21     0.1800    1.0067
22     0.1900    0.9583
23     0.2000    0.9083
```

plain text file Ln 1 Col 1 OVR



Command Window

```
File Edit Debug Desktop Window Help
New to MATLAB? Watch this Video, see Demos, or read Getting Started.

>> e=importdata ('randomexp.txt')

e =

      data: [1001x2 double]
    textdata: {'This is an Exp with random fluctuations'}

>> e.textdata

ans =

    'This is an Exp with random fluctuations'

>> e.data

ans =

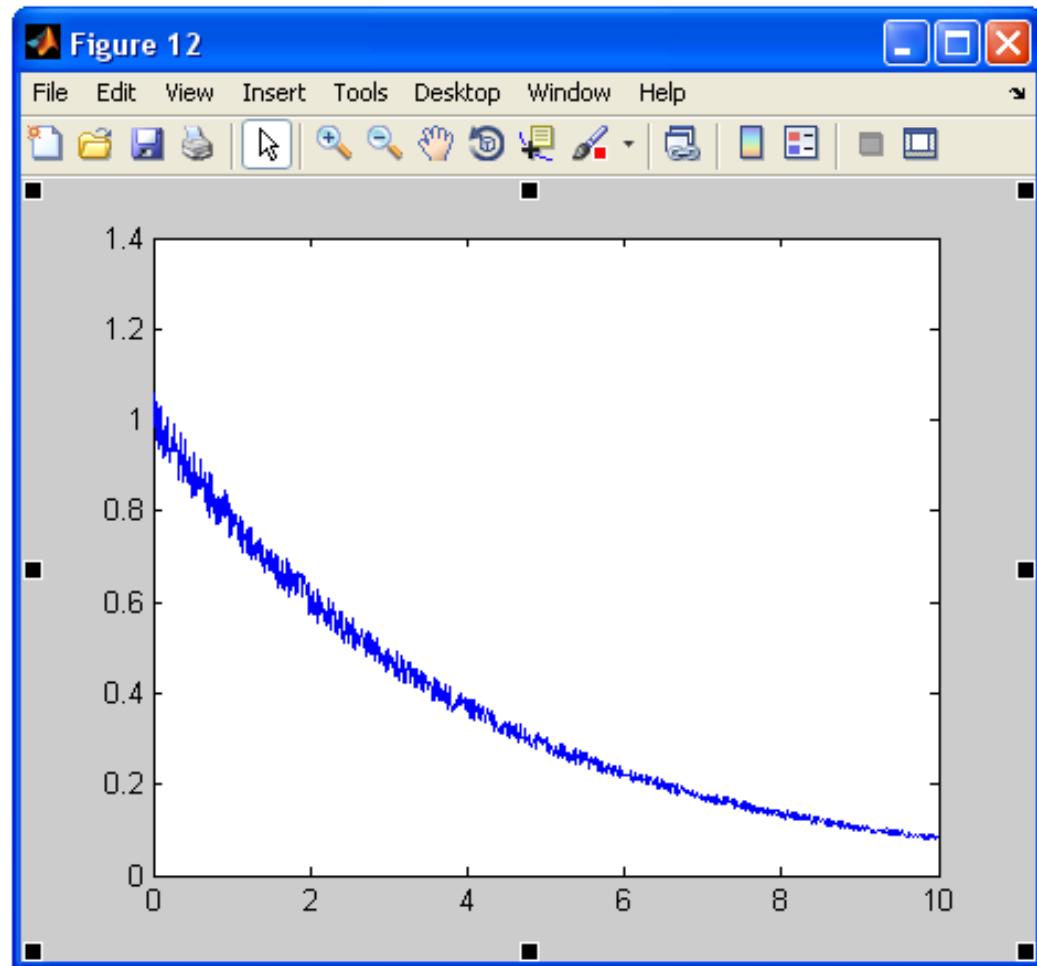
      0      1.0329
    0.0100    0.9852
    0.0200    1.0584
    0.0300    1.0209
    0.0400    0.9582
    0.0500    1.0374
    0.0600    1.0205
    0.0700    0.9833
    0.0800    0.9350
    0.0900    1.0280
    0.1000    0.9687
    0.1100    0.9715
```

# Simple Plot

```
0.0853  
0.0811  
0.0808  
0.0906  
0.0817  
0.0859  
0.0882  
0.0819  
0.0809  
0.0873  
0.0868  
0.0833  
0.0880  
0.0840  
0.0875  
0.0856  
0.0859  
0.0822  
0.0850  
0.0849  
0.0818
```

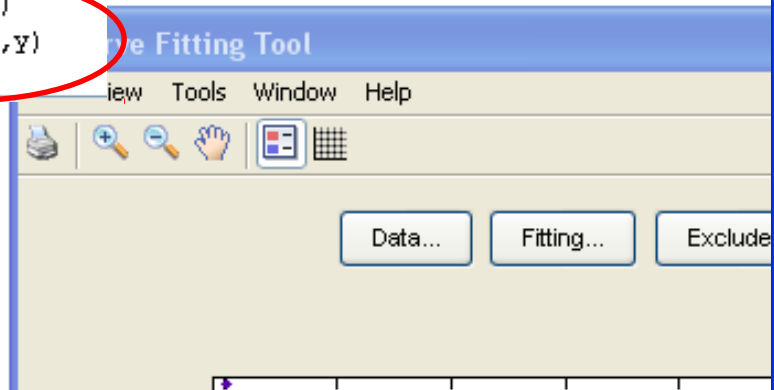
```
>> x=e.data(:,1);  
>> y=e.data(:,2);  
>> plot(x,y)
```

```
fx >>
```



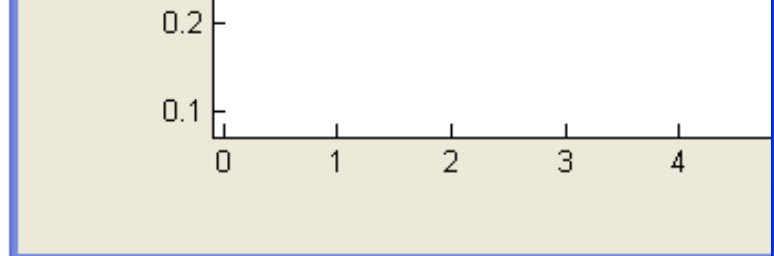
# Fit D

```
>> x=e.data(:,1);  
>> y=e.data(:,2);  
>> plot(x,y)  
>> cftool(x,y)
```



```
>> x=0:.01:10;  
>> size(x)  
  
ans =  
  
         1         1001  
  
>> length(x)  
  
ans =  
  
         1001  
  
>> xr=x+0.5*(rand(1,1001)-.5);  
>> [x' exp(-0.25*xr)'];  
fx >> |
```

Made up data



### Fit Editor

New fit Copy fit

Fit name: fit 1

Data set: y vs. x Exclusion rule: (none)

Type of fit: Exponential  Center and scale X data

Exponential

- a\*exp(b\*x)
- a\*exp(b\*x) + c\*exp(d\*x)

Fit options...  Immediate apply Cancel Apply

### Results

General model Exp1:

$$f(x) = a \cdot \exp(b \cdot x)$$

Coefficients (with 95% confidence bounds):

a = 1.001 (0.9982, 1.005)

b = -0.2501 (-0.2513, -0.2489)

Goodness of fit:

SSE: 0.2473

### Table of Fits

Fit name	Data set	Equation name	SSE
fit 1	y vs. x	Exp1	0.247339364...

Delete fit Save to workspace... Table options...

# Histograms

Histograms are very easy to produce!

1<sup>st</sup> Define the bin's centers. Matlab/Octave will automatically set the bins out of them.

2<sup>nd</sup> plot the histogram...

Examples:

```
> e=rande(1000,1);
```

```
rande?...  
>help rande
```

BINS CENTERS

```
> b=.5:1:6.5
```

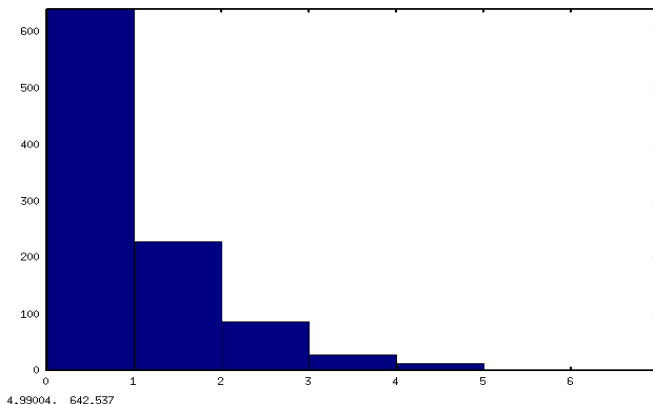
```
b =
```

```
0.50000 1.50000 2.50000 3.50000 4.50000 5.50000 6.50000
```

1<sup>st</sup> bin goes from 0 to 1, 2<sup>nd</sup> bin from 1 to 2, ... last bin goes from 6 to 7

You can set variable bin sizes!

```
> hist (e,b)
```



OR YOU CAN ALSO DO: 

```
> [yh xh]=hist (e,b);
```

yh and xh will be the coordinates of the top and the center of each bar

```
> bar(xh,yh)
```

```
> bar(xh,yh,1)
```

# 3D Plots

TRY THE BUILT IN MEXICAN HAT FUNCTION

```
>sombrero
```

This function is  $\sin(r)/r$  and you don't need the function to generate it.

```
> x=-8:1:8;
```

Can you guess what the semicolon does at the end of a command?

```
> y=x
```

```
y =
```

```
-8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8
```

```
> [xx yy]=meshgrid (x,y)
```

```
xx =
```

```
-8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8
```

```
-8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8
```

```
-8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8
```

```
...
```

```
-8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8
```

17 rows all the same

```
yy =
```

```
-8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8
```

```
-7 -7 -7 -7 -7 -7 -7 -7 -7 -7 -7 -7 -7 -7 -7 -7
```

```
-6 -6 -6 -6 -6 -6 -6 -6 -6 -6 -6 -6 -6 -6 -6 -6
```

```
...
```

```
8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
```

17 columns all the same

# 3D Plots

```
> r = sqrt (xx .^ 2 + yy .^ 2) + eps;
```

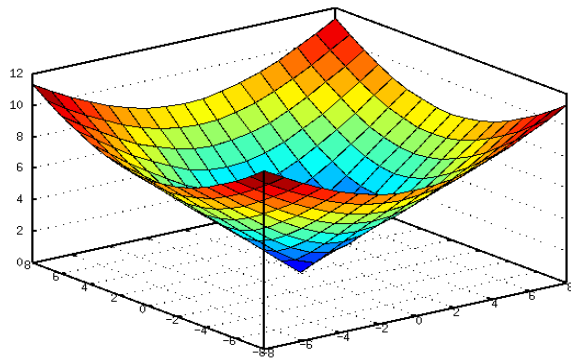
```
eps?  
>help eps
```

How does r look?

Let's plot it then

```
> surf(x,y,r)
```

```
surf?  
>help surf
```

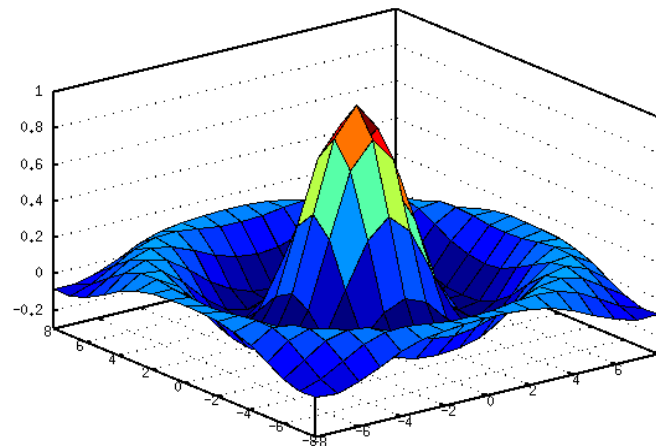


You should be able to rotate your 3D plot

```
view: 60,0000, 322,500 scale: 1,00000, 1,00000
```

```
> figure(2)
```

```
> surf(x,y,sin(r)./r)
```



```
view: 60,0000, 322,500 scale: 1,00000, 1,00000
```

# Hands on

THIS IS THE MOST IMPORTANT SLIDE!!

Please go to a lab computer and go through this presentation.

Redo every single line and make sure you understand how each of them works.

And please ask questions!!