MATLAB Input and Output

Loading your data and plotting it

Beyond The Mouse
April 29, 2009
Glenn Thompson

Outline

- Review of MATLAB programming
- Plotting your data
- Saving your plots as image files
- Loading your data
- Saving your data

Plotting your data with MATLAB

Graphics

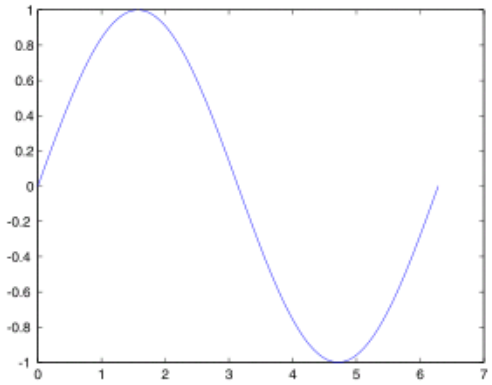
[edit]

Function *plot* can be used to produce a graph from two vectors *x* and *y*. The code:

```
x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y)
```

Just 3 statements to produce a graph of a sine wave

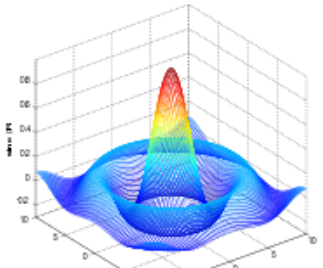
produces the following figure of the sine function:



Three-dimensional graphics can be produced using the functions *surf*, *plot3* or *mesh*.

```
[X,Y] = meshgrid(-10:0.25:10,-10:0.25:10);
f = sinc(sqrt((X/pi).^2+(Y/pi).^2));
mesh(X,Y,f);
axis([-10 10 -10 10 -0.3 1])
xlabel('{\bfx}')
ylabel('{\bfy}')
zlabel('{\bfsinc} ({\bfR})')
hidden off
```

This code produces a **wireframe** 3D plot of the two-dimensional unnormalized sinc function:



```
[X,Y] = meshgrid(-10:0.25:10,-10:0.25:10);
f = sinc(sqrt((X/pi).^2+(Y/pi).^2));
surf(X,Y,f);
axis([-10 10 -10 10 -0.3 1])
xlabel('{\bfx}')
ylabel('{\bfy}')
zlabel('{\bfsinc} ({\bfR})')
```

This code produces a **surface** 3D plot of the two-dimensional unnormalized sinc function:

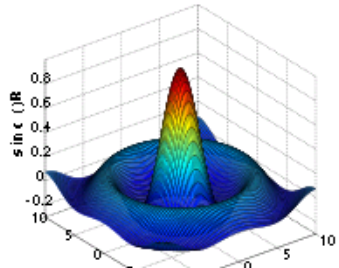
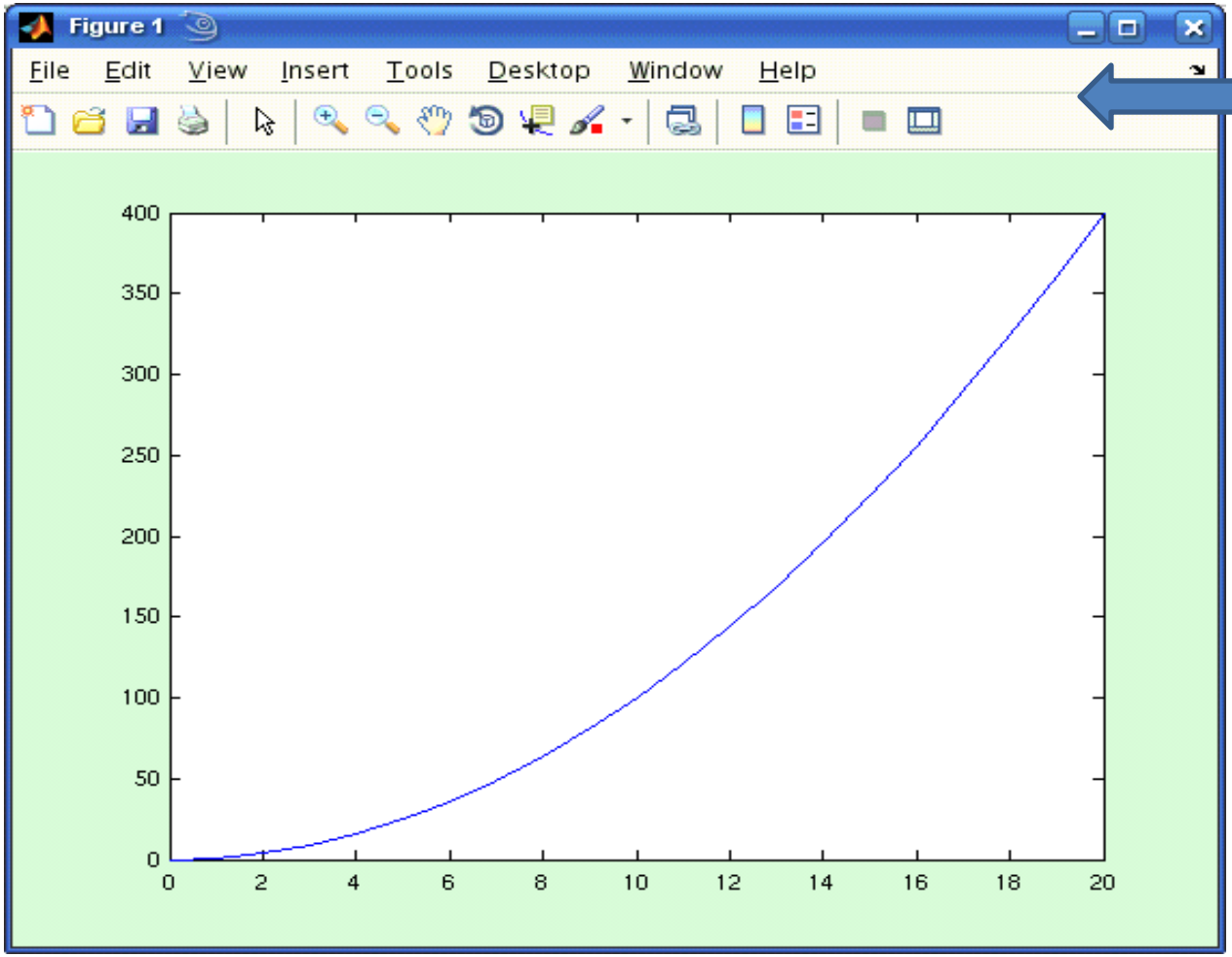


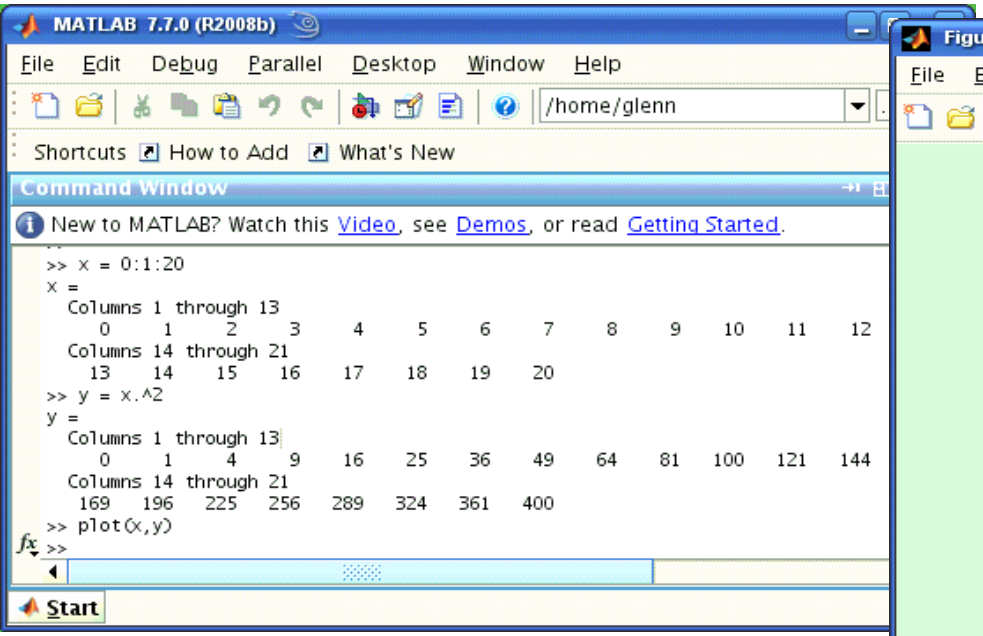
Figure window GUI controls

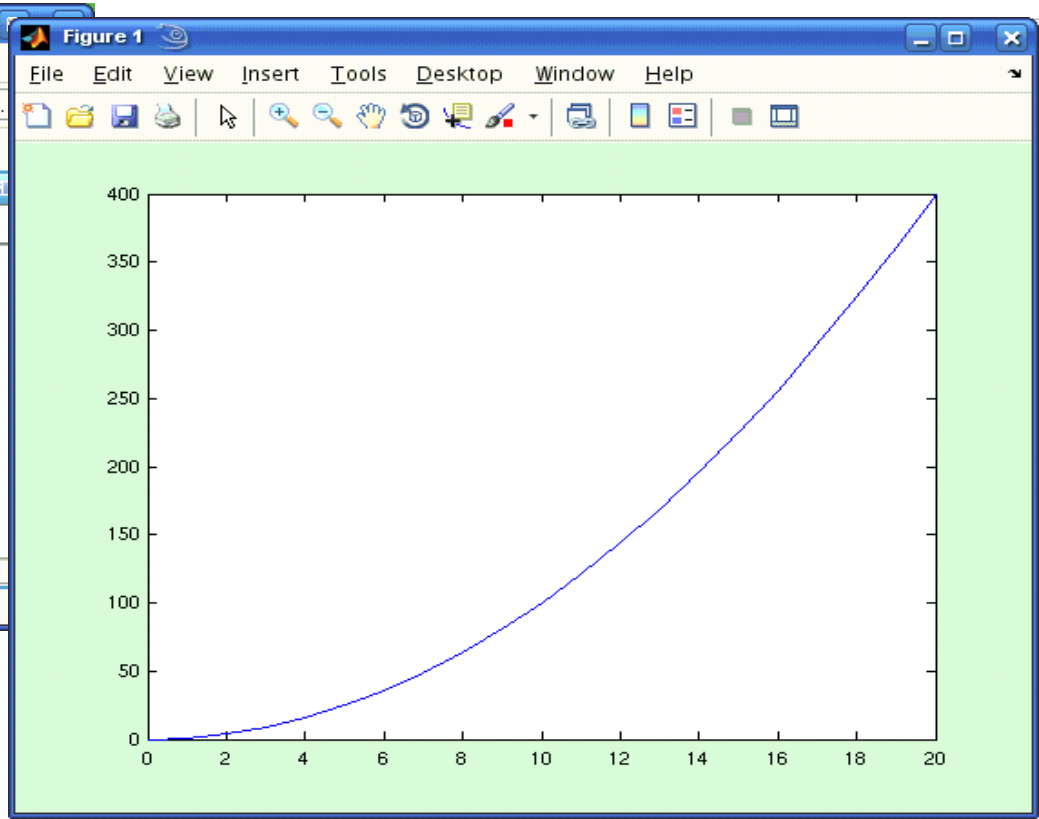


I ignore this stuff, because I want to generate figures in a way that is scriptable and repeatable

plot(x, y)

1. Define the x-vector (can be a matrix)
2. Define the y-vector (can be a matrix)
3. **plot(x,y)** then generates a figure window, a set of axes, and then plots y versus x





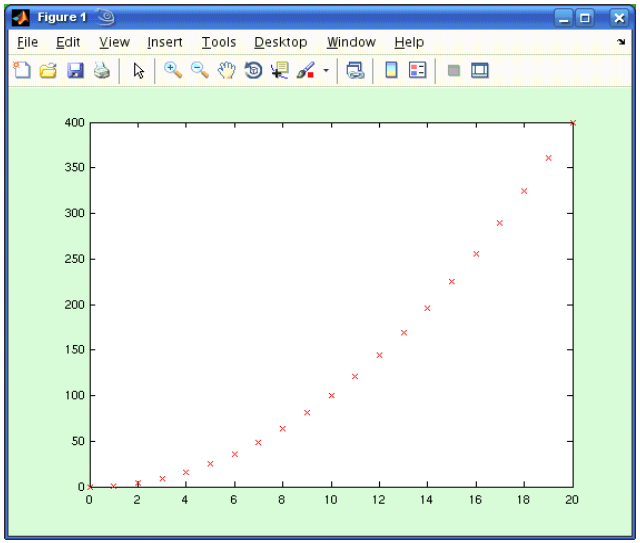
plot(x, y, s)

Let's call s the 'linestyle':
By default, plot uses a blue line to connect data points. But if you do:
>> help plot
You'll see it says:

```
Various line types, plot symbols and colors may be obtained with
    PLOT(X,Y,S) where S is a character string made from one element
    from any or all the following 3 columns:

        b     blue          .     point              -     solid
        g     green         o     circle             :     dotted
        r     red           x     x-mark             -.    dashdot
        c     cyan          +     plus               --    dashed
        m     magenta       *     star             (none)  no line
        y     yellow        s     square
        k     black         d     diamond
        w     white         v     triangle (down)
                            ^     triangle (up)
                            <     triangle (left)
                            >     triangle (right)
                            p     pentagram
                            h     hexagram
```

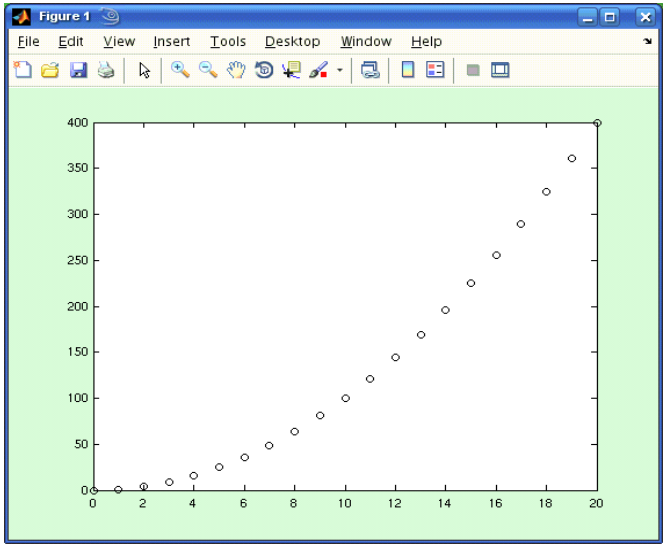
plot(x, y) is the same as **plot(x, y, 'b-')**

plot(x,y,'rx')



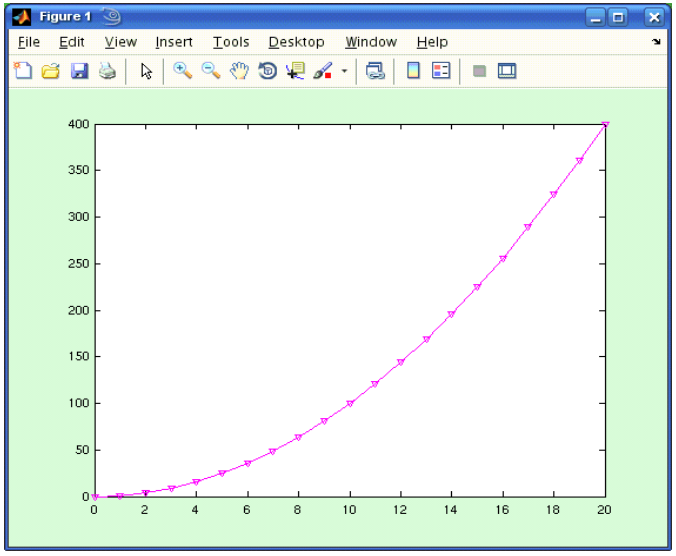
plot y against x using red crosses

plot(x,y,'bo')



plot y against x using black circles

plot(x, y, 'mv-')



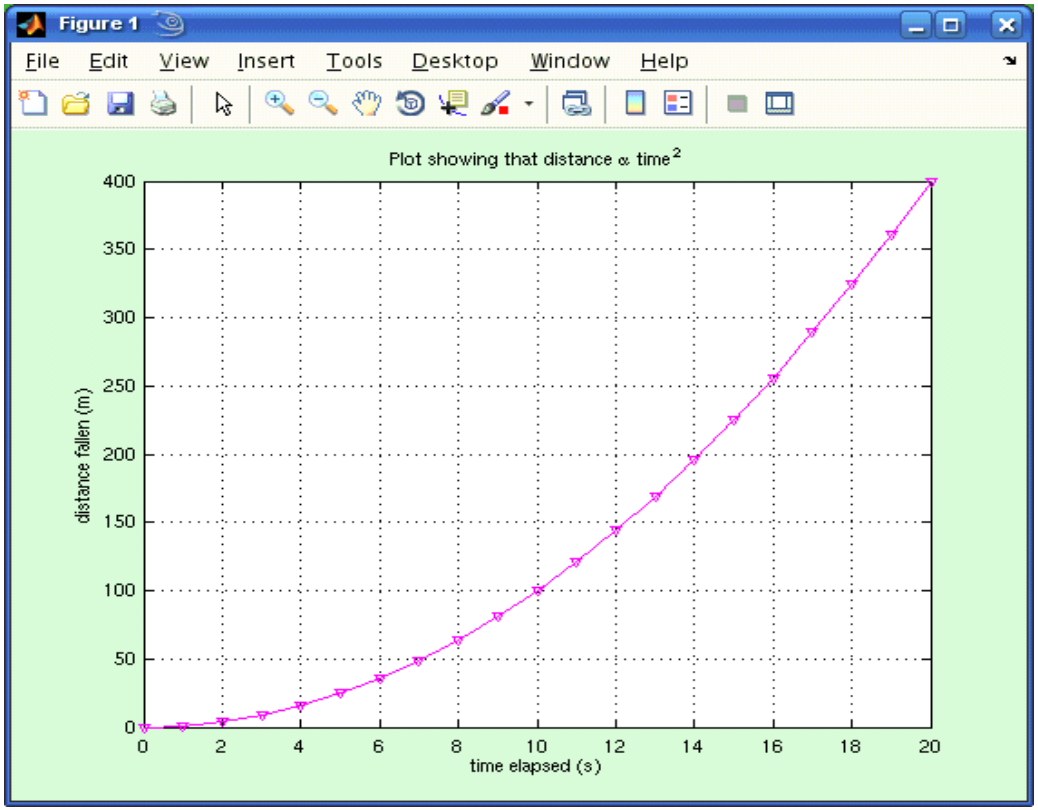
plot y against x using magenta triangles and connect with a line

Annotating your plots

```
>> xlabel('time elapsed (s)');
>> ylabel('distance fallen (m)')
>> title('Plot showing that distance \alpha time^2')
>> grid on
>>
```

xlabel
ylabel
title
grid on

Superscripts: ‘time^2’ => time^2
Subscripts: ‘SO_2’ => SO_2
Greek characters: \alpha => α



Logarithmic axes?

plot(x, y)	x linear,	y linear
semilogx(x, y)	x logarithmic,	y linear
semilogy(x, y)	x linear,	y logarithmic
loglog(x, y)	x logarithmic,	y logarithmic

Otherwise they work exactly the same.

MATLAB Graphics Object Hierarchy

Screen

- Figure1
  - Axes1 (xlabel, ylabel, title, tick marks, tick labels)
    - Graph1 (linestyle, legendlabel)
    - Graph2
    - …
  - Axes2
    - Graph1
    - …
- Figure2
  - Axes1
    - Graph1
    - Graph2
  - Axes2
    - Graph1
- …

figure

To create a new figure with no axes:
>> figure;

To highlight a figure that is already displayed (if it doesn't already exist, it will be created):
>> figure(2)

To get all the properties associated with a figure:
>> get(figure(2))

To get a particular property associated with a figure:
>> get(figure(1), 'Position')
[420 528 560 420]

To modify a particular property associated with a figure:
>> set(figure(1), 'Position', [100 100 560 420])

This particular example will just move where figure(1) is plotted on the screen.

To get a 'handle' for the current active figure window use **gcf**.
>> get(gcf, 'Position')
Will return the screen position of the current active figure window.

axes

New figures are created without a set of axes.

To get a 'handle' for the current active set of axes use **gca** (get current axes).
Example: get a list of all properties associated with current axes
>> get(gca)

>> get(gca, 'position')
This will return the screen position of the current active figure window, which by default is:
[0.13 0.11 0.775 0.815]
Format here is [xorigin yorigin xwidth yheight] in fractions of the figure window width.

To modify the position of the current axes within a figure:
>> set(gca, 'position', [0.2 0.3 0.6 0.4])
The axes would start 20% of the way across the screen, 30% of the way up,
and be 60% the screen width, and 40% the screen height.

An alternative syntax is just to call the axes command:

>> axes('position', [0.2 0.3 0.6 0.4]);
Either will create a figure if none already exists. Or modify the current set of axes on the current figure.

plot

plot will create a figure and a set of axes at the default position, if there is currently no figure.

Otherwise it will modify the current figure / current axes (this can be changed with '**hold on**').
So be careful not to overwrite other graphs!

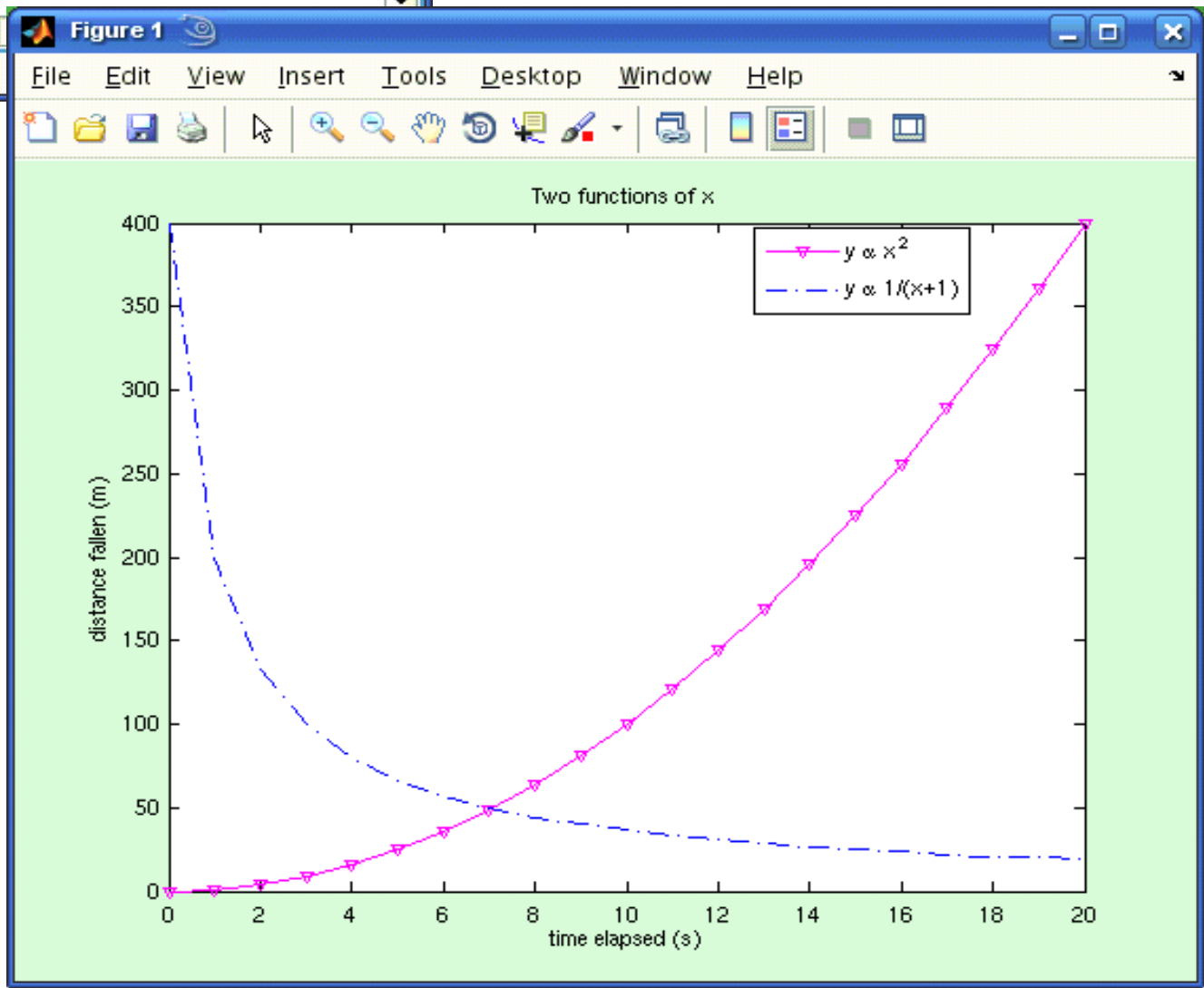
Multiple plots on a figure: hold on

```
>> grid off
>> y2 = 400./(x+1);
>> hold on
>> plot(x,y2,'b-.')
>> title('Two functions of x')
>> legend('y \alpha x^2', 'y \alpha 1/(x+1)')
>>
```

hold on "holds on" to graphs already in the current axes.
Normally they would be erased

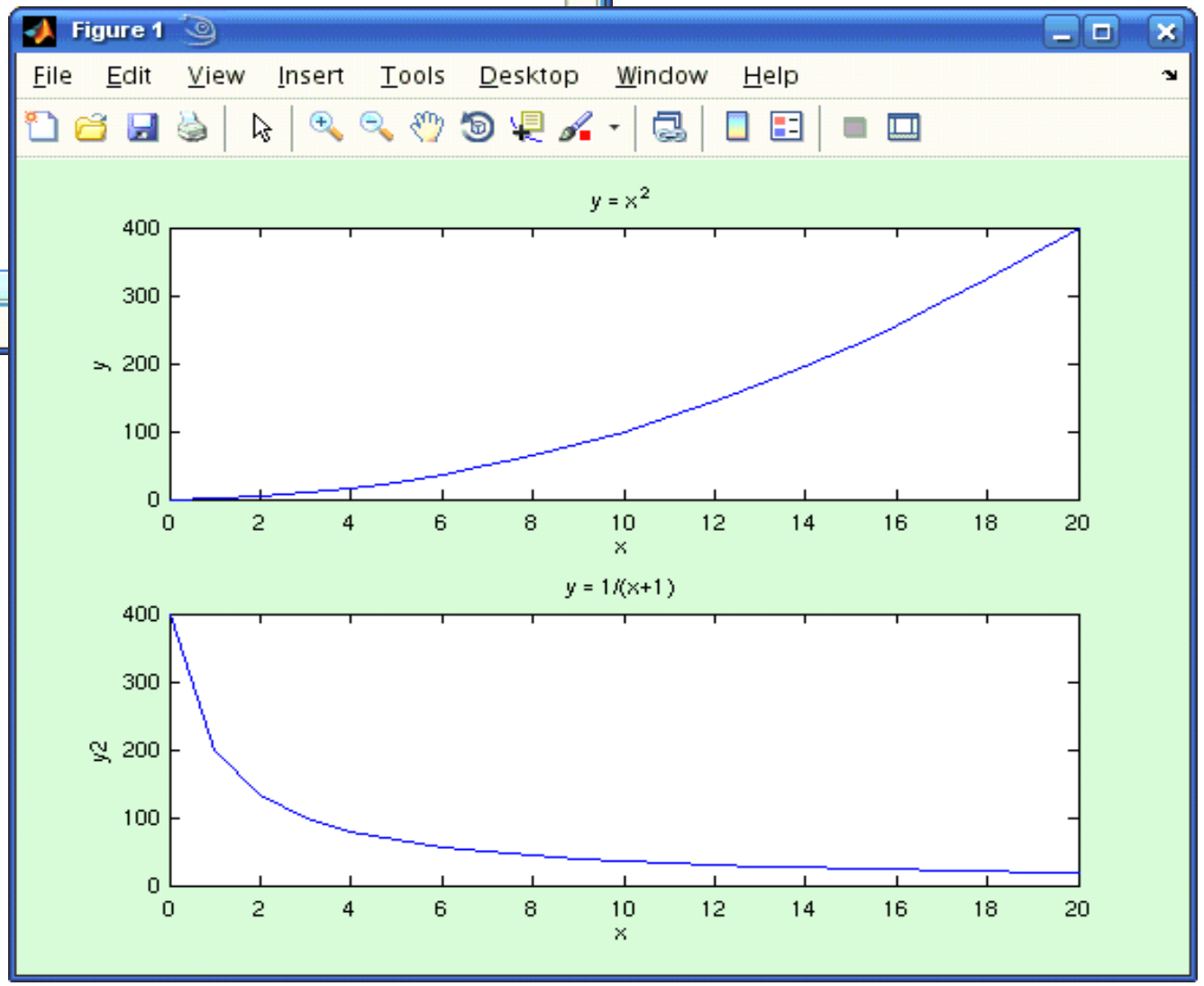
hold on
plot(x,y,'-.')
title
legend
hold off

If your graphs have very different scales, and you have just two, try **plotyy**



Multiple plots on a figure: subplot()

```
>> close all
>> figure
>> subplot(2,1,1), plot(x,y)
>> title('y = x^2')
>> xlabel('x')
>> ylabel('y')
>> subplot(2,1,2), plot(x,y2)
>> title('y = 1/(x+1)')
>> xlabel('x')
>> ylabel('y2')
```



close all
figure

subplot(M, N, plotnum) - an M x N array of plot axes

Multiple plots on a figure: axes()

Shortcuts How to Add What's New

Command Window

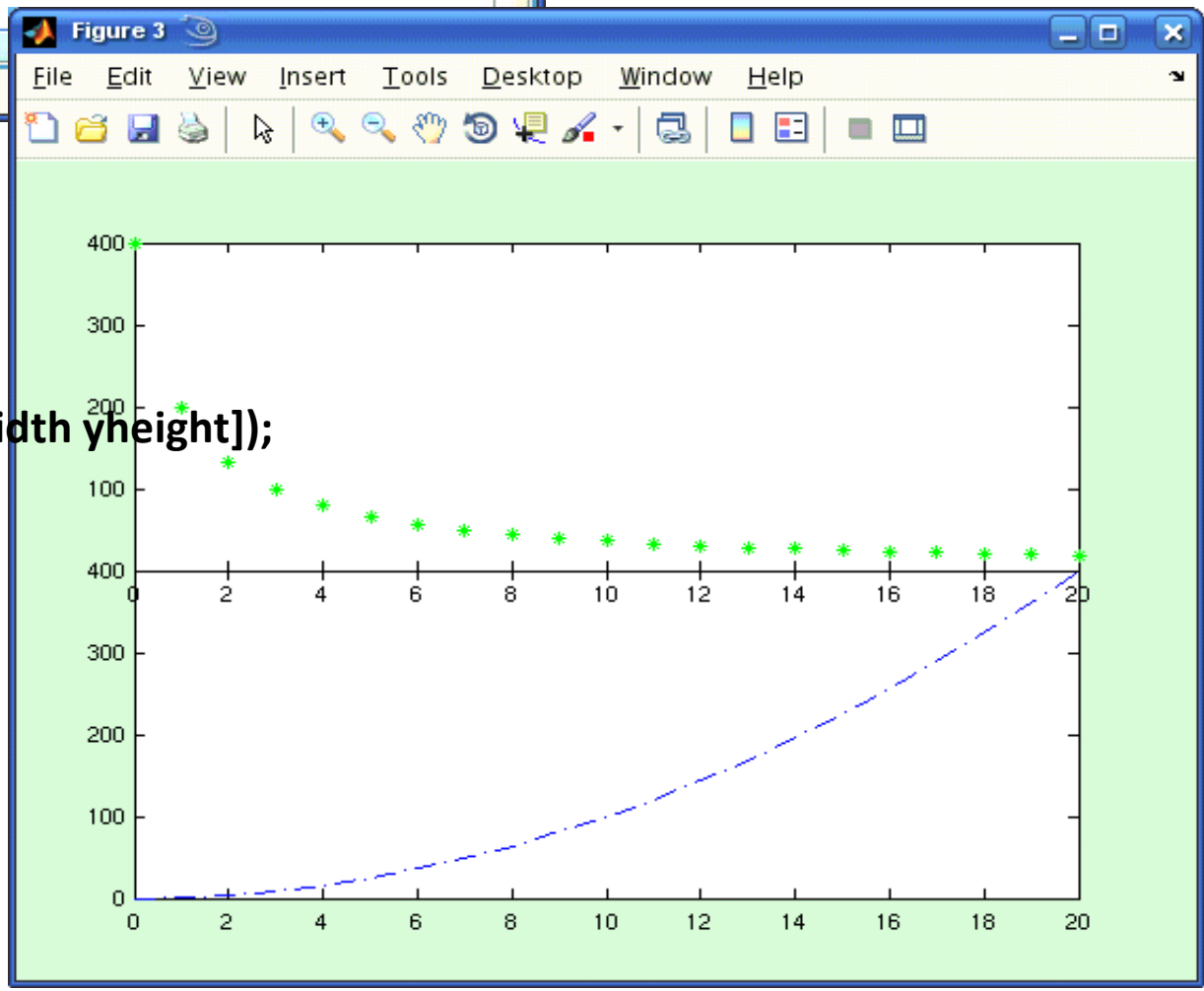
New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
>> figure
>> axes('position', [0.1 0.1 0.8 0.4])
>> plot(x,y,'b-.')
>> axes('position', [0.1 0.5 0.8 0.4])
>> plot(x,y2,'g*')
>>
```

Start

axes('position', [xorigin yorigin xwidth yheight]);
– for finer control than subplot

set(gca, 'XTickLabel', {})
- remove x tick labels



Other ways to call plot()

plot(y) - assumes x = [1:length(y)];

plot(x1, y1, x2, y2, …, xn, yn) - a way of plotting multiple graphs without using **hold on**

plot(x1, y1, s1, x2, y2, s2, …, xn, yn, sn) – as above, but override the default line styles.

Data range

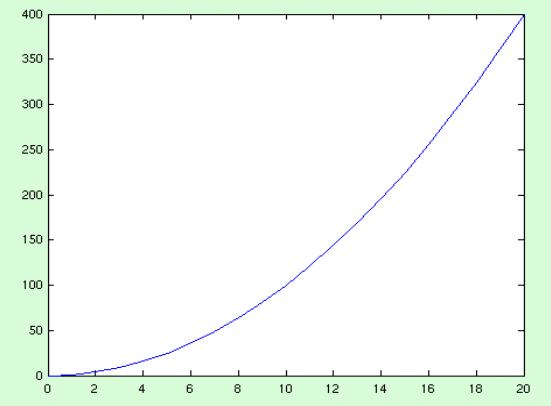
By default, the plot range to show all the data. To override the range of values on the x and y axes use:

>> **set(gca, 'XLim', [xmin xmax]);** % x-axis only
>> **set(gca, 'YLim', [ymin ymax]);** % y-axis only
>> **set(gca, 'XLim', [xmin xmax], 'YLim', [ymin ymax]);** % both axes

Adding text

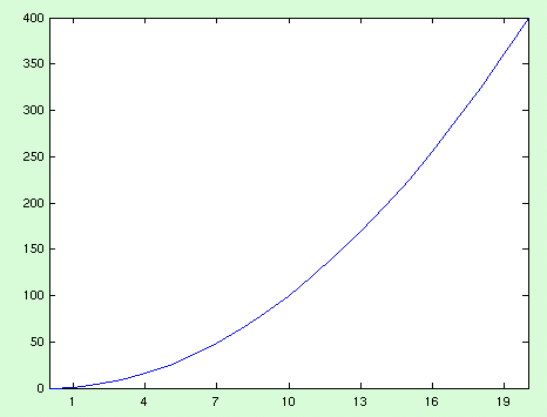
To add text at the position xpos, ypos to the current axes use:
>> **text(xpos, ypos, 'some_string')**;
Remember you can use the sprintf variable.
>> **text(2.3, 5.1, sprintf('station %s',station{stationNum});**

Changing tick marks and tick labels

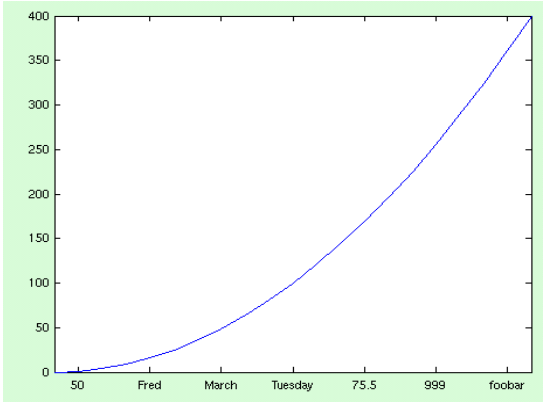



```
>> get(gca, 'XTick')
ans =
     0     2     4     6     8    10    12    14    16    18    20
```

set(gca, 'XTick', 1:3:22)



set(gca, 'XTickLabel', {50, 'Fred', 'March', 'Tuesday', 75.5, 999, 'foobar'})



datetick: plotting timeseries data

datetick('x', dateform)

If your x-vector is a list of dates/times, and it's in MATLAB's datenum format, you can use the datetick() function to label your x-axis conveniently.

It just uses the techniques we've seen to change tick marks and tick labels.

dateform can be a number from 0 to 31, or it can be a string like 'yyyy-mm-dd HH:MM'.

datetick('x') will just try to use what it thinks is the best dateform for your data range.

datenum() returns the number of days since 1st January in the year 0 AD. (Excel dates and times are similar except Excel uses a different origin. Unix on the other hand uses seconds since 1970 rather than days since a particular date: nevertheless, conversion is trivial).

datestr() is used to generate a human-readable string from an array of dates/times in datenum format.

dateform codes

DATEFORM number	DATEFORM string	Example
0	'dd-mmm-yyyy HH:MM:SS'	01-Mar-2000 15:45:17
1	'dd-mmm-yyyy'	01-Mar-2000
2	'mm/dd/yy'	03/01/00
3	'mmm'	Mar
4	'm'	M
5	'mm'	03
6	'mm/dd'	03/01
7	'dd'	01
8	'ddd'	Wed
9	'd'	W
10	'yyyy'	2000
11	'yy'	00
12	'mmmyy'	Mar00
13	'HH:MM:SS'	15:45:17
14	'HH:MM:SS PM'	3:45:17 PM
15	'HH:MM'	15:45
16	'HH:MM PM'	3:45 PM
17	'QQ-YY'	Q1-96
18	'QQ'	Q1
19	'dd/mm'	01/03
20	'dd/mm/yy'	01/03/00
21	'mmm.dd,yyyy HH:MM:SS'	Mar.01,2000 15:45:17
22	'mmm.dd,yyyy'	Mar.01,2000
23	'mm/dd/yyyy'	03/01/2000
24	'dd/mm/yyyy'	01/03/2000
25	'yy/mm/dd'	00/03/01
26	'yyyy/mm/dd'	2000/03/01
27	'QQ-YYYY'	Q1-1996
28	'mmmyyyy'	Mar2000
29 (ISO 8601)	'yyyy-mm-dd'	2000-03-01
30 (ISO 8601)	'yyyymmddTHHMMSS'	20000301T154517
31	'yyyy-mm-dd HH:MM:SS'	2000-03-01 15:45:17

datenum(): MATLAB's way to store dates and times

datenum() returns the day number (and fractional day number) in the calendar starting 1st January in the year 0 AD.

Excel dates and times are similar except Excel uses the origin 1st January 1900. But you normally ask Excel to format those cells with a particular date/time format, so you don't see the raw numbers. In MATLAB, datenum gives those raw numbers.

To convert from Excel day-numbers to MATLAB datenum format:

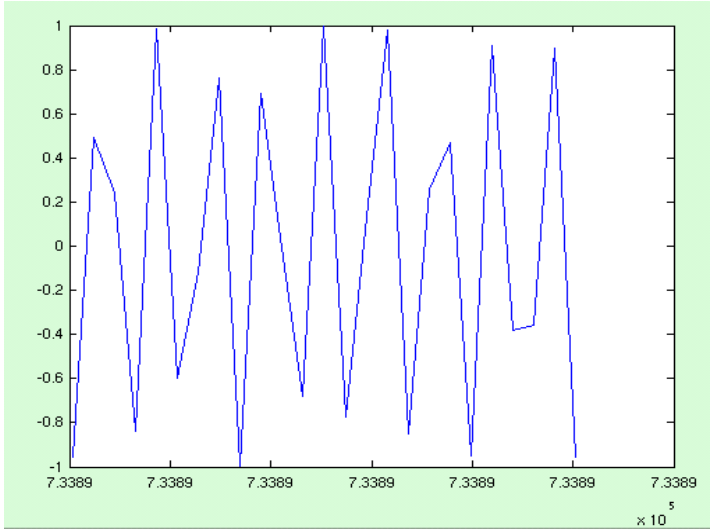
```
mtime = etime + datenum(1900, 1, 1);
```

Call it like:

```
datenum(YYYY, MM, DD)
datenum(YYYY, MM, DD, hh, mi, ss)
datenum('2009/04/29 18:27:00')
```

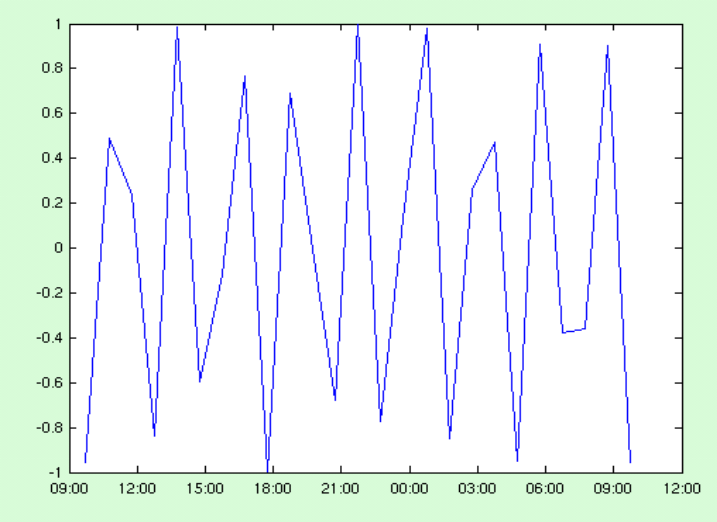
Remember to use vectorisation:

```
redoubtEventTimes = {'2009/03/22 22:38'; '2009/03/23 04:11'; '2009/03/23 06:23'}
dnum = datenum(redoubtEventTimes);
% result is a 3 x 1 vector of datenums.
```



Not very useful to plot against datenum – just a number like 733000

datetick('x', 15); % HH:MM



datestr()

datestr(array, dateform) is used to generate a human-readable string from an array of dates/times in datenum format.

```
>> lectureTime = datenum(2009, 4, 29, 12, 30, 0)
733890.5208
>> datestr(lectureTime, 30)
20090427T123000
>> datestr(lectureTime, 31)
2009-04-29 12:30:00
>> datestr(lectureTime, 'mm/dd/yyyy')
04/29/2009
```

Seismology - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://www.avo.alaska.edu/admin/seismology/

matlab anova

openSUSE Getting Started Latest Headlines

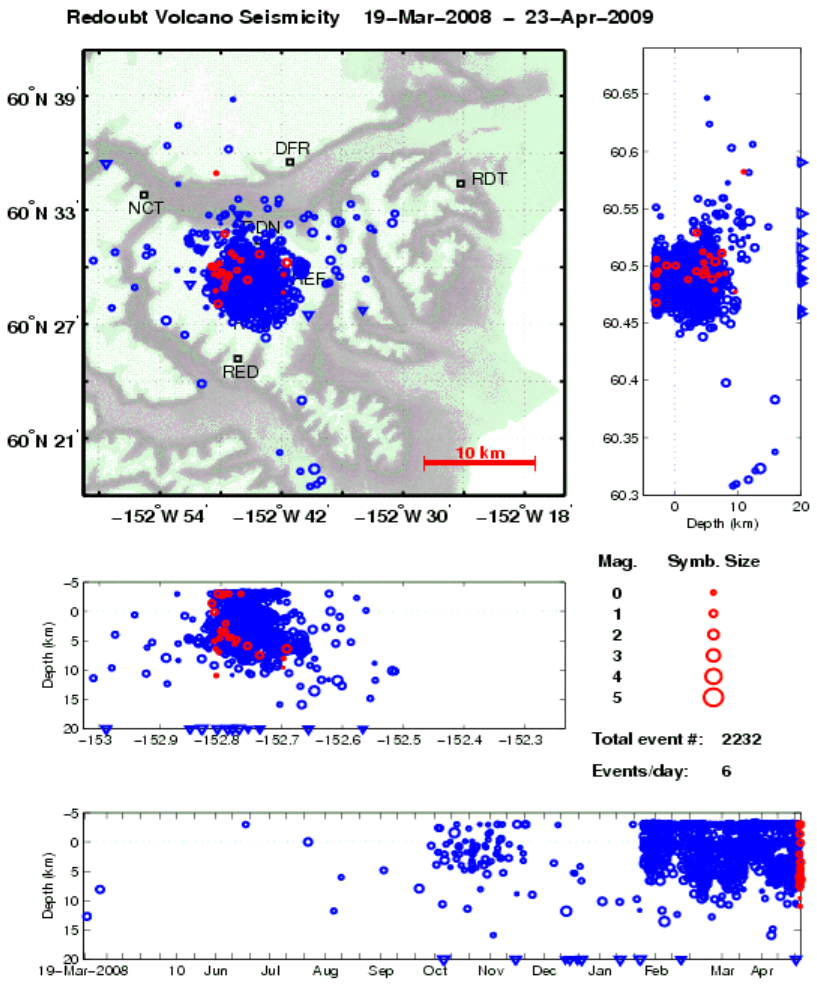
Mail :: Inbox: Re: Programming ... | Seismology | Physics 693 | Fall 2008 Syllabus

Menu | Login | 0 items | 2009-04-22 2246

Short-term Seismicity

Long-term Seismicity

- Wrangell (more)
- Cook Inlet
  - Strandline
  - Spurr (more)
  - Redoubt (more)
  - Iliamna (more)
  - Augustine (more)
  - Fourpeaked
- Katmai Volcanic Cluster
  - Snowy
  - Griggs
  - Katmai
  - Novarupta/Trident
  - Martin/Mageik
- Alaska Peninsula
  - Peulik
  - Aniakchak (more)
  - Veniaminof (more)
- Cold Bay
  - Pavlof (more)
  - Dutton (more)
- Unimak Island
  - Shishaldin (more)
  - Westdahl (more)
- Western Fox Islands
  - Akutan (more)
  - Makushin (more)
  - Okmok (more)
  - Umnak Is.
- Eastern Andreanof Islands
  - Cleveland
  - Korovin (more)
- Middle Andreanof Islands
  - Great Sitkin (more)
  - Kanaga (more)
- Western Andreanof Islands
  - Tanaga (more)



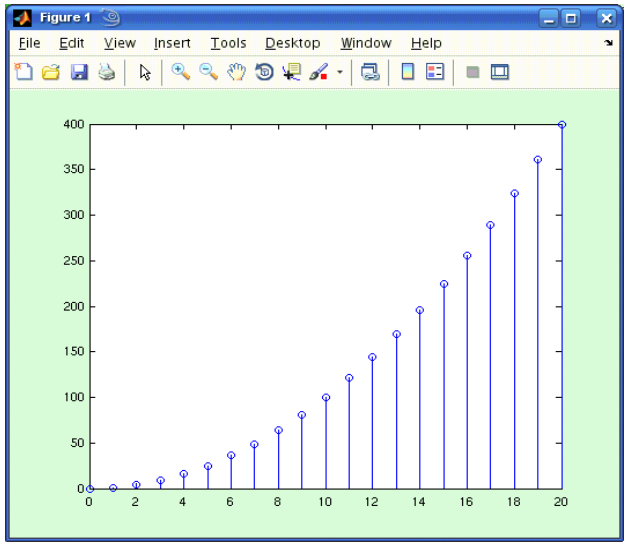
- 5 axes
- text(lon, lat, 'NCT')
- set(gca, 'XTick', [-152.9 -152.8]);
- set(gca, 'XTickLabel', {'-152^oW 54', …});
- datetick('x');

```
% change symbol is depth > 20km
symbol='o';
if (depth>20)
  symbol='v';
  depth=20;
end

% change symbol size
% according to magnitude

line(x, y);
axis square
```

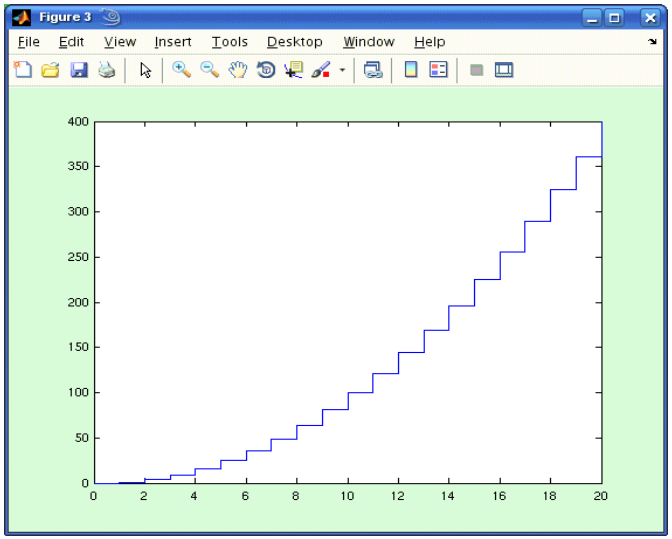
Done

Other simple 2D plot types

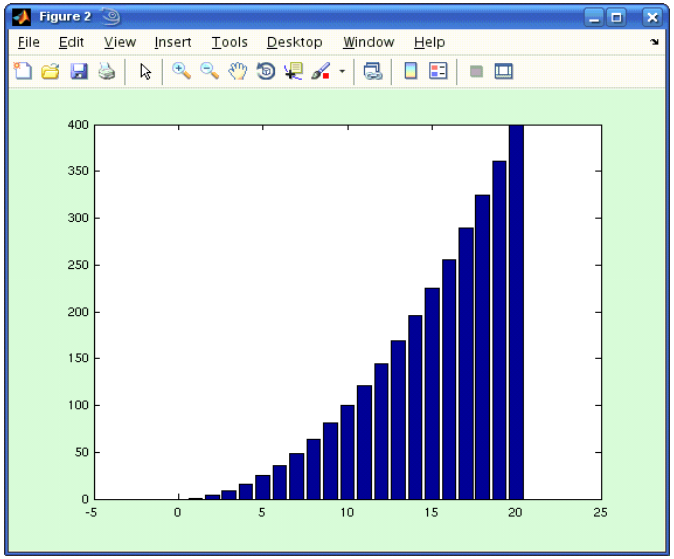
stem(x, y)



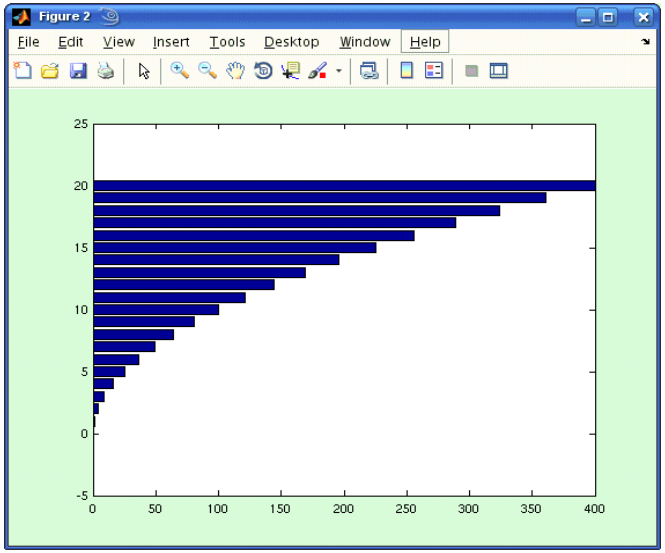
stairs(x,y)



bar(x,y)



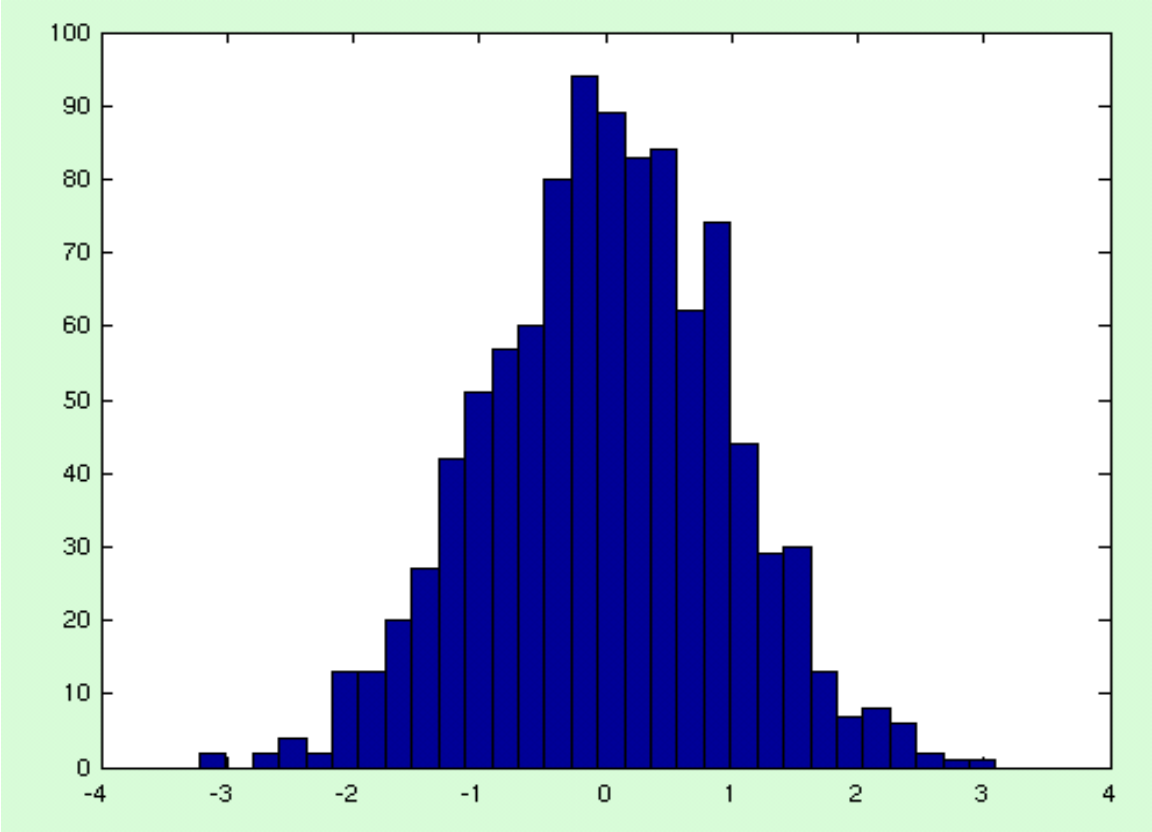
barh(x,y)



hist – for plotting histograms

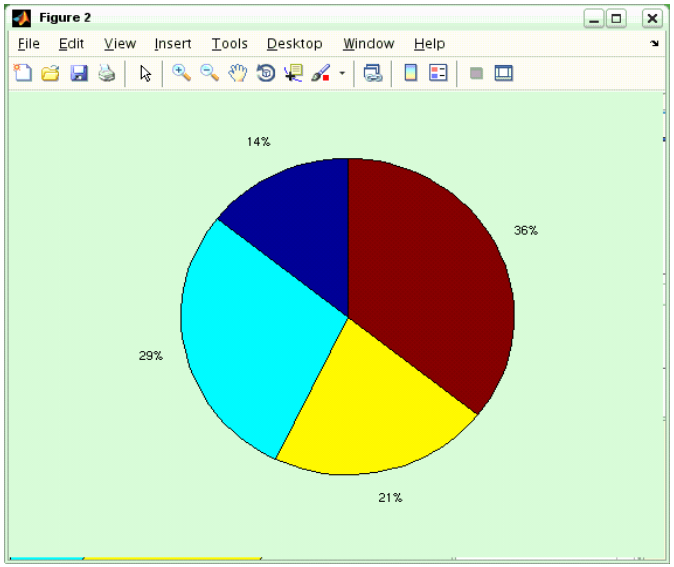
create a histogram of 1000 random data points from a normal distribution

```
>> r=randn(1000, 1);
>> hist(r, 30)
```

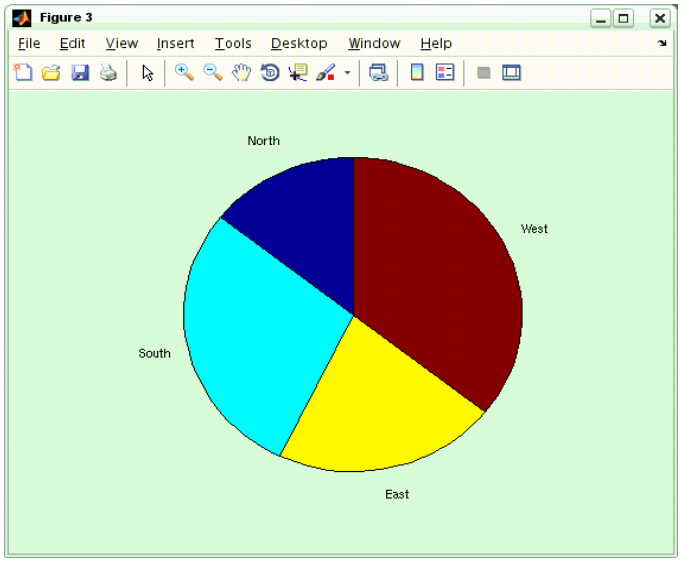


Pie Charts: pie()

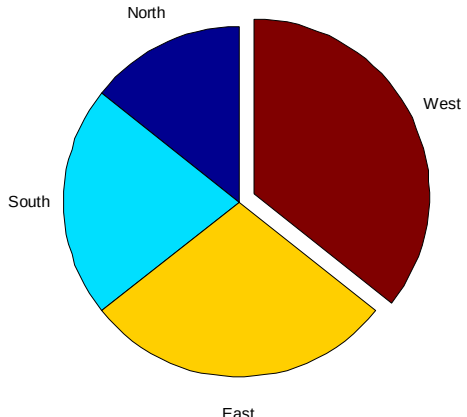
```
x = [2 3 4 5]; pie(x)
```



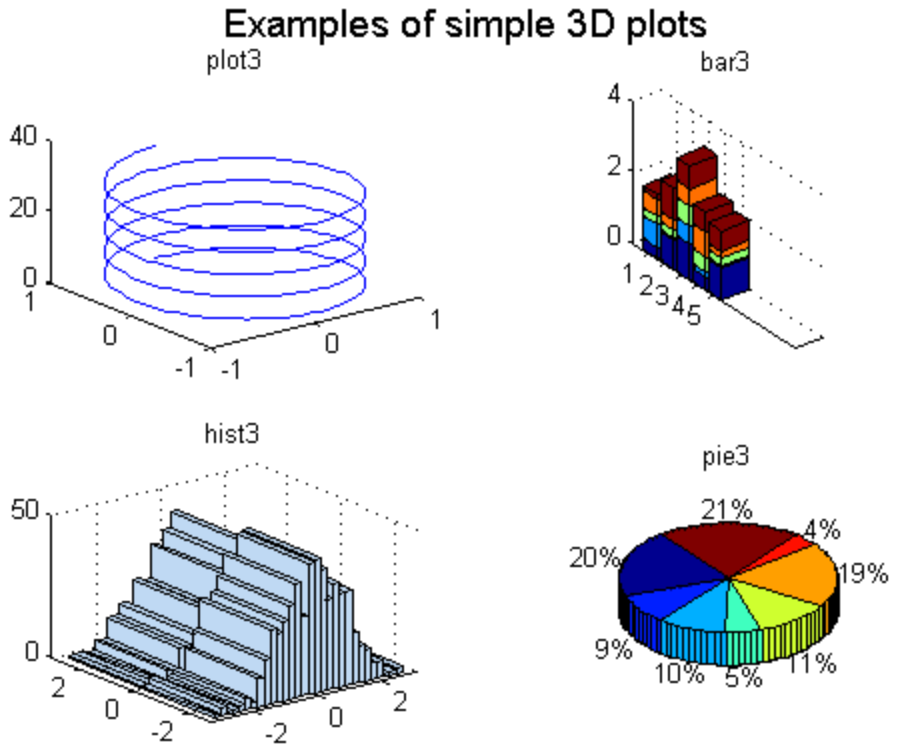

```
labels = {'North', 'South', 'East', 'West'};
pie(x, labels)
```



PIE(X,EXPLODE) is used to specify slices that should be pulled out from the pie. The vector EXPLODE must be the same size as X. The slices where EXPLODE is non-zero will be pulled out.




```
ex = [0 0 0 1];
pie(x, ex, labels);
```

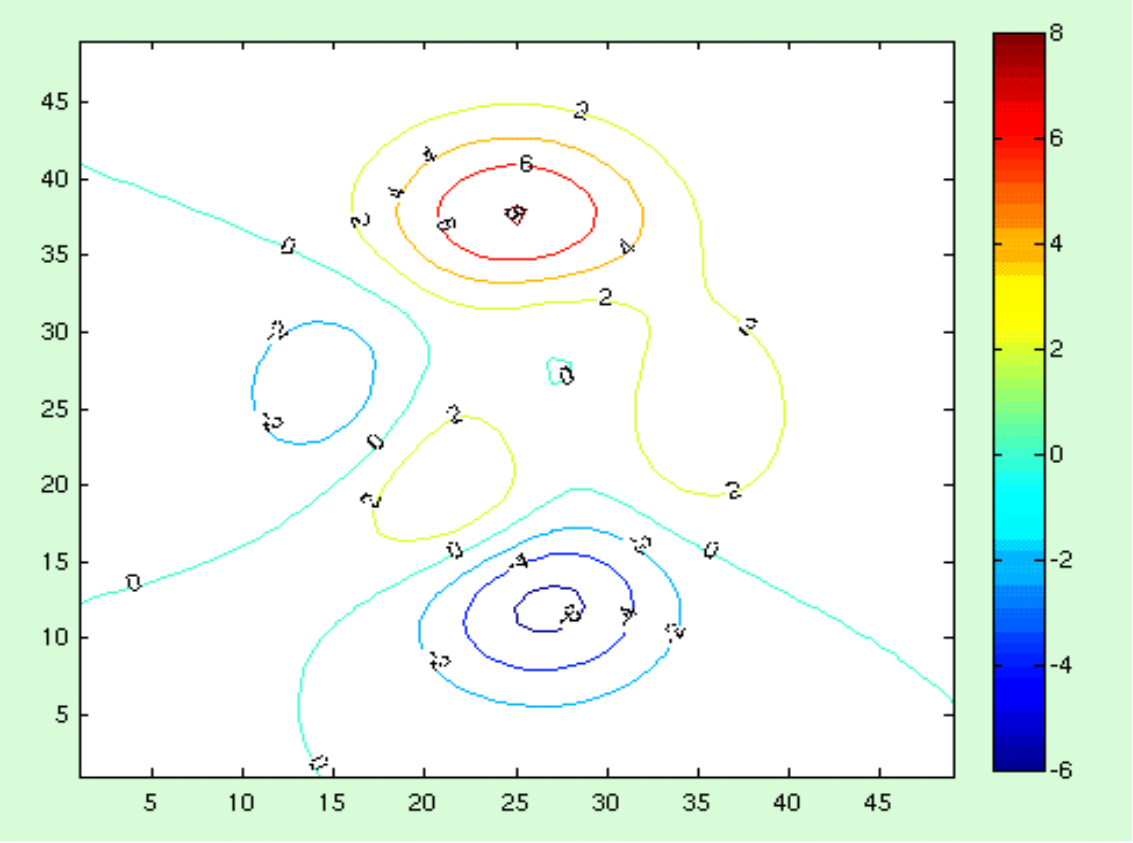



```
>> subplot(2,2,1), plot3(sin(t),cos(t),t); title('plot3')
>> subplot(2,2,2),bar3(rand(5),'stacked'), title('bar3')
>> subplot(2,2,3),hist3(randn(1000,2), [30 2]), title('hist3')
>> subplot(2,2,4),pie3(rand(8,1)), title('pie3')
>> suptitle('Examples of simple 3D plots')
```

Here we have a 2 x 2 array of subplots. We give a ‘super title’ to them all with the suptitle() function.
We used the 3D forms of the plot, bar, hist and pie commands.

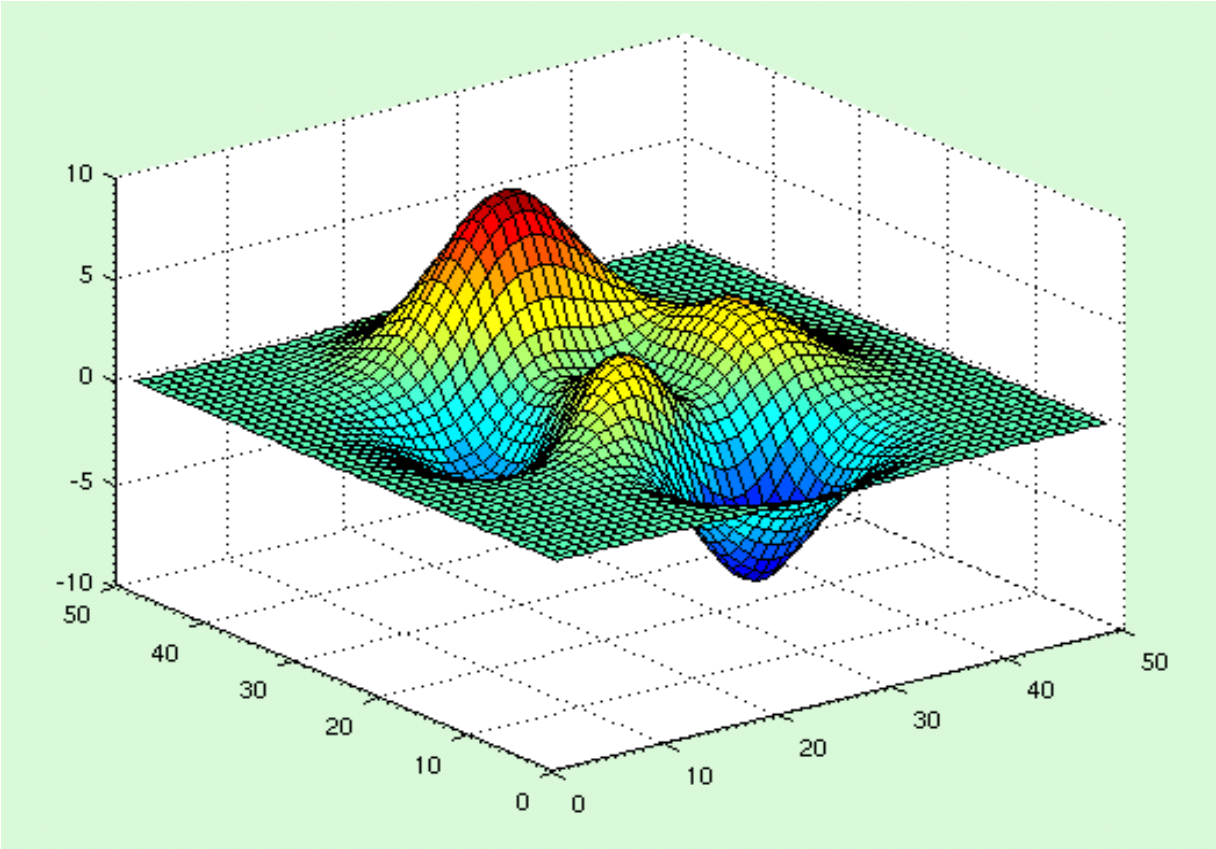
Contour plots: contour()

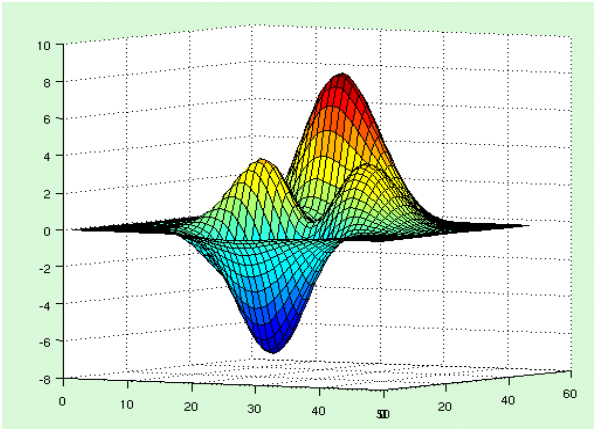
[c,h] = contour(peaks); clabel(c,h), colorbar



surf

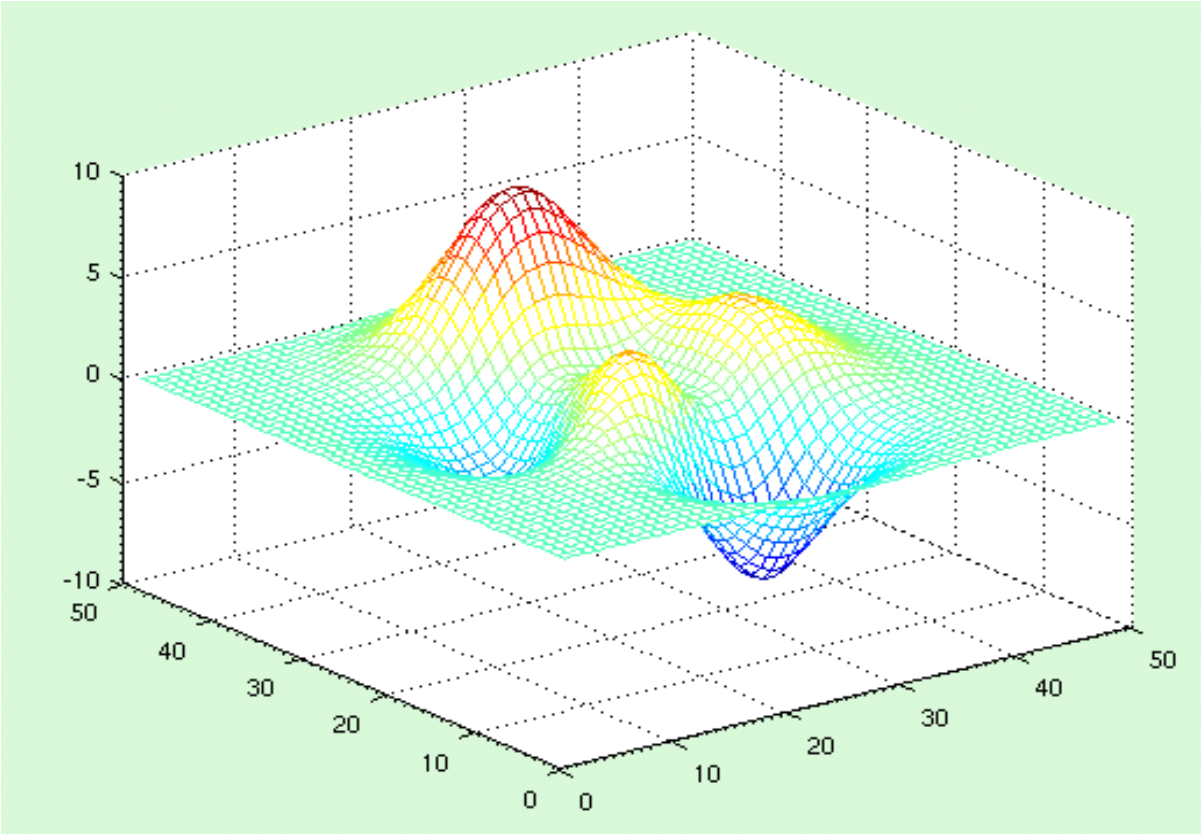
surf(peaks)





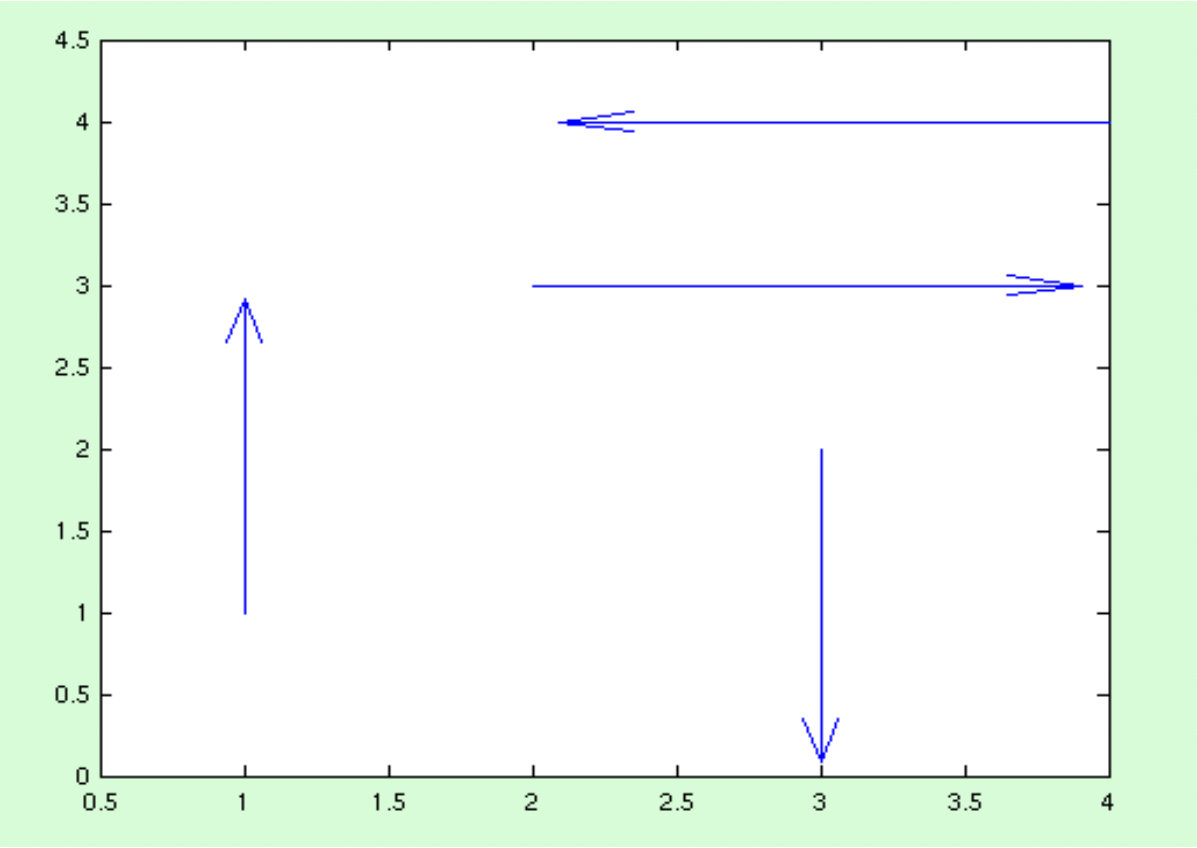
mesh

mesh(peaks) – creates a wireframe



quiver – for plotting vectors

```
>> x = [1 2  3 4];
>> y = [1 3 2 4];
>> u = [0 0.5 0 -0.5];
>> v = [0.5 0 -0.5 0];
>> quiver(x,y,u,v)
```

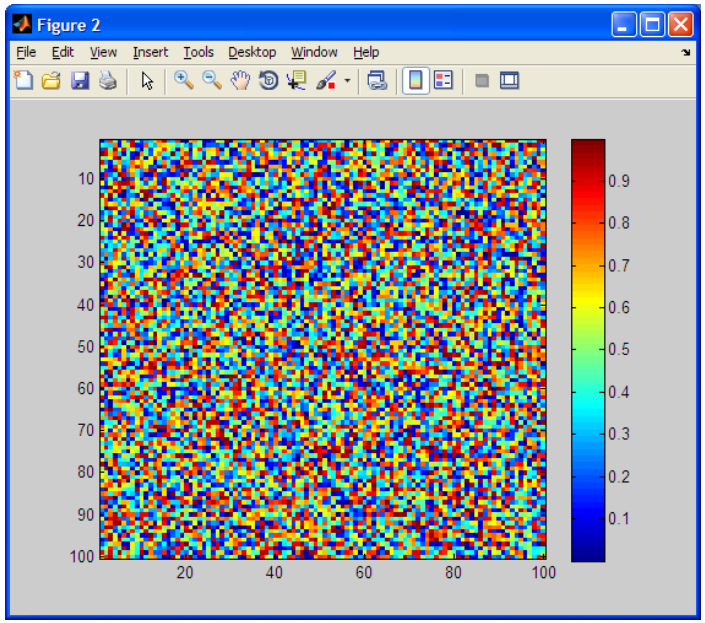


Graphical representation of an array: image(), imagesc(), colorbar

An array can be plotted, using different colours to represent different values.

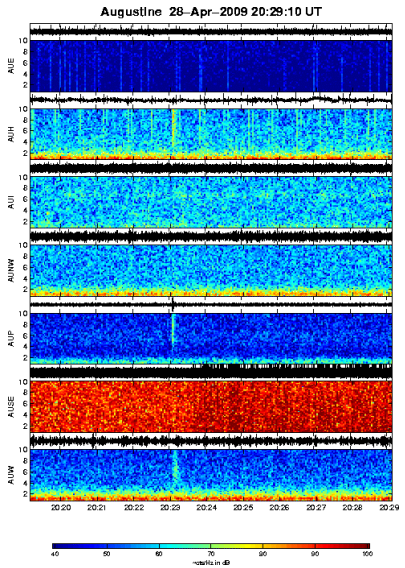
Example:

```
>> a = rand(100, 100);  % 100 x 100 array of random numbers from 0 to 1
>> imagesc(a);
>> colorbar;
```



Spectrograms, on the AVO internal webpage, are created in this way, except the array is generated using the **specgram()** command.

There are 15 different axes on this plot.

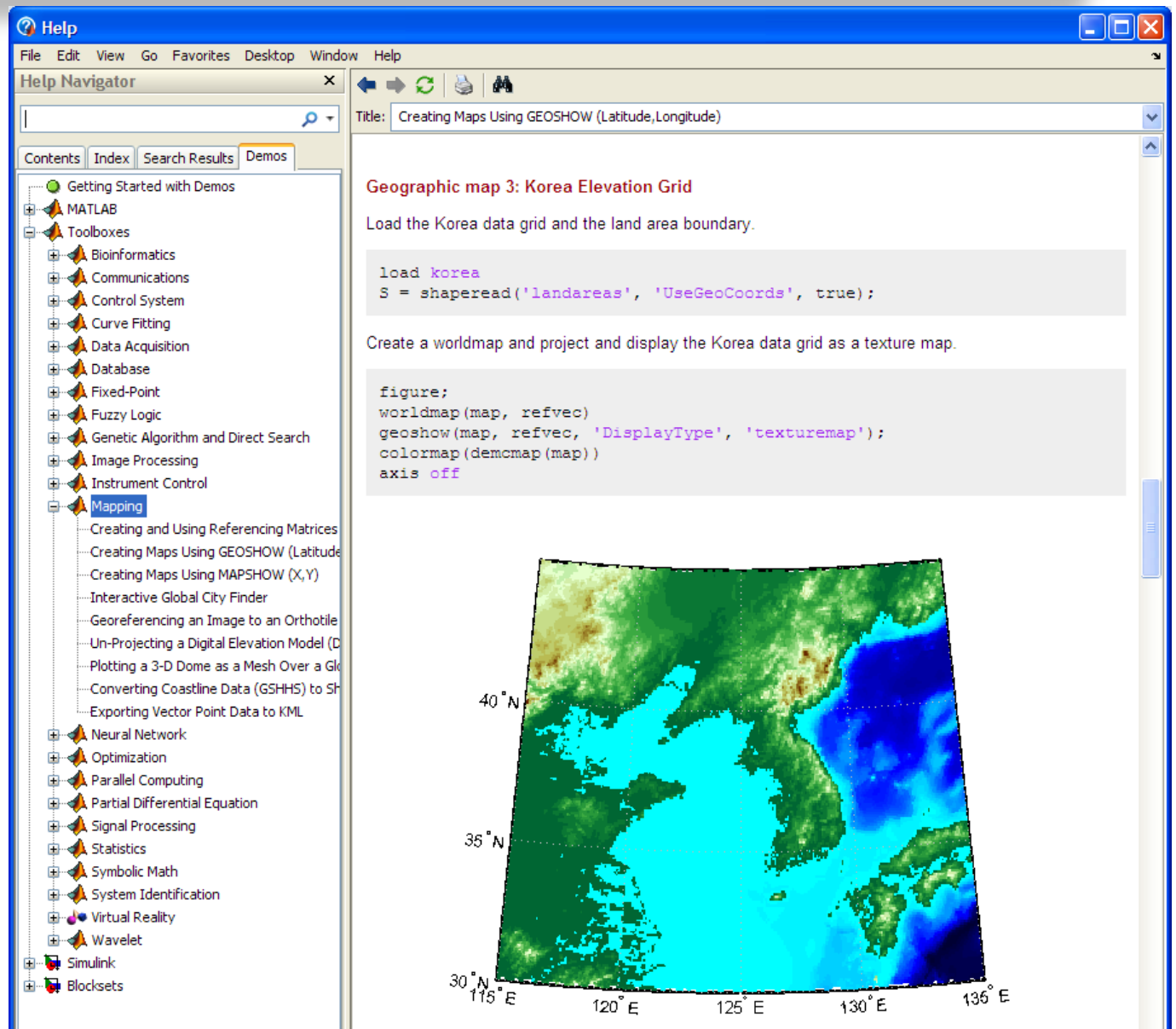


MATLAB does maps too! The mapping toolbox

>> help map
>> mapdemos

Can write KML:
>> help kmlwrite

Alternative to
Using GMT



Graphics files I/O

Writing your plots to image files

Loading in image files

Saving your figures to image files: print()

print -f1 -dpng myplotfilename.png - dpng means device PNG

print('-f1', '-dpng', '-r200', 'myplotfilename.png') - functional form

-r200 means print with resolution 200 dots per inch (use lower number for small plot)

-f2 means print figure 2

Devices include:

ps, psc, ps2, psc2	- Postscript (c = colour, 2 = level 2)
eps, epsc, eps2, eps2	- Encapsulated Postscript (c = colour, 2 = level 2)
ill	- Adobe Illustrator format
jpeg90	- JPEG with quality 90 (can be 01 to 99)
tiff	- TIFF
png	- PNG

Example:

say you have (numberOfPlots) figures and you want to save all of them as level-2 color encapsulated postscript files with names like myplot1.eps, myplot2.eps:

```
for plotNum = 1 : numberOfPlots
    print('-depsc2', sprintf('-f%d',plotNum), '-r70', sprintf('myplot%d.eps',plotNum) );
end
```

For plotNum = 2, the print line would evaluate to:

```
print('-depsc2', '-f2', '-r70', 'myplot2.eps')
```

You can also capture a figure window with:
>> print –dmeta
on a Windows system, and paste it into your document. It does the same thing as ALT-PRT SC.

Loading a graphics file into MATLAB

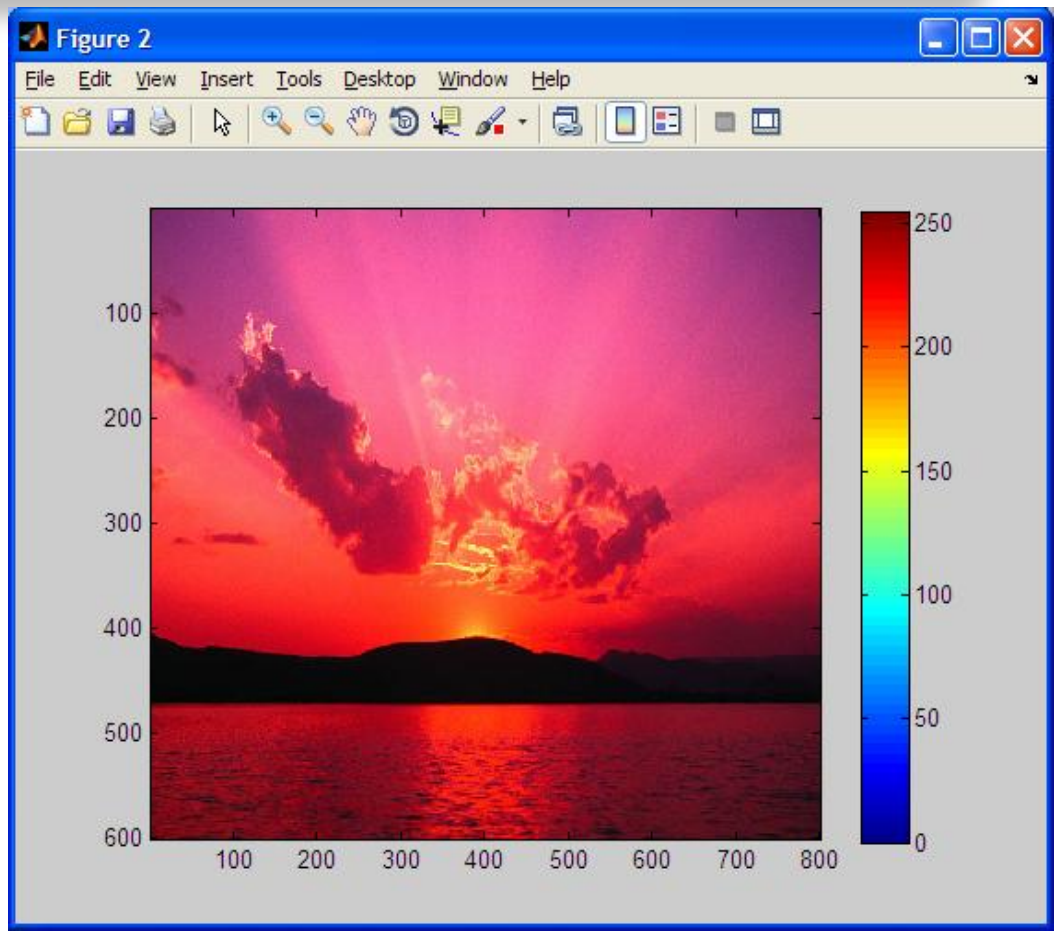
Load a raster graphics file into a numerical array with:
A = imread('filename', 'format')

Plot it in the current axes with:
image(A);

Add a colorbar with:
colorbar;

Example:

```
>> A = imread('Sunset.jpg');
>> size(A)
   600 800 3
>> image(A);
>> colorbar
```



Alter it however you want (add labels, title, text, change position, tick marks).

MATLAB file I/O

Getting your data into MATLAB and getting it out again

Text and Binary files

Text files:

- **Files you can read with a text editor, or with the cat, more or less commands at a Unix prompt.**
- **Might not be able to understand the data they encode.**
- **Advantage: Human readable.**
- **We will assume all text files are in ASCII format, which represents English alphabet, the digits 0 – 9 and other symbols you see on your keyboard. ASCII(65) = 'A'; ASCII(66) = 'B'; ASCII(97) = 'a';**

Binary files:

- **Garbage if you try to open them with a text editor, or with the cat, more or less commands.**
- **Advantage: Compression.**

Example:

Take the number 65535. It is equal to 2^16 – 1.
This can be stored in 2 bytes in a binary file. In a text file it needs at least 5 bytes.

Some binary files contain a mixture of text and binary data, e.g. the SEISAN data format. It contains metadata in plain text, but waveform data in binary format.

MATLAB binary files:
load
save

MAT files (MATLAB binary files)

MAT files are MATLAB binary files. Only MATLAB can read/write them. They are useful for storing (workspace) variables, so you can reload them later. Use **save** and **load**.

Examples:

>> save foobar.mat

% saves all workspace variables to the file foobar.mat (.mat extension is optional)

>> save foobar.mat x y

% saves only the workspace variables x and y to the file foobar2.mat

>> save foobar.mat sta*

% saves all workspace variables that begin with the letters 'sta' (* is a wildcard)

>> load foobar.mat % loads the file foobar.mat

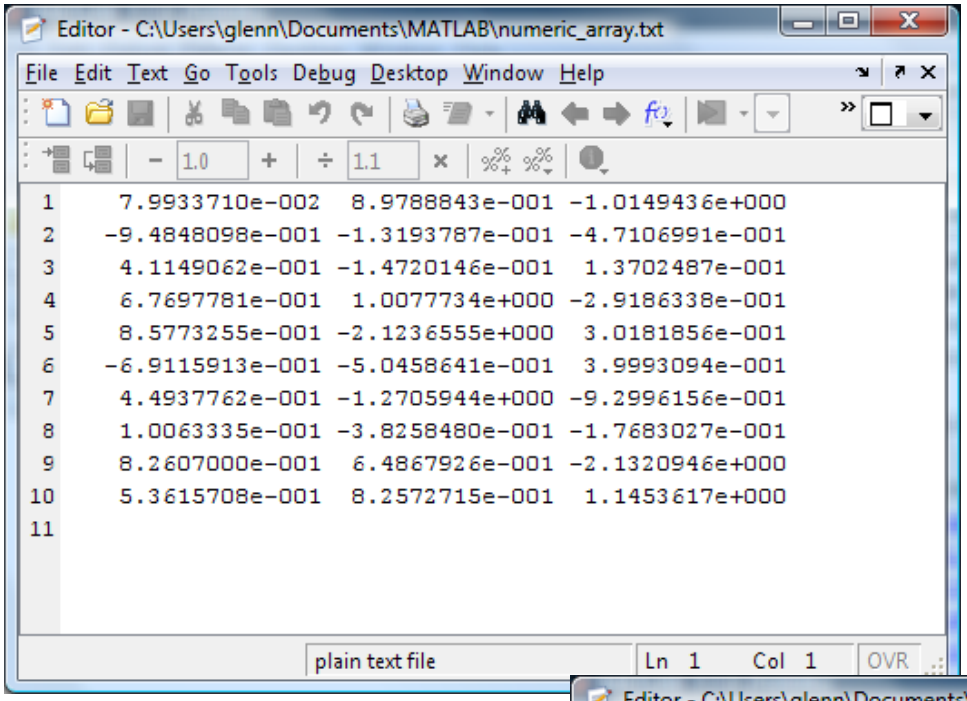
>> load foobar x % loads only the variable x from foobar.mat

save & load fully support numeric arrays, strings, cell arrays and structs for MAT files.

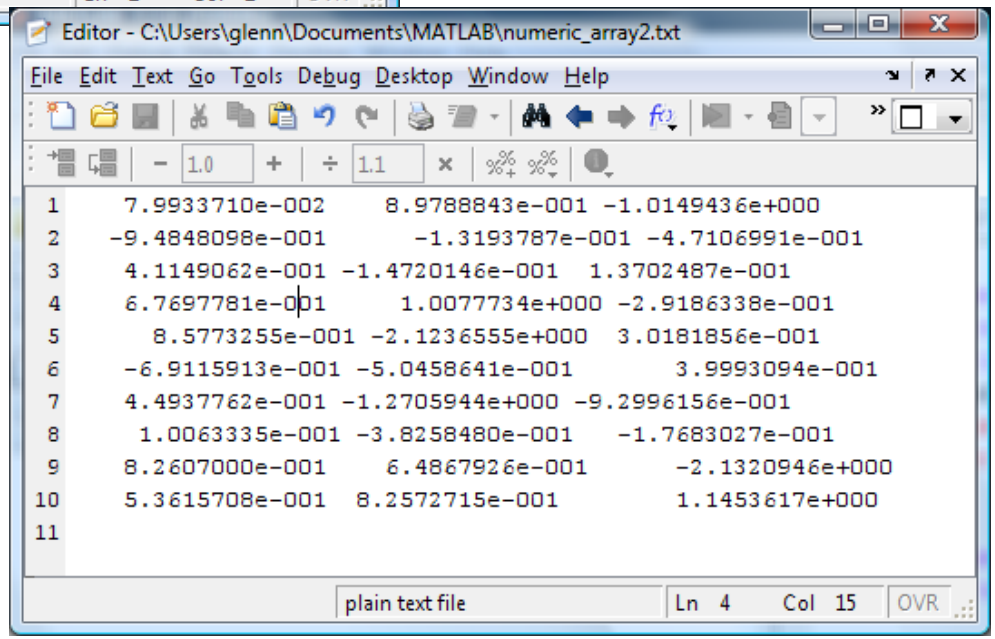
Text file I/O:

load
dlmread/csvread
importdata
fopen/textscan/fclose
fopen/fgetl/fclose
fopen/fscanf/fclose

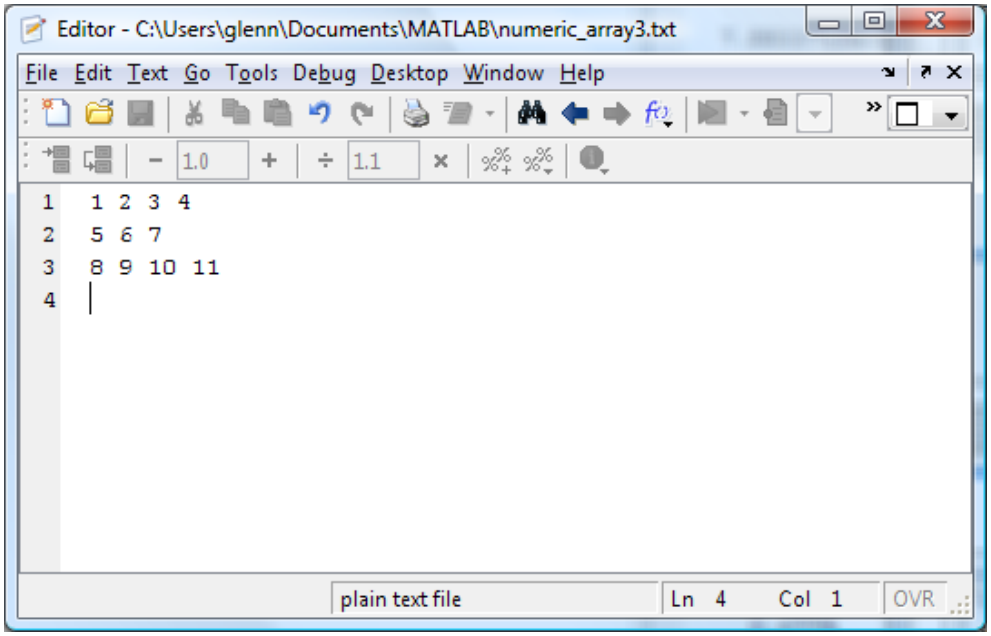
save
dlmwrite/csvwrite
fopen/fprintf/fclose




```
>> a=load('numeric_array.txt')

a =

    0.0799    0.8979   -1.0149
   -0.9485   -0.1319   -0.4711
    0.4115   -0.1472    0.1370
    0.6770    1.0078   -0.2919
    0.8577   -2.1237    0.3018
   -0.6912   -0.5046    0.3999
    0.4494   -1.2706   -0.9300
    0.1006   -0.3826   -0.1768
    0.8261    0.6487   -2.1321
    0.5362    0.8257    1.1454
```



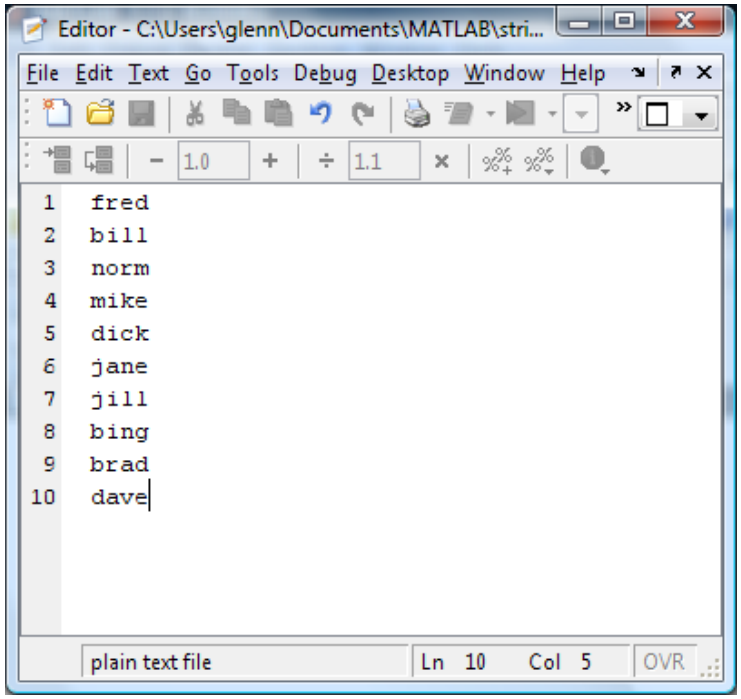

```
>> a2=load('numeric_array2.txt')

a2 =

    0.0799    0.8979   -1.0149
   -0.9485   -0.1319   -0.4711
    0.4115   -0.1472    0.1370
    0.6770    1.0078   -0.2919
    0.8577   -2.1237    0.3018
   -0.6912   -0.5046    0.3999
    0.4494   -1.2706   -0.9300
    0.1006   -0.3826   -0.1768
    0.8261    0.6487   -2.1321
    0.5362    0.8257    1.1454
```



But as soon as it no longer has the same number of numbers on each row, it is no longer a valid array, and it wont load.

```
>> a2=load('numeric_array3.txt')
??? Error using ==> load
Number of columns on line 1 of ASCII file numeric_array3.txt
must be the same as previous lines.
```

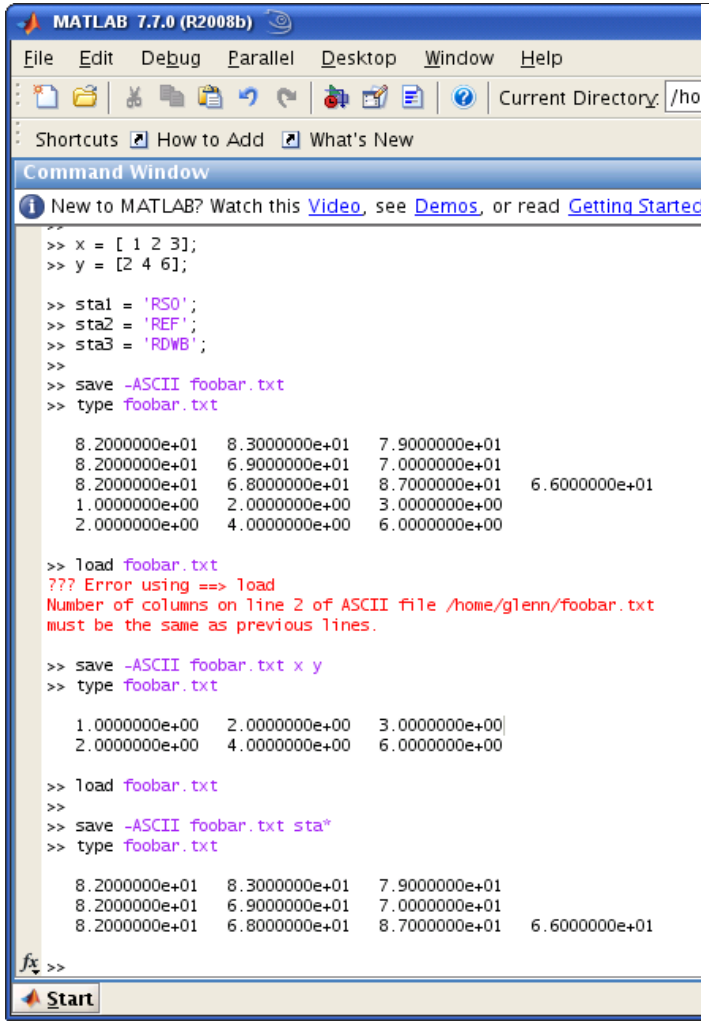
load() wont work at all with alphabetic characters




```
s=load('string_array.txt')
??? Error using ==> load
Unknown text on line number 1 of ASCII file string_array.txt
"free".
```

save/load: numeric ASCII (text) files

Suggested file extension is .dat or .txt.



save -ASCII filename [list of variables]

For numeric variables only
Will turn your strings into ASCII sequences.

load will only work for numerical arrays stored in a file.

Expects same number of columns on each row of an input file.

Neither of them will work with cell arrays or structs.

Example:

```
clear all;  x = ones(10, 1); y = randn(10, 1);
save foobar.txt –ASCII x y
clear all;
a=load('foobar.txt');
% a will be an array containing x and y
```

dlmread()

dlmread() is for reading a delimited numeric ASCII text file.

A = dlmread('filename.txt', 'delimiter')

For comma-separated-variables (CSV text files) there is a special command:

A = csvread('filename.txt')

But it just calls:

A = dlmread('filename.txt', ',');

So we'll ignore it.

dlmread() can be used to load numeric_array3.txt, padding each row with zeros as necessary:

>> a3=dlmread('numeric_array3.txt')

a3 =

1	2	3	4
5	6	7	0
8	9	10	11

Although **dlmread()** gets us around the restriction of having the same number of numbers of each row, it wont help us to load any non-numeric data. Nor will it work if you have different delimiters within the same input file.

importdata()

Our luck improves considerably with **importdata()**
A = importdata('filename.txt', 'delimiter')

It works without any difficulty for any of the text files we've seen so far:

```
>> a=importdata('numeric_array2.txt')

a =

    0.0799    0.8979   -1.0149
   -0.9485   -0.1319   -0.4711
    0.4115   -0.1472    0.1370
    0.6770    1.0078   -0.2919
    0.8577   -2.1237    0.3018
   -0.6912   -0.5046    0.3999
    0.4494   -1.2706   -0.9300
    0.1006   -0.3826   -0.1768
    0.8261    0.6487   -2.1321
    0.5362    0.8257    1.1454
```

It even loads string_array.txt into a cell array!

```
>> a=importdata('numeric_array3.txt')

a =

    1    2    3    4
    5    6    7  NaN
    8    9   10   11

>> s=importdata('string_array.txt')

s =

    'fred'
    'bill'
    'norm'
    'mike'
    'dick'
    'jane'
    'jill'
    'bing'
    'brad'
    'dave'

>>
```

Now try something more ambitious – each row is a string followed of length 1 to 11 followed by 0 to 4 numbers (reals and integers).

Editor - C:\Users\glenn\Documents\MATLAB\mixed_array.txt

```
1  frederick 1 2 3 4
2  bob 5 6 7
3  M 8 9
4  michael 10 11 12
5  dick 13.1 14.2 15
6  jane 16 17.5 18
7  jill 19 20
8  bing 21
9  brad
10 christopher 22 23
```

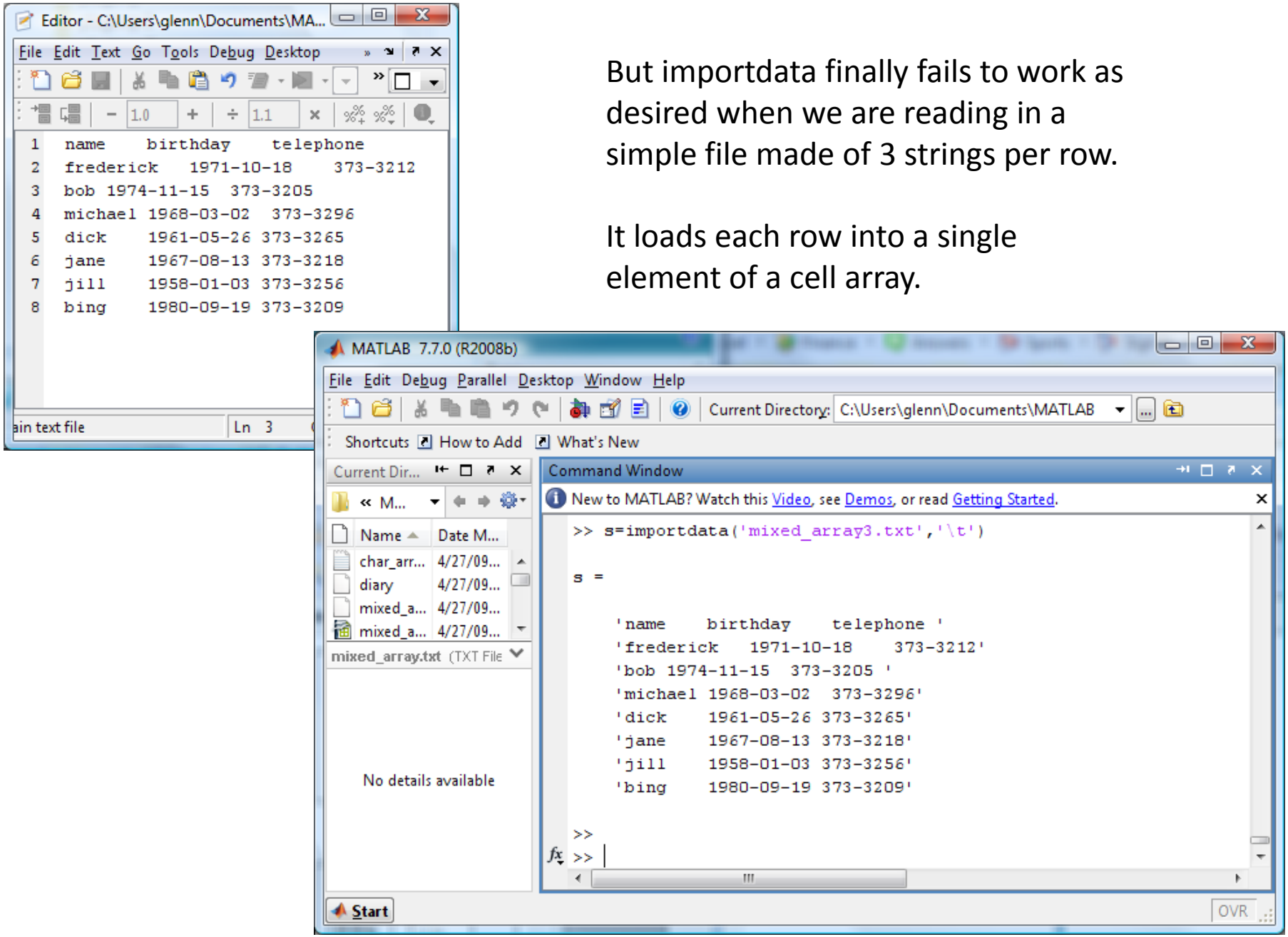
plain text file Ln 10 Col 12

MATLAB 7.7.0 (R2008b)

```
>>
>> s2 = importdata('mixed_array.txt')

s2 =

          data: [10x4 double]
      textdata: {10x1 cell}
    rowheaders: {10x1 cell}

>> s2.data

ans =

    1.0000    2.0000    3.0000    4.0000
    5.0000    6.0000    7.0000       NaN
    8.0000    9.0000       NaN       NaN
   10.0000   11.0000   12.0000       NaN
   13.1000   14.2000   15.0000       NaN
   16.0000   17.5000   18.0000       NaN
   19.0000   20.0000       NaN       NaN
   21.0000       NaN       NaN       NaN
      NaN       NaN       NaN       NaN
   22.0000   23.0000       NaN       NaN

>> s2.textdata

ans =

    'frederick'
    'bob'
    'M'
    'michael'
    'dick'
    'jane'
    'jill'
    'bing'
    'brad'
    'christopher'

>>
```

It has created a struct,
s2.data holds the numeric array,
s2.textdata holds the string data in a cell array

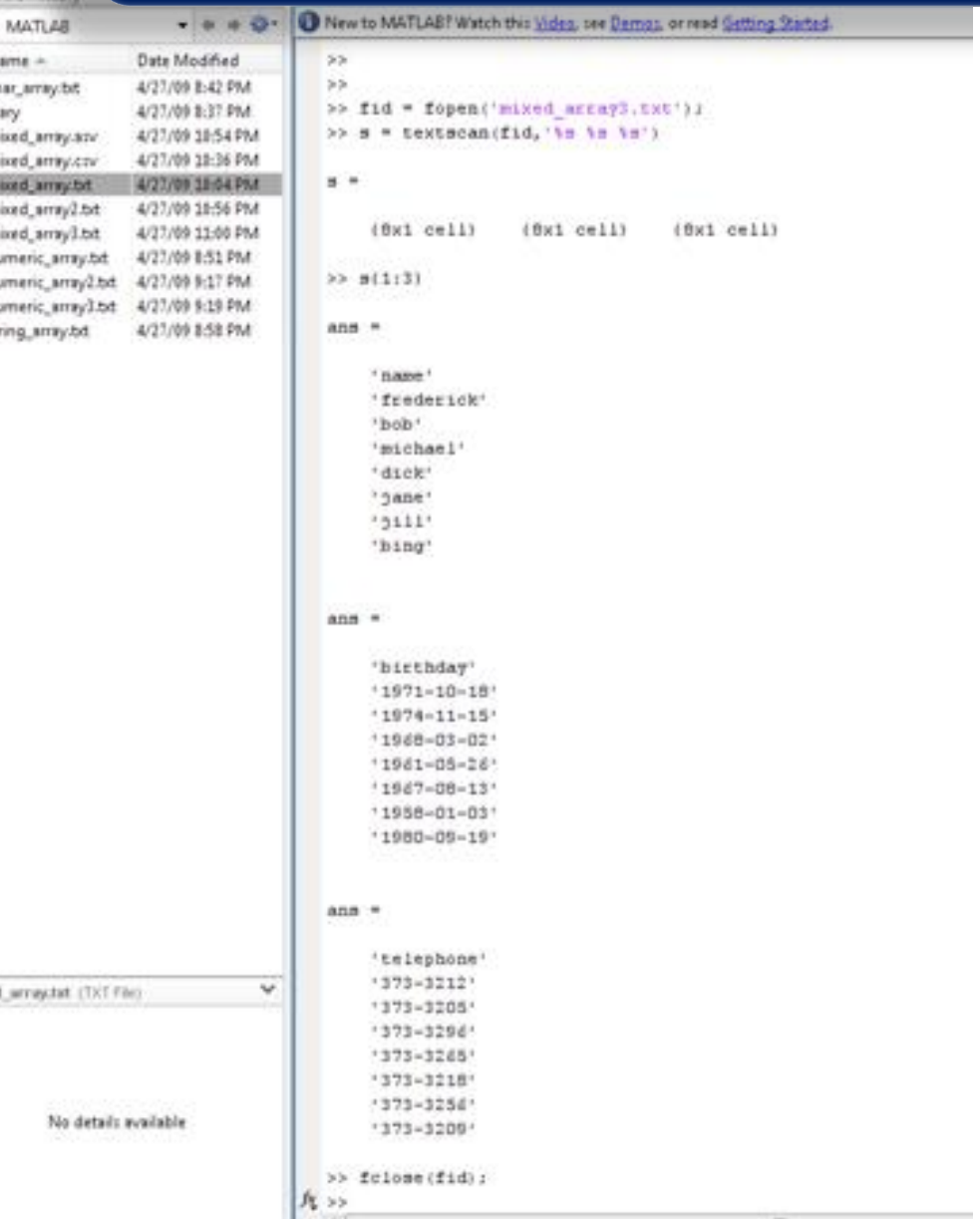
Non-existent values replaced with NaN in numeric array



But importdata finally fails to work as desired when we are reading in a simple file made of 3 strings per row.

It loads each row into a single element of a cell array.

textscan()



```
>>
>>
>> fid = fopen('mixed_array3.txt');
>> s = textscan(fid,'%s %s %s')

s =

    (0x1 cell)    (0x1 cell)    (0x1 cell)

>> s(1:3)

ans =

    'name'
    'frederick'
    'bob'
    'michael'
    'dick'
    'jane'
    'jill'
    'bing'


ans =

    'birthday'
    '1971-10-18'
    '1974-11-15'
    '1968-03-02'
    '1961-05-26'
    '1967-08-13'
    '1958-01-03'
    '1980-09-19'


ans =

    'telephone'
    '373-3212'
    '373-3205'
    '373-3296'
    '373-3265'
    '373-3218'
    '373-3256'
    '373-3209'

>> fclose(fid);
>>
```

cols = textscan(fid, format) works. Each column goes into a separate element of a cell array.

You are responsible for opening and closing the file though.

fid = fopen(filename, mode)

Is used to open a file.
Mode is:
‘r’ read (default)
‘w’ write (overwrite if file already exists)
‘a’ append (append to existing file if it already exists)

The latter are only used for writing data out to file.

fclose(fid) is used to close the file, after you’ve read (or written) it.

Editor - C:\Users\glenn\Documents\MATLAB\mixed_array3.txt

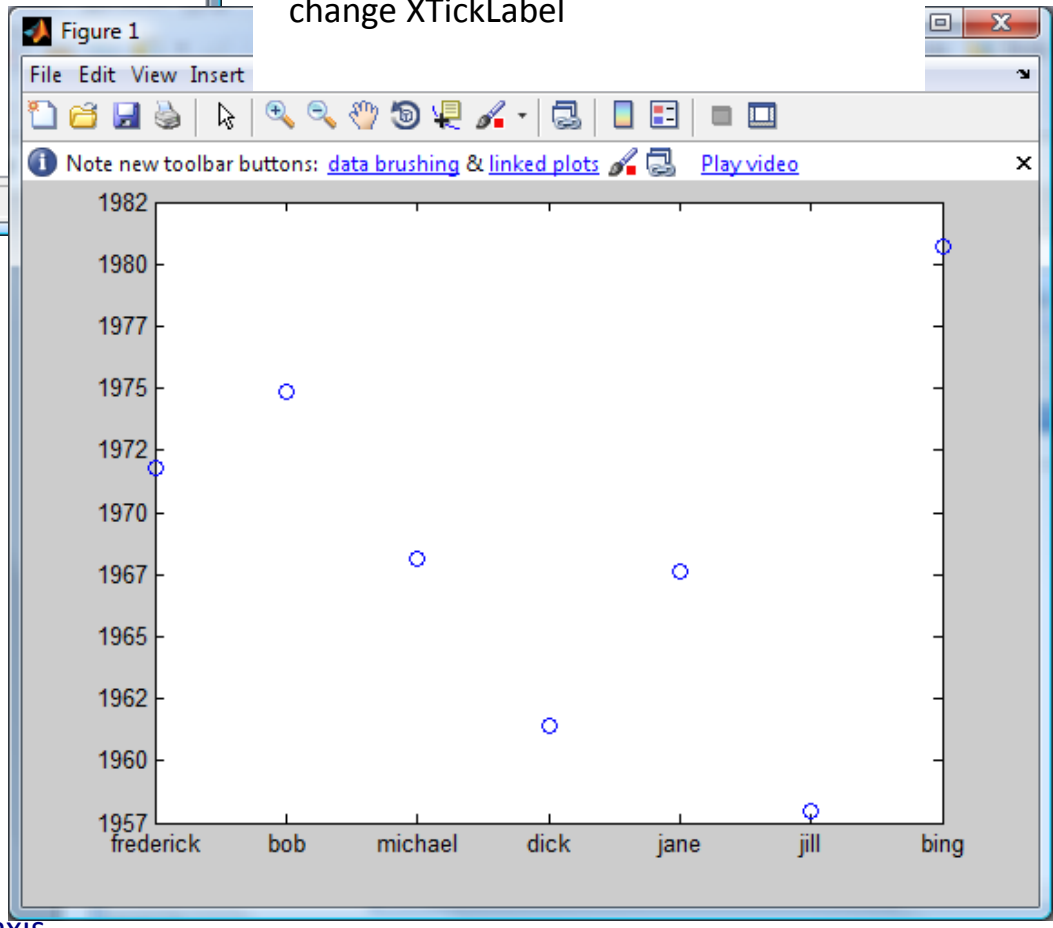
File Edit Text Go Tools Debug Desktop Window Help

```
1  frederick   37   1971-10-18    373-3212
2  bob 34  1974-11-15  373-3205
3  michael 40  1968-03-02  373-3296
4  dick    47  1961-05-26 373-3265
5  jane    41  1967-08-13 373-3218
6  jill    50  1958-01-03 373-3256
7  bing    28  1980-09-19 373-3209
```

plain text file Ln 1

Example to plot birthdays against name using:

- fopen/textscan/fclose
- struct
- datenum
- plot(y)
- datetick
- change XTickLabel



Script:

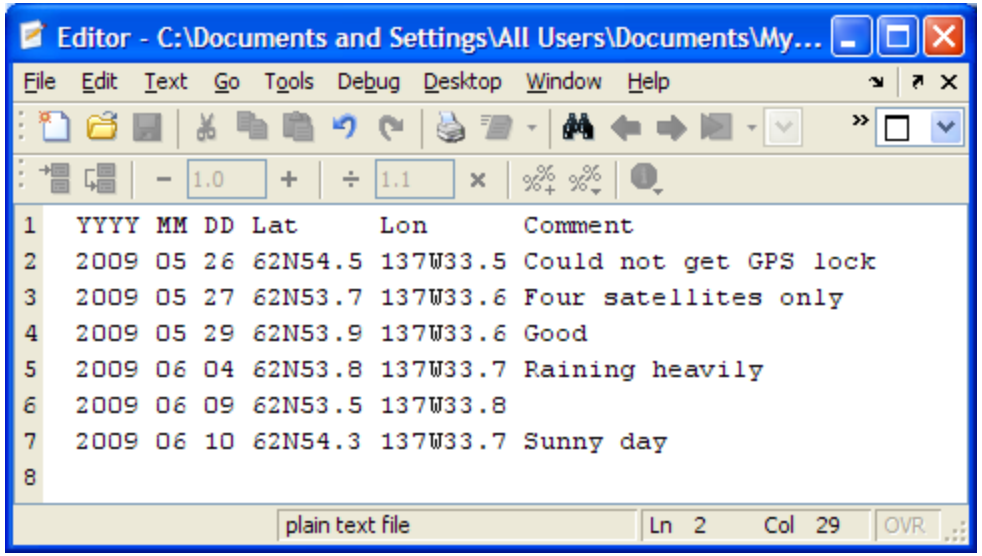
```
% read the file
fid = fopen('mixed_array3.txt');
A = textscan(fid, '%s %d %s %s');
fclose(fid);

% convert data into a struct
person.name = A{1};
person.age = A{2};
person.bday = datenum(A{3}); % convert to a datenum
person.phoneNum = A{4};

% plot the data
figure;
plot(person.bday);
datetick('y'); % let Matlab figure out how to label the y-axis
set(gca, 'XTickLabel', person.name); % change the XtickLabels from1:7 to names
```

Reading a line at a time: fgetl()

Line can be any length, any format.

```
Example
    fid=fopen('fgetl.m');
    while 1
        tline = fgetl(fid);
        if ~ischar(tline), break, end
        disp(tline)
    end
    fclose(fid);
```

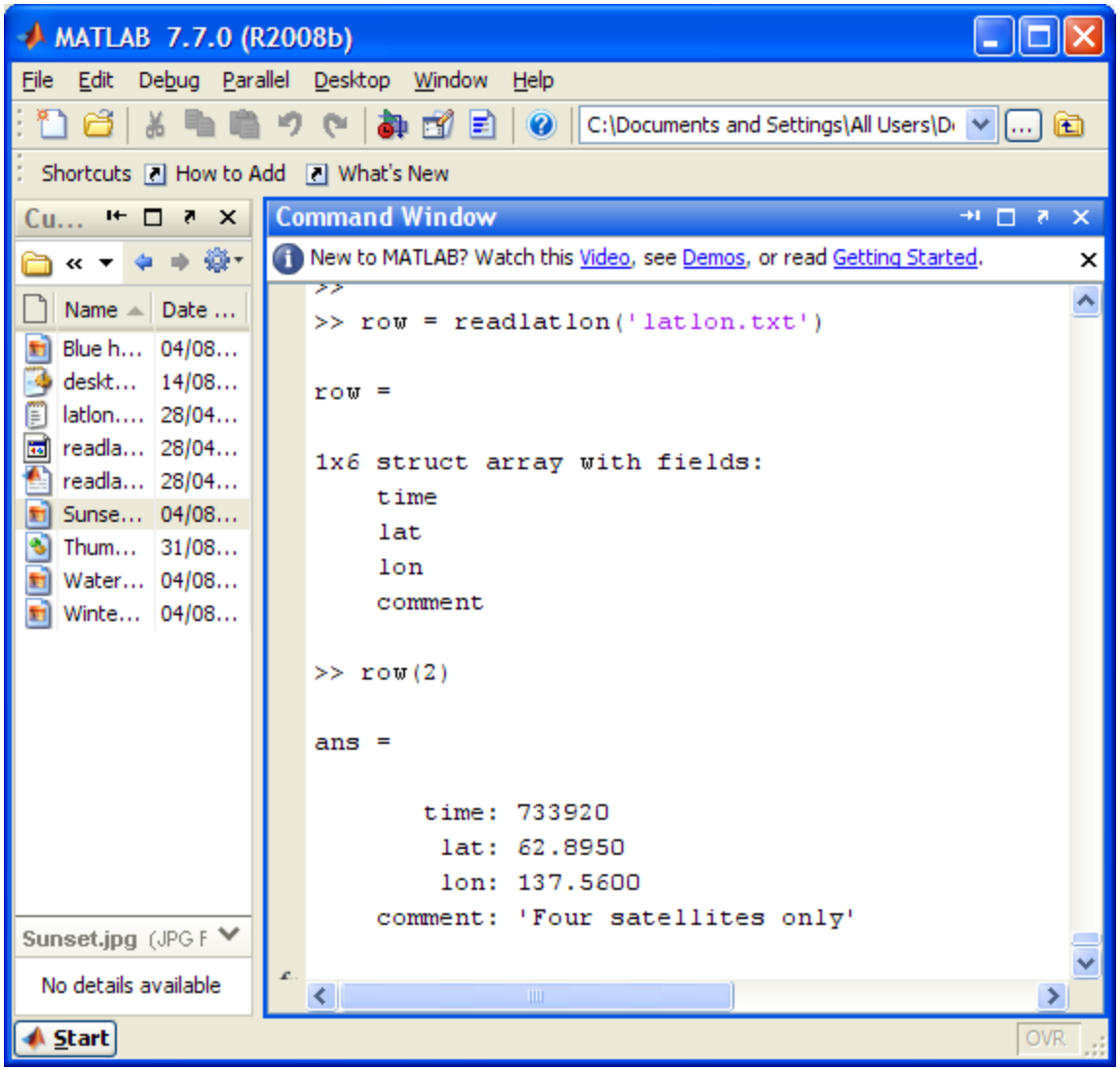
I often use it when each line has fields which appear in fixed positions.



Editor - C:\Documents and Settings\All Users\Documents\My Pictures\Sample Pictures\readlatlon.m

File Edit Text Go Cell Tools Debug Desktop Window Help

```
function row = readlatlon(filename)
%READLATLON read a latlon file

% initialise variables
linenum = 0;

if(exist(filename, 'file'))     % check if the file exists before trying to open it
    fid = fopen(filename);      % try to open the file, creating a pointer to it called fid
    while 1, % loop over all rows in the file
        myline = fgetl(fid);    % read the next line
        if ~ischar(myline), break, end % end loop when hit a blank line (end of file)

        % break up the line into components

        if strcmp(myline(1:2),'20') % if line starts with 20, it's probably a data row
            linenum = linenum + 1;

            % time
            yyyy = str2num(myline(1:4));
            mm = str2num(myline(6:7));
            dd = str2num(myline(9:10));
            row(linenum).time = datenum(yyyy, mm, dd);

            % lat
            row(linenum).lat = str2num(myline(12:13)) + str2num(myline(15:18))/60;

            % lon
            row(linenum).lon = str2num(myline(20:22)) + str2num(myline(24:27))/60;

            % comment
            if length(myline)>28
                row(linenum).comment = myline(29:end);
            end
```


MATLAB 7.7.0 (R2008b)
File Edit Debug Parallel Desktop Window Help
C:\Documents and Settings\All Users\D
Shortcuts How to Add What's New
Cu...
Name Date ...
Blue h... 04/08...
deskt... 14/08...
latlon.... 28/04...
readla... 28/04...
readla... 28/04...
Sunse... 04/08...
Thum... 31/08...
Water... 04/08...
Winte... 04/08...
Sunset.jpg (JPG F
No details available
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> row = readlatlon('latlon.txt')
row =
1x6 struct array with fields:
time
lat
lon
comment
>> row(2)
ans =
time: 733920
lat: 62.8950
lon: 137.5600
comment: 'Four satellites only'
Start
OVR

Reading one data type at a time: fscanf()

```
Examples:
    S = fscanf(fid,'%s')    reads (and returns) a character string.
    A = fscanf(fid,'%5d')   reads 5-digit decimal integers.
```

writing text files: fprintf()

print formatted string to a file

```
fout = fopen(filename, 'w')  % write to new file filename (replacing file it if already exists)
for (r=1:numRows )          % loop over all rows

        fprintf(fout, '%s\t%12.7f\n', datestr(dnum(r),31), data(r));

end
fclose(fout)
```

\t = <tab>
\n = <return>
datestr(dnum(r), 31) = print dnum(r) as a datestr using dateform 31
%12.7f= print this real variable as 12 characters with 7 after the decimal point

Output file might be like:

```
20090423T180000     1234.1234567
20090423T180100     1357.1357911
20090423T180200     1470.1470369
```

Related functions:
dlmwrite – for delimited fields
(csvwrite for comma delimited fields)

Reading and Writing Excel files:

xlsread
xlswrite

Reading Excel files

[numeric, txt, raw] = **xlsread**('myfile.xls'); % will attempt to read all sheets

[numeric, txt, raw] = **xlsread**('myfile.xls', 'sheet1'); % read sheet1 only

numeric – a matrix that contains all the numeric columns

txt – a cell array contain all text columns

raw – a cell array contain any columns xlsread could not interpret

Writing Excel files

Related functions are **csvread** and **dlmread**

xlswrite('myfile.xls', myarray, 'sheet2');

myarray - a numeric array or a cell array

Related functions are **csvwrite**, **dlmwrite**

Writing Excel files

xlswrite('myfile.xls', myarray, 'sheet2');

myarray - a numeric array or a cell array

Related functions are **csvwrite**, **dlmwrite**

EXCEL EXAMPLES

Two real examples:

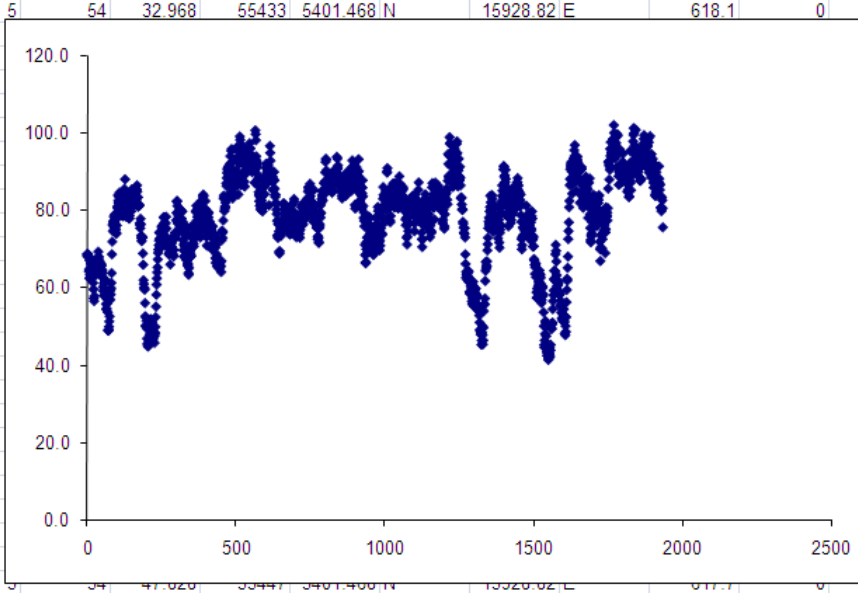
(1) Grasshopper diet data from Ellen Trainor

(2) Gas data from Taryn Lopez

Kary72208_P20.xls [Compatibility Mode] - Microsoft Excel

Home | Insert | Page Layout | Formulas | Data | Review | View

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	550	2008	7	21	5	54	26.203	55426	5401.468	N	15928.82	E	618	0	0	392.12	68.6			
2	551	2008	7	21	5	54	26.812	55426	5401.468	N	15928.82	E	618	0	0	388.94	68.0			
3	552	2008	7	21	5	54	27.437	55427	5401.468	N	15928.82	E	618	0	0	392.44	68.6			
4	553	2008	7	21	5	54	28.046	55428	5401.468	N	15928.82	E	618	0	0	390.58	68.3			
5	554	2008	7	21	5	54	28.656	55428	5401.468	N	15928.82	E	618	0						
6	555	2008	7	21	5	54	29.281	55429	5401.468	N	15928.82	E	618.1	0						
7	556	2008	7	21	5	54	29.89	55430	5401.468	N	15928.82	E	618.1	0						
8	557	2008	7	21	5	54	30.5	55430	5401.468	N	15928.82	E	618.1	0						
9	558	2008	7	21	5	54	31.125	55431	5401.468	N	15928.82	E	618.1	0						
10	559	2008	7	21	5	54	31.734	55431	5401.468	N	15928.82	E	618.1	0						
11	560	2008	7	21	5	54	32.343	55432	5401.468	N	15928.82	E	618.1	0						
12	561	2008	7	21	5	54	32.968	55433	5401.468	N	15928.82	E	618.1	0						
13	562	2008	7	21																
14	563	2008	7	21																
15	564	2008	7	21																
16	565	2008	7	21																
17	566	2008	7	21																
18	567	2008	7	21																
19	568	2008	7	21																
20	569	2008	7	21																
21	570	2008	7	21																
22	571	2008	7	21																
23	572	2008	7	21																
24	573	2008	7	21																
25	574	2008	7	21																
26	575	2008	7	21																
27	576	2008	7	21																
28	577	2008	7	21																
29	578	2008	7	21																
30	579	2008	7	21																
31	580	2008	7	21																
32	581	2008	7	21																
33	582	2008	7	21																
34	583	2008	7	21																
35	584	2008	7	21																
36	585	2008	7	21																
37	586	2008	7	21	5	54	48.39	55448	5401.468	N	15928.82	E	617.7	0						
38	587	2008	7	21	5	54	48.984	55449	5401.468	N	15928.82	E	617.7	0						
39	588	2008	7	21	5	54	49.671	55449	5401.468	N	15928.82	E	617.7	0						
40	589	2008	7	21	5	54	50.25	55450	5401.468	N	15928.82	E	617.6	0						
41	590	2008	7	21	5	54	50.859	55450	5401.468	N	15928.82	E	617.6	0						
42	591	2008	7	21	5	54	51.515	55451	5401.468	N	15928.82	E	617.6	0						



ScanP20

Ready

"The FLYSPEC File Kary72208_P20.xls, contains:
scan number
year
date
month
hour
min
second
time (HHMMSS)
lat (degree dec min)
(N)
long (deg dec min)
(E)
Don't know the next 3 columns...
Column 17 is SO2 column density
Column 18 is SO2 emission rate

plot... SO2 emission rate vs time."

Editor - C:\Documents and Settings\GLENN\Desktop\matlab_lecture\...

File Edit Text Go Cell Tools Debug Desktop Window Help

```
function so2emissionrate()
%SO2EMISSIONRATE reads an Excel file and plots
%emission rate against time

% xls filename
filename = 'Kary72208_P20.xls';

% read data into a struct
s = readdata(filename);

% plot emission rate vs. time
figure;
plot(s.time, s.emissionrate);

% label the axes
xlabel(sprintf('time on %s',datestr(s.time(1),1)))
ylabel('emission rate')
datetick('x');

% print to file
print -dpng emissionrate.png

function st = readdata(fname)
[a, t] = xlsread(fname);
st.time = datenum(a(:,2:7));
st.scannum = a(:,1);
st.lat = a(:,9);
st.lon = a(:,11);
st.columndensity = a(:,16);
st.emissionrate = a(:,17);
```

latlon.txt | so2emissionrate.m

so2emissionrate / readdata Ln 24 Col 25

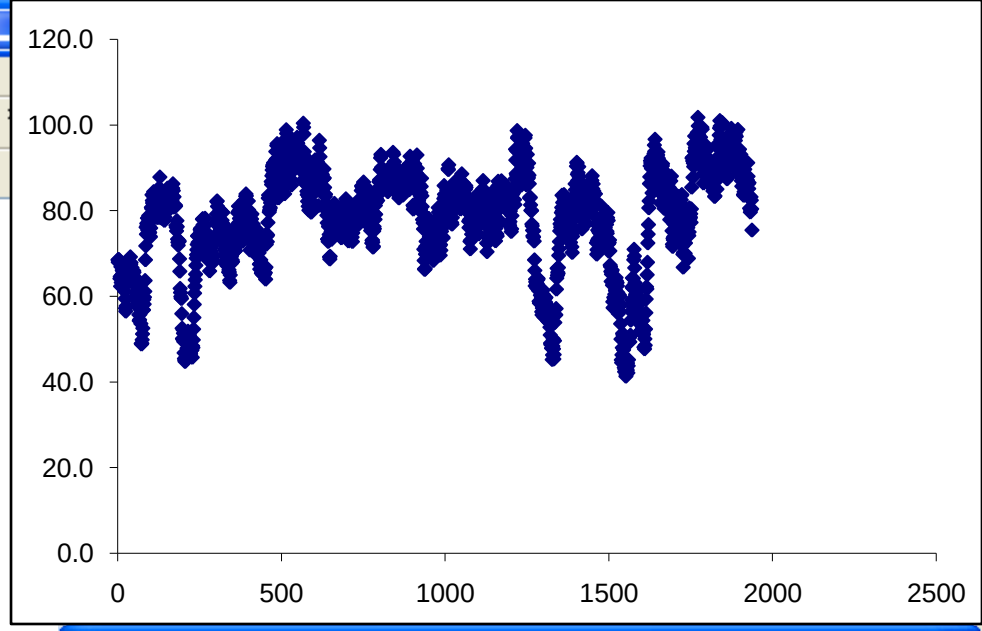
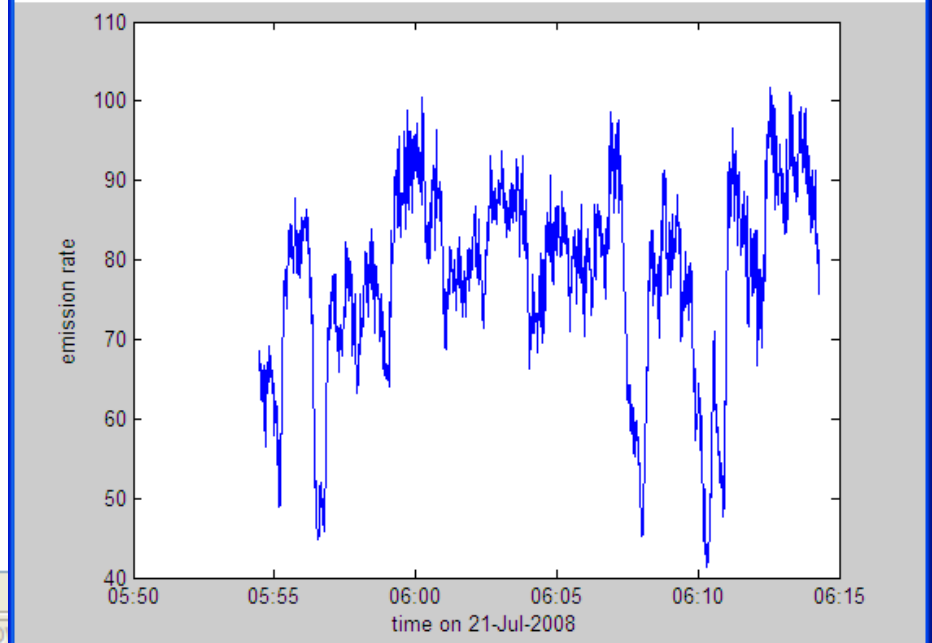
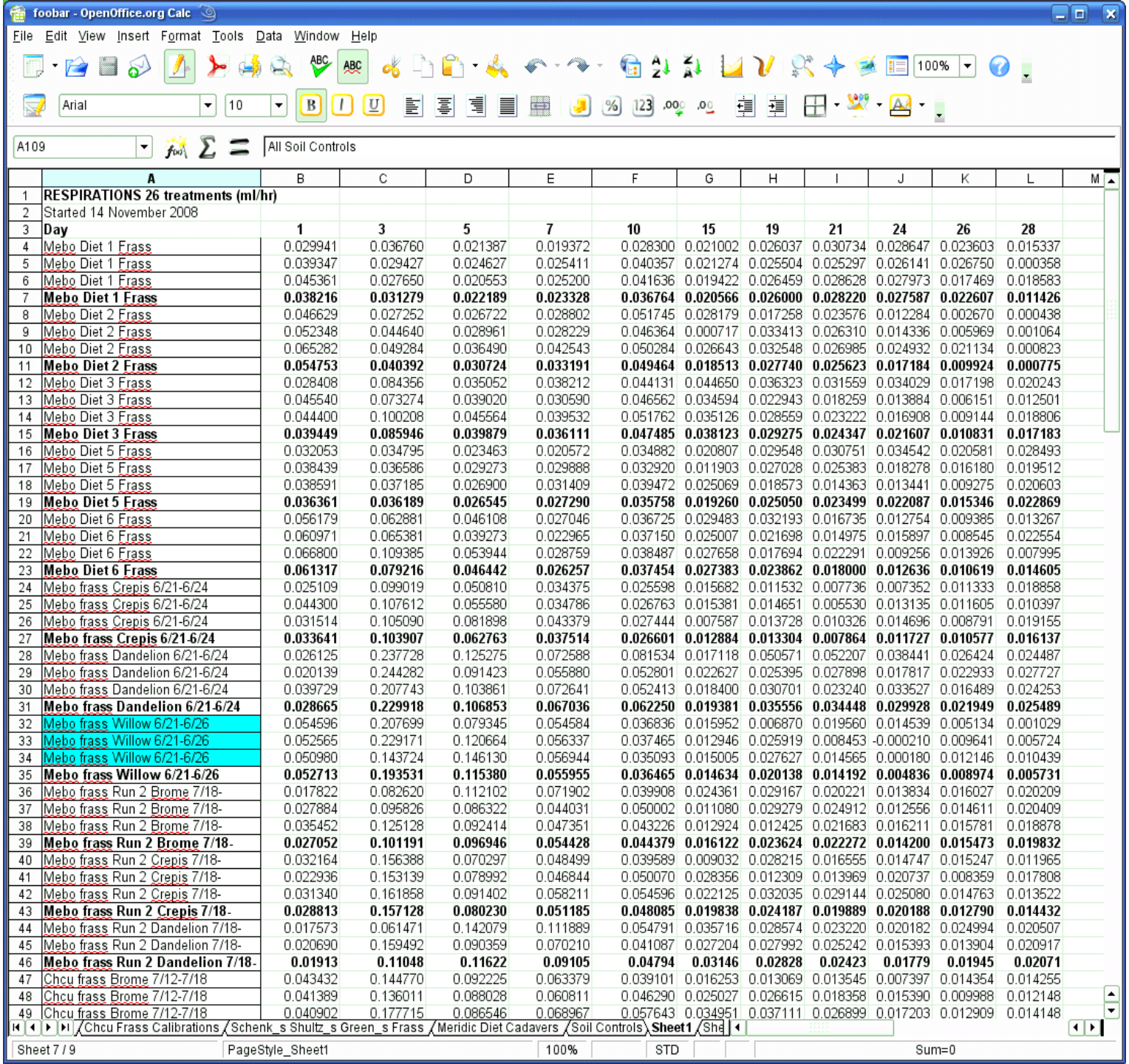


Figure 2

File Edit View Insert Tools Desktop Window Help

Note new toolbar buttons: data brushing & linked plots Play video





	A	B	C	D	E	F	G	H	I	J	K	L	M
1	**RESPIRATIONS 26 treatments (ml/hr)**												
2	Started 14 November 2008												
3	**Day**	**1**	**3**	**5**	**7**	**10**	**15**	**19**	**21**	**24**	**26**	**28**	
4	Mebo Diet 1 Frass	0.029941	0.036760	0.021387	0.019372	0.028300	0.021002	0.026037	0.030734	0.028647	0.023603	0.015337	
5	Mebo Diet 1 Frass	0.039347	0.029427	0.024627	0.025411	0.040357	0.021274	0.025504	0.025297	0.026141	0.026750	0.000358	
6	Mebo Diet 1 Frass	0.045361	0.027650	0.020553	0.025200	0.041636	0.019422	0.026459	0.028628	0.027973	0.017469	0.018583	
7	**Mebo Diet 1 Frass**	**0.038216**	**0.031279**	**0.022189**	**0.023328**	**0.036764**	**0.020566**	**0.026000**	**0.028220**	**0.027587**	**0.022607**	**0.011426**	
8	Mebo Diet 2 Frass	0.046629	0.027252	0.026722	0.028802	0.051745	0.028179	0.017258	0.023576	0.012284	0.002670	0.000438	
9	Mebo Diet 2 Frass	0.052348	0.044640	0.028961	0.028229	0.046364	0.000717	0.033413	0.026310	0.014336	0.005969	0.001064	
10	Mebo Diet 2 Frass	0.065282	0.049284	0.036490	0.042543	0.050284	0.026643	0.032548	0.026985	0.024932	0.021134	0.000823	
11	**Mebo Diet 2 Frass**	**0.054753**	**0.040392**	**0.030724**	**0.033191**	**0.049464**	**0.018513**	**0.027740**	**0.025623**	**0.017184**	**0.009924**	**0.000775**	
12	Mebo Diet 3 Frass	0.028408	0.084356	0.035052	0.038212	0.044131	0.044650	0.036323	0.031559	0.034029	0.017198	0.020243	
13	Mebo Diet 3 Frass	0.045540	0.073274	0.039020	0.030590	0.046562	0.034594	0.022943	0.018259	0.013884	0.006151	0.012501	
14	Mebo Diet 3 Frass	0.044400	0.100208	0.045564	0.039532	0.051762	0.035126	0.028559	0.023222	0.016908	0.009144	0.018806	
15	**Mebo Diet 3 Frass**	**0.039449**	**0.085946**	**0.039879**	**0.036111**	**0.047485**	**0.038123**	**0.029275**	**0.024347**	**0.021607**	**0.010831**	**0.017183**	
16	Mebo Diet 5 Frass	0.032053	0.034795	0.023463	0.020572	0.034882	0.020807	0.029548	0.030751	0.034542	0.020581	0.028493	
17	Mebo Diet 5 Frass	0.038439	0.036586	0.029273	0.029888	0.032920	0.011903	0.027028	0.025383	0.018278	0.016180	0.019512	
18	Mebo Diet 5 Frass	0.038591	0.037185	0.026900	0.031409	0.039472	0.025069	0.018573	0.014363	0.013441	0.009275	0.020603	
19	**Mebo Diet 5 Frass**	**0.036361**	**0.036189**	**0.026545**	**0.027290**	**0.035758**	**0.019260**	**0.025050**	**0.023499**	**0.022087**	**0.015346**	**0.022869**	
20	Mebo Diet 6 Frass	0.056179	0.062881	0.046108	0.027046	0.036725	0.029483	0.032193	0.016735	0.012754	0.009385	0.013267	
21	Mebo Diet 6 Frass	0.060971	0.065381	0.039273	0.022965	0.037150	0.025007	0.021698	0.014975	0.015897	0.008545	0.022554	
22	Mebo Diet 6 Frass	0.066800	0.109385	0.053944	0.028759	0.038487	0.027658	0.017694	0.022291	0.009256	0.013926	0.007995	
23	**Mebo Diet 6 Frass**	**0.061317**	**0.079216**	**0.046442**	**0.026257**	**0.037454**	**0.027383**	**0.023862**	**0.018000**	**0.012636**	**0.010619**	**0.014605**	
24	Mebo frass Crepis 6/21-6/24	0.025109	0.099019	0.050810	0.034375	0.025598	0.015682	0.011532	0.007736	0.007352	0.011333	0.018858	
25	Mebo frass Crepis 6/21-6/24	0.044300	0.107612	0.055580	0.034786	0.026763	0.015381	0.014651	0.005530	0.013135	0.011605	0.010397	
26	Mebo frass Crepis 6/21-6/24	0.031514	0.105090	0.081898	0.043379	0.027444	0.007587	0.013728	0.010326	0.014696	0.008791	0.019155	
27	**Mebo frass Crepis 6/21-6/24**	**0.033641**	**0.103907**	**0.062763**	**0.037514**	**0.026601**	**0.012884**	**0.013304**	**0.007864**	**0.011727**	**0.010577**	**0.016137**	
28	Mebo frass Dandelion 6/21-6/24	0.026125	0.237728	0.125275	0.072588	0.081534	0.017118	0.050571	0.052207	0.038441	0.026424	0.024487	
29	Mebo frass Dandelion 6/21-6/24	0.020139	0.244282	0.091423	0.055880	0.052801	0.022627	0.025395	0.027898	0.017817	0.022933	0.027727	
30	Mebo frass Dandelion 6/21-6/24	0.039729	0.207743	0.103861	0.072641	0.052413	0.018400	0.030701	0.023240	0.033527	0.016489	0.024253	
31	**Mebo frass Dandelion 6/21-6/24**	**0.028665**	**0.229918**	**0.106853**	**0.067036**	**0.062250**	**0.019381**	**0.035556**	**0.034448**	**0.029928**	**0.021949**	**0.025489**	
32	Mebo frass Willow 6/21-6/26	0.054596	0.207699	0.079345	0.054584	0.036836	0.015952	0.006870	0.019560	0.014539	0.005134	0.001029	
33	Mebo frass Willow 6/21-6/26	0.052565	0.229171	0.120664	0.056337	0.037465	0.012946	0.025919	0.008453	-0.000210	0.009641	0.005724	
34	Mebo frass Willow 6/21-6/26	0.050980	0.143724	0.146130	0.056944	0.035093	0.015005	0.027627	0.014565	0.000180	0.012146	0.010439	
35	**Mebo frass Willow 6/21-6/26**	**0.052713**	**0.193531**	**0.115380**	**0.055955**	**0.036465**	**0.014634**	**0.020138**	**0.014192**	**0.004836**	**0.008974**	**0.005731**	
36	Mebo frass Run 2 Brome 7/18-	0.017822	0.082620	0.112102	0.071902	0.039908	0.024361	0.029167	0.020221	0.013834	0.016027	0.020209	
37	Mebo frass Run 2 Brome 7/18-	0.027884	0.095826	0.086322	0.044031	0.050002	0.011080	0.029279	0.024912	0.012556	0.014611	0.020409	
38	Mebo frass Run 2 Brome 7/18-	0.035452	0.125128	0.092414	0.047351	0.043226	0.012924	0.012425	0.021683	0.016211	0.015781	0.018878	
39	**Mebo frass Run 2 Brome 7/18-**	**0.027052**	**0.101191**	**0.096946**	**0.054428**	**0.044379**	**0.016122**	**0.023624**	**0.022272**	**0.014200**	**0.015473**	**0.019832**	
40	Mebo frass Run 2 Crepis 7/18-	0.032164	0.156388	0.070297	0.048499	0.039589	0.009032	0.028215	0.016555	0.014747	0.015247	0.011965	
41	Mebo frass Run 2 Crepis 7/18-	0.022936	0.153139	0.078992	0.046844	0.050070	0.028356	0.012309	0.013969	0.020737	0.008359	0.017808	
42	Mebo frass Run 2 Crepis 7/18-	0.031340	0.161858	0.091402	0.058211	0.054596	0.022125	0.032035	0.029144	0.025080	0.014763	0.013522	
43	**Mebo frass Run 2 Crepis 7/18-**	**0.028813**	**0.157128**	**0.080230**	**0.051185**	**0.048085**	**0.019838**	**0.024187**	**0.019889**	**0.020188**	**0.012790**	**0.014432**	
44	Mebo frass Run 2 Dandelion 7/18-	0.017573	0.061471	0.142079	0.111889	0.054791	0.035716	0.028574	0.023220	0.020182	0.024994	0.020507	
45	Mebo frass Run 2 Dandelion 7/18-	0.020690	0.159492	0.090359	0.070210	0.041087	0.027204	0.027992	0.025242	0.015393	0.013904	0.020917	
46	**Mebo frass Run 2 Dandelion 7/18-**	**0.01913**	**0.11048**	**0.11622**	**0.09105**	**0.04794**	**0.03146**	**0.02828**	**0.02423**	**0.01779**	**0.01945**	**0.02071**	
47	Chcu frass Brome 7/12-7/18	0.043432	0.144770	0.092225	0.063379	0.039101	0.016253	0.013069	0.013545	0.007397	0.014354	0.014255	
48	Chcu frass Brome 7/12-7/18	0.041389	0.136011	0.088028	0.060811	0.046290	0.025027	0.026615	0.018358	0.015390	0.009988	0.012148	
49	Chcu frass Brome 7/12-7/18	0.040902	0.177715	0.086546	0.068967	0.057643	0.034951	0.037111	0.026899	0.017203	0.012909	0.014148	

apply a conversion factor to all measurements

plot the first six rows in bold as 6 lines on a plot

compute the area under each line

do the same for the next 6 bold rows

then the next 5

then the next 5

then the next 7

then the next 4

1) Define the outline

```
function grasshopperDiets()
% define conversion factors

% load the data
[a, t] = xlsread('grasshopperDiets.xls');
day = a(3, :); % day is in row 3

% define a vector which has the row numbers for the first plot
i = 7:4:23;

% plot these rows & compute area under graph
[area{1}, legendStr] = areas(day, a, i, color, t);

% plot a bar graph of the area
plotarea(area{1}, color, legendStr)

...

% define a vector which has the row numbers for the fifth plot
i = 82:4:102;

% plot these r
[area{5}, lege

% plot a bar graph of the area
plotarea(area6, color, legendStr)
```

not ideal as we repeat these 3 lines for each set of rows we want to plot

2) Write the functions

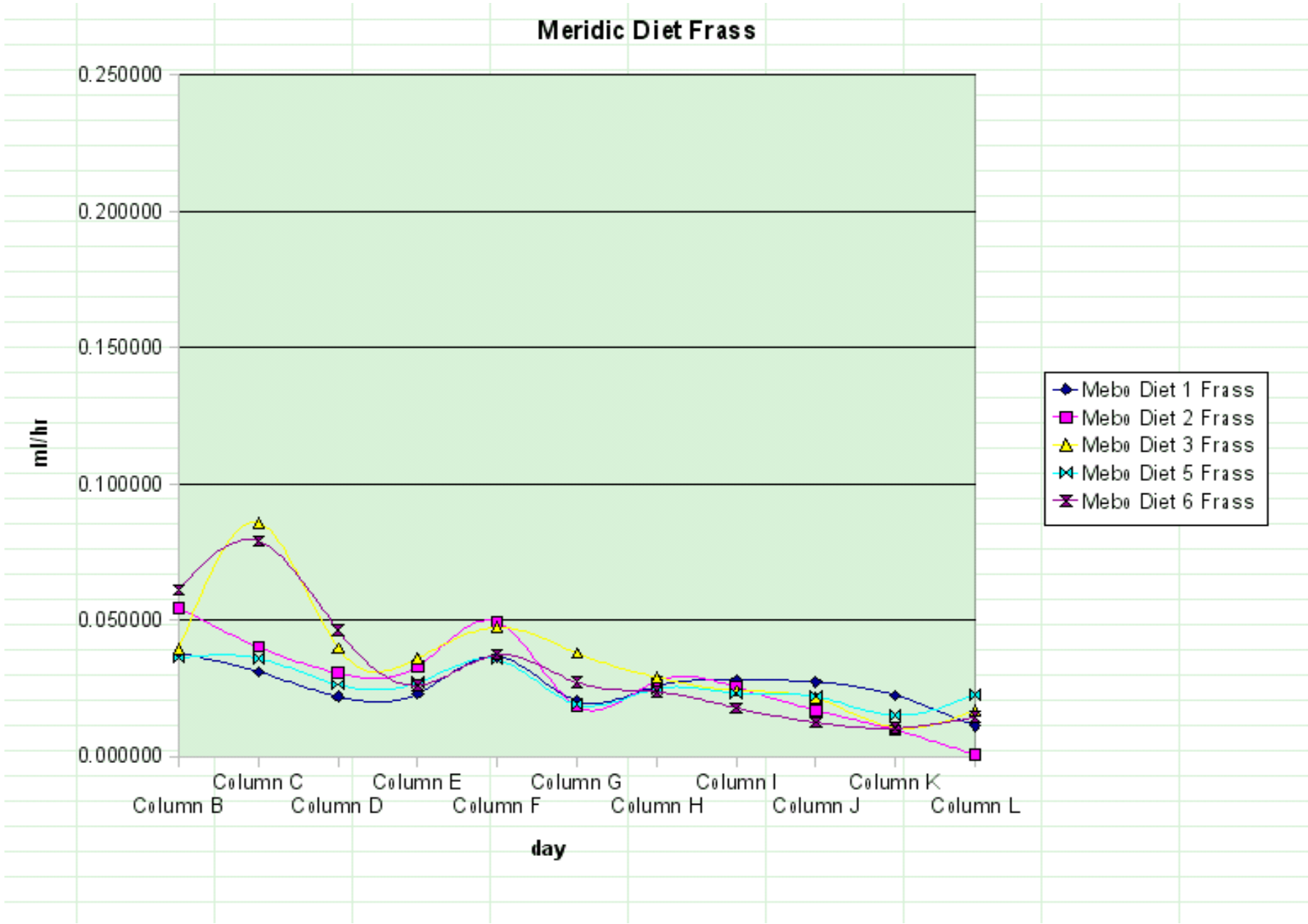
```
function [area,leg]=areas(day, a, i, color, t)
...

function plotarea(area, color,leg);
...
```

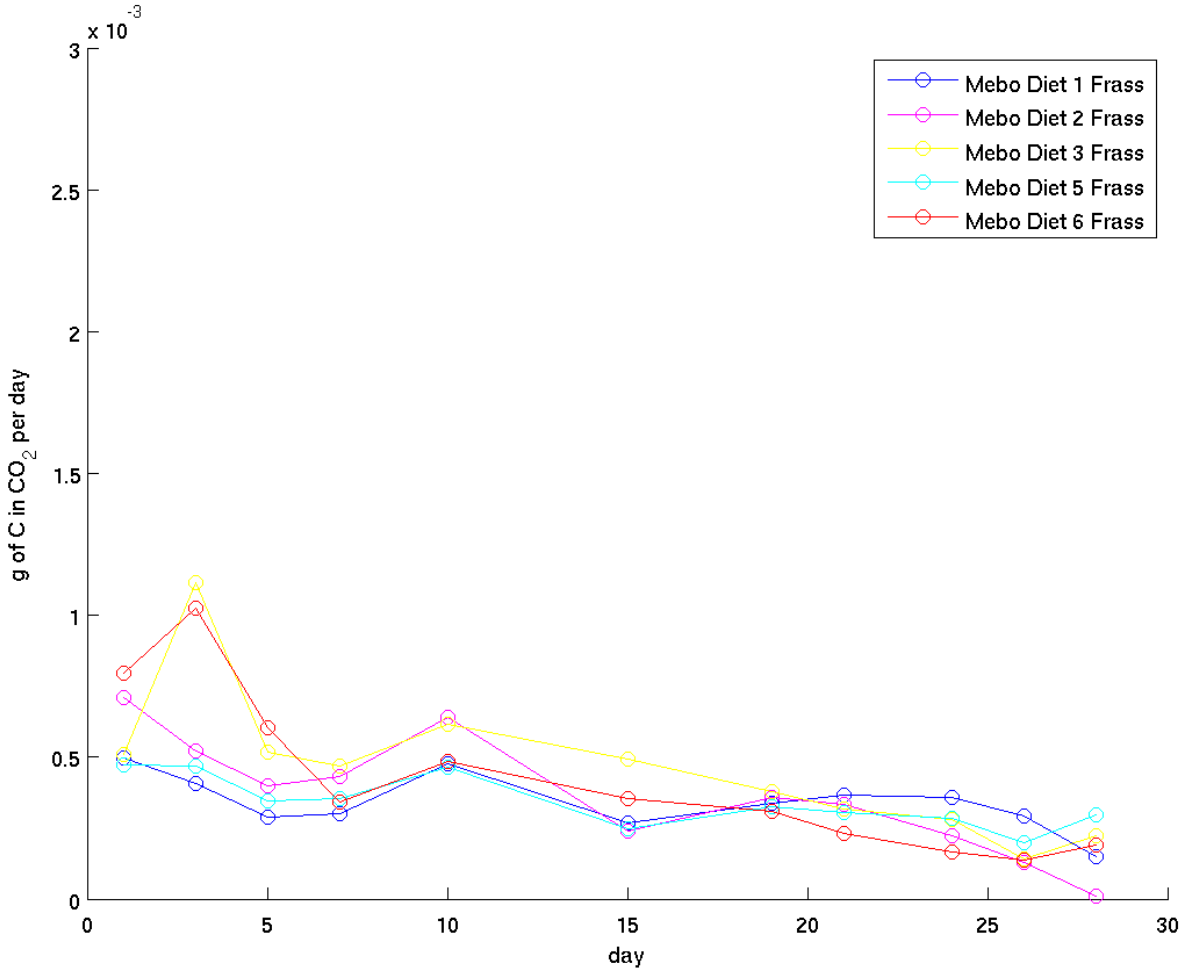
trapz

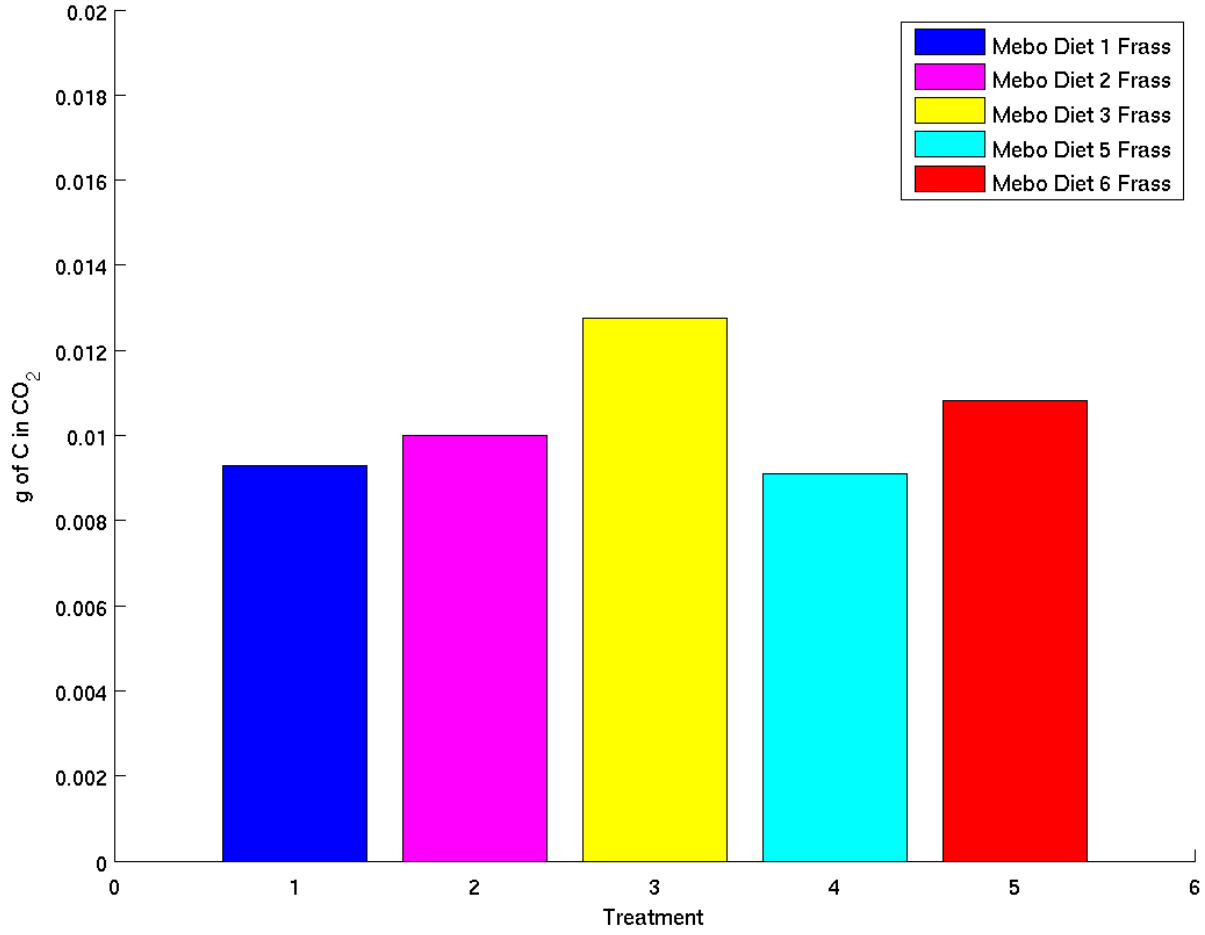
3) Simplify outline further, and fill in details

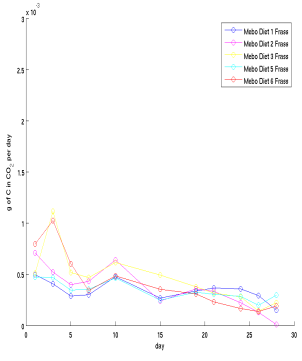
```
function grasshopperDiets()

% define conversion factor
conversionFactor = 1.98 * 12 / (12 + 2 * 16);

% load the data
[a, t] = xlsread('grasshopperDiets.xls');
day = a(3, :); % day is in row 3

% apply conversion factor
a[4:109, :] = a[4:109, :] * conversionFactor;

% set colours to match Excel
color = 'bmycrgk';

% define a vectors in a cell array which have row numbers for each plot
i{1} = 7:4:23;
...
i{5} = 82:4:102;

% loop over each set of row
for count = 1:length(i)
        % plot these rows & compute area under graph
        [area{count}, legendStr] = areas(day, a, i{count}, color, t);

        % plot a bar graph of the area
        plotarea(area{count}, color, legendStr)

end
```

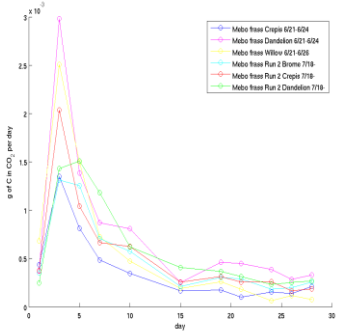
if spreadsheet grows, applying this to more rows is as simple as adding a new element to i{ } !!

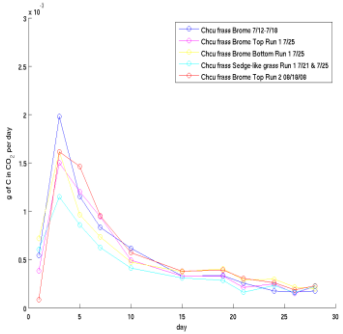

Meridic Diet Frass
0.250000
0.200000
0.150000
0.100000
0.050000
0.000000
ml/hr
Column B
Column C
Column D
Column E
Column F
Column G
Column H
Column I
Column J
Column K
Column L
day
Mebo Diet 1 Frass
Mebo Diet 2 Frass
Mebo Diet 3 Frass
Mebo Diet 5 Frass
Mebo Diet 6 Frass

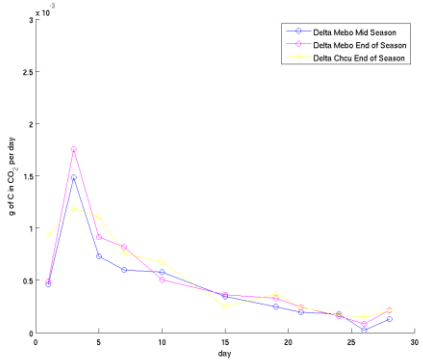

x 10^-3
3
2.5
2
1.5
1
0.5
0
g of C in CO2 per day
0
5
10
15
20
25
30
day
Mebo Diet 1 Frass
Mebo Diet 2 Frass
Mebo Diet 3 Frass
Mebo Diet 5 Frass
Mebo Diet 6 Frass

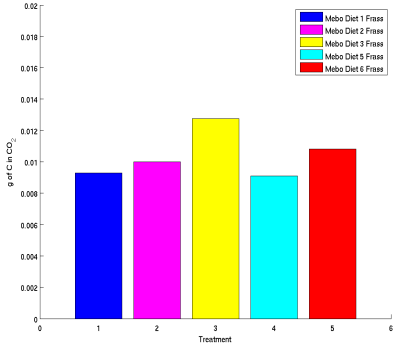

Mebo Diet 1 Frass
Mebo Diet 2 Frass
Mebo Diet 3 Frass
Mebo Diet 5 Frass
Mebo Diet 6 Frass
g of C in CO_2
0.02
0.018
0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
0
1
2
3
4
5
6
Treatment

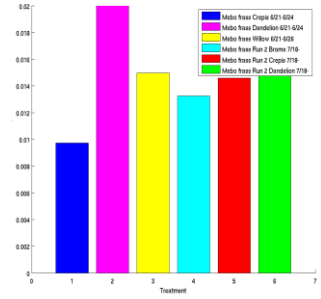

Mebo Diet 1 Frass
Mebo Diet 2 Frass
Mebo Diet 3 Frass
Mebo Diet 5 Frass
Mebo Diet 6 Frass
g of C in CO2 per day
day

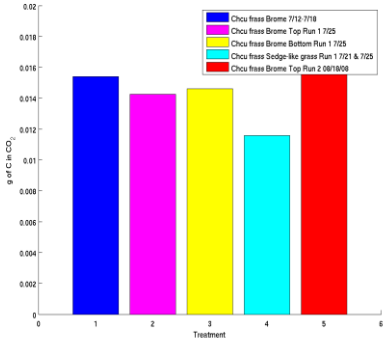

Mebo frass Crepis 6/21-6/24
Mebo frass Dandelion 6/21-6/24
Mebo frass Willow 6/21-6/26
Mebo frass Run 2 Brome 7/19-
Mebo frass Run 2 Crepis 7/19-
Mebo frass Run 2 Dandelion 7/19
g of C in CO2 per day
day

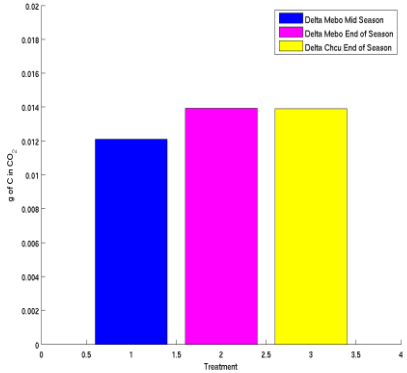

Chcu frass Brome 7/12-7/18
Chcu frass Brome Top Run 1 7/25
Chcu frass Brome Bottom Run 1 7/25
Chcu frass Sedge-like grass Run 1 7/21 & 7/25
Chcu frass Brome Top Run 2 08/18/08
g of C in CO2 per day
day

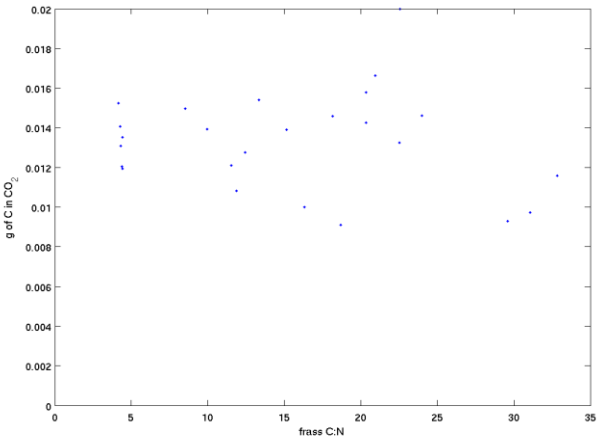

Delta Mebo Mid Season
Delta Mebo End of Season
Delta Chcu End of Season
g of C in CO2 per day
day

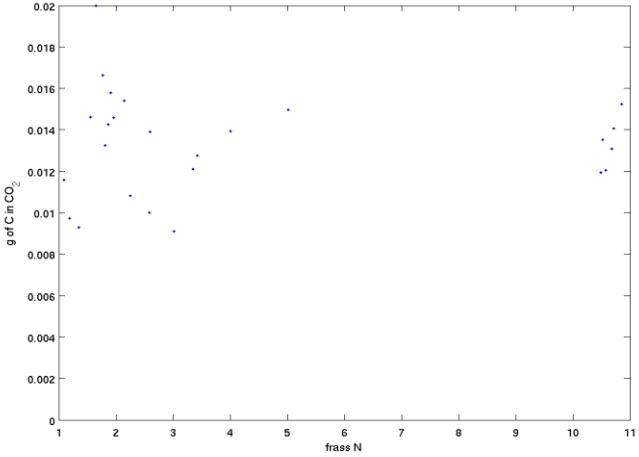

Mebo Diet 1 Frass
Mebo Diet 2 Frass
Mebo Diet 3 Frass
Mebo Diet 5 Frass
Mebo Diet 6 Frass
g of C in CO2
Treatment


Mebo frass Crepis 6/21-6/24
Mebo frass Dandelion 6/21-6/24
Mebo frass Willow 6/21-6/26
Mebo frass Run 2 Brome 7/19-
Mebo frass Run 2 Crepis 7/19-
Mebo frass Run 2 Dandelion 7/19
Treatment


Chcu frass Brome 7/12-7/18
Chcu frass Brome Top Run 1 7/25
Chcu frass Brome Bottom Run 1 7/25
Chcu frass Sedge-like grass Run 1 7/21 & 7/25
Chcu frass Brome Top Run 2 08/18/08
g of C in CO2
Treatment


Delta Mebo Mid Season
Delta Mebo End of Season
Delta Chcu End of Season
g of C in CO2
Treatment


g of C in CO2
frass C:N


g of C in CO2
frass N

summary

Plotting commands:
- plot, semilogx, semilogy, loglog, bar, barh, stem, stairs, hist, pie
- plot3, bar3, pie3, hist3, contour, surf, mesh, quiver, (mapping toolbox)
- image, imagesc
- datetick (datenum, datestr), subplot, hold on, axes

Graphical files:
- imread, print

MAT(LAB binary) files:
-load, save

Numerical ASCII files:
-load, dlmread, importdata, save, dlmwrite

Text files:
- importdata, textscan, fgetl, fscanf, fprintf (fopen/fclose)

Excel files:
- xlsread, xlswrite

Not covered: reading and writing generic binary files with: fopen, fread, fwrite, fseek, fclose

fin

Next up: Matt Gardine will talk about using the Matlab-Antelope interface