Beyond the Mouse

MATLAB Input & Output

(Getting data into MATLAB and plotting it)

The goal

Spend less time doing stuff computers are good at, and more time doing science (i.e. stuff you can publish).

OR

A program that generates all the figures you need for a paper (or a chapter of your thesis). New dataset -> rerun program -> new paper.

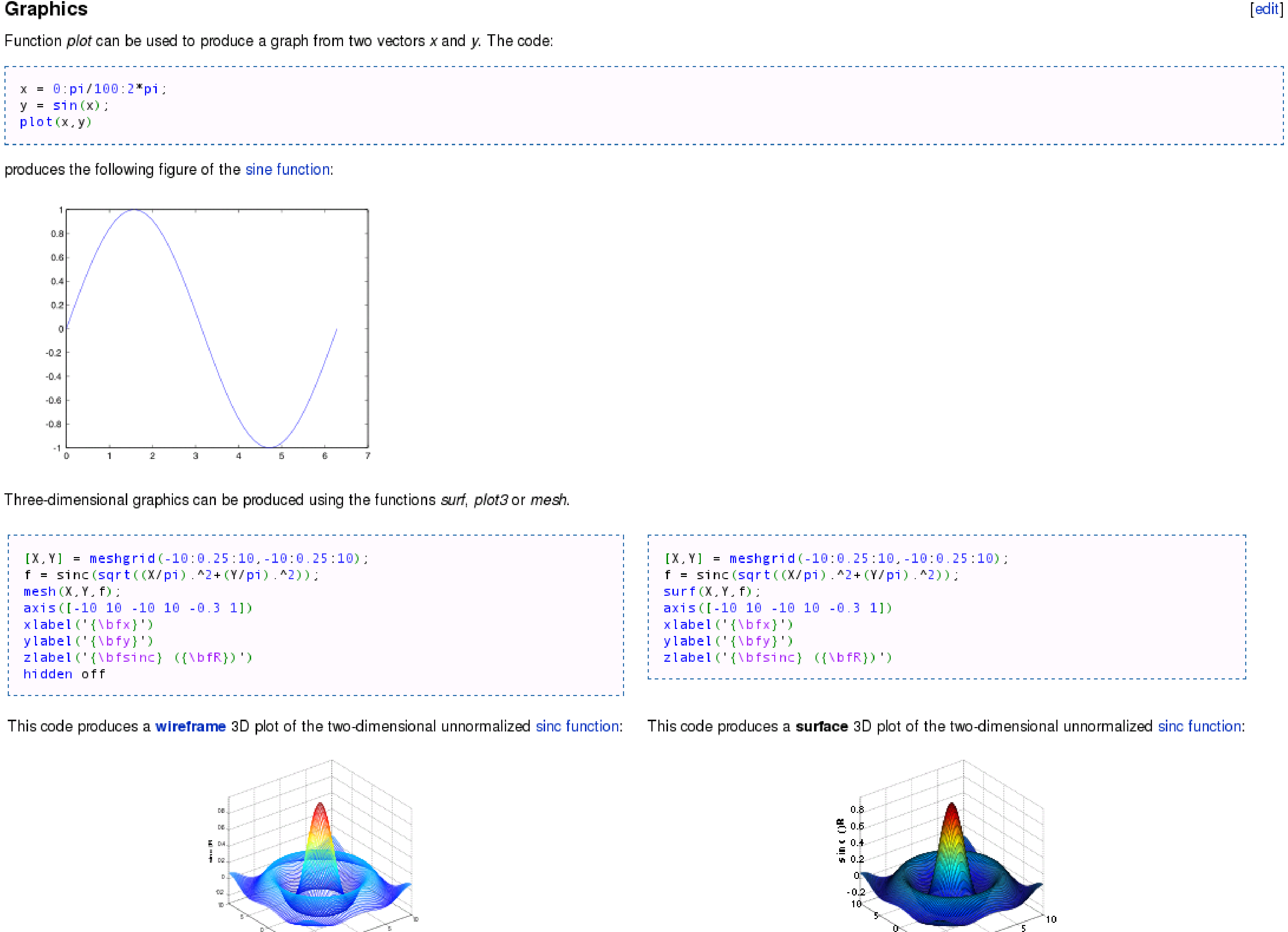
EFFICIENCY / PRODUCTIVITY

Today’s schedule

1. Plotting data with MATLAB
2. Annotating plots (xlabel, ylabel, legend, …)
3. Multiple plots on a figure
4. Saving figures
5. Getting data into MATLAB
6. Miscellaneous
7. Examples
8. Exercises

1. Plotting data with MATLAB

plot



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)
```

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:

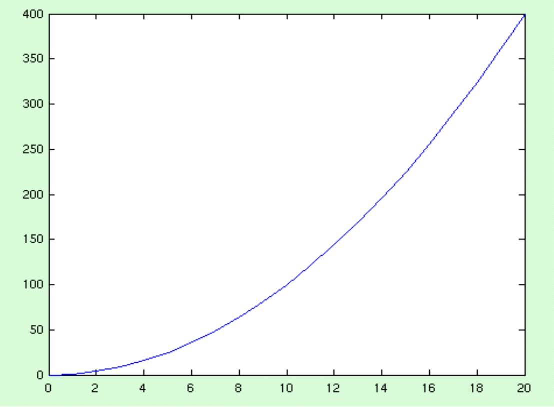
```
[X,Y] = meshgrid(-10:0.25:10,-10:0.25:10);
f = sinc(sqrt((X/pi).^2+(Y/pi).^2));
surf(X,Y,f);
axis([-10 10 -10 10 -0.3 1])
xlabel('{\bfx}')
ylabel('{\bfy}')
zlabel('{\bfsinc} ({\bfR})')
```

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

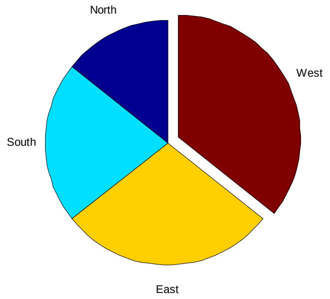
2D plotting

1. Define x-vector
2. Define y-vector
3. plot(x,y)

```
>> x = 1:20;
>> y = x^2;
>> plot(x, y)
```



plot just gives a normal x-y graph with linear axes.

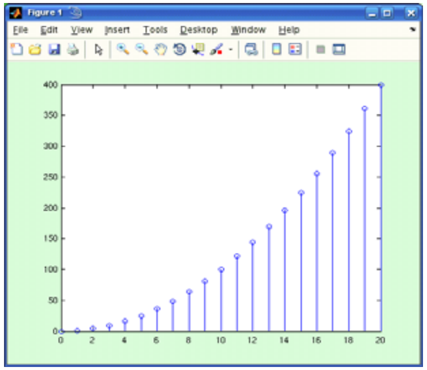


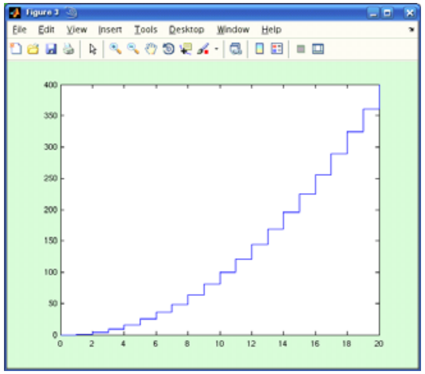
There are other 2D plotting commands, e.g:

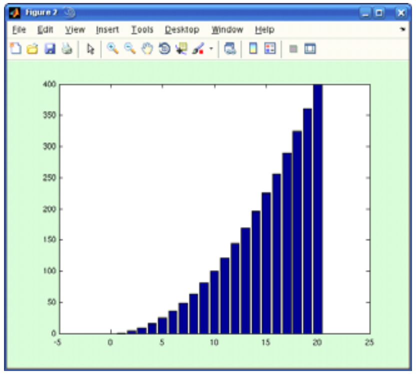
semilogy, semilogx, loglog

stem, stairs, bar

pie, hist

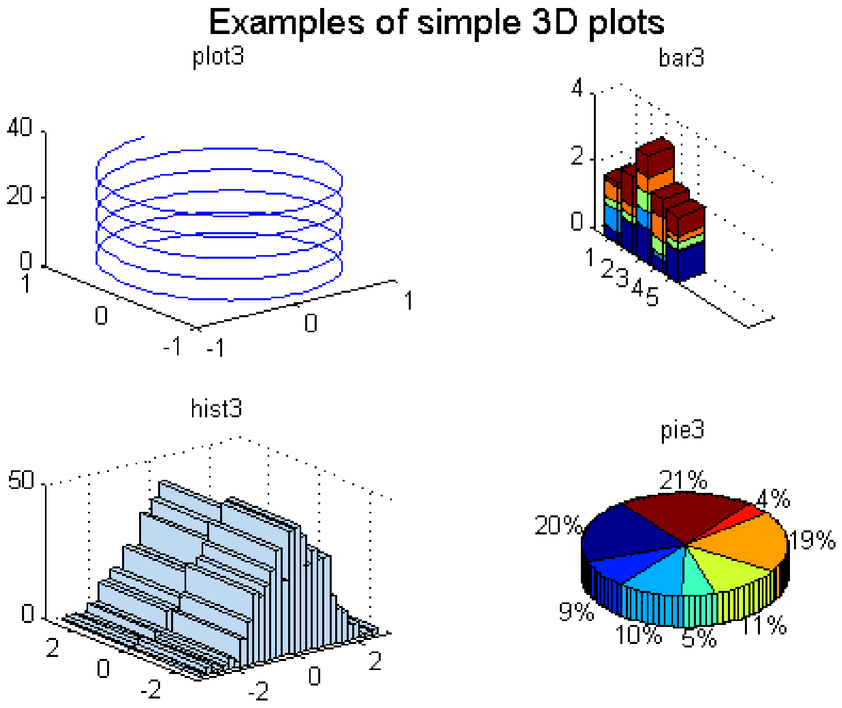






3D plotting

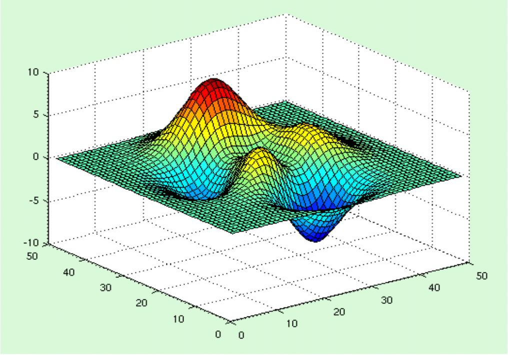
1. Define x-vector
2. Define y-vector
3. Define z-vector
4. plot3(x,y,z)

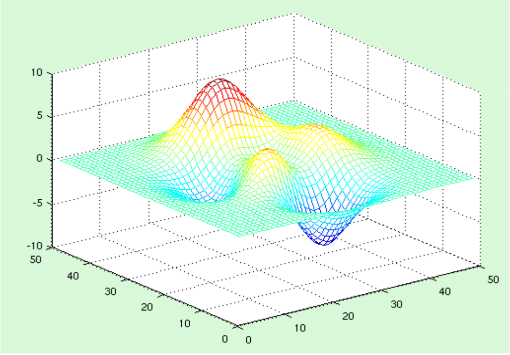


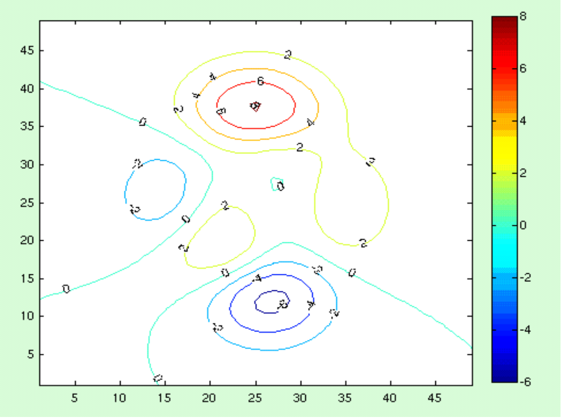
There are other 3D plotting commands, e.g:

surf, mesh, contour

pie3, bar3, hist3





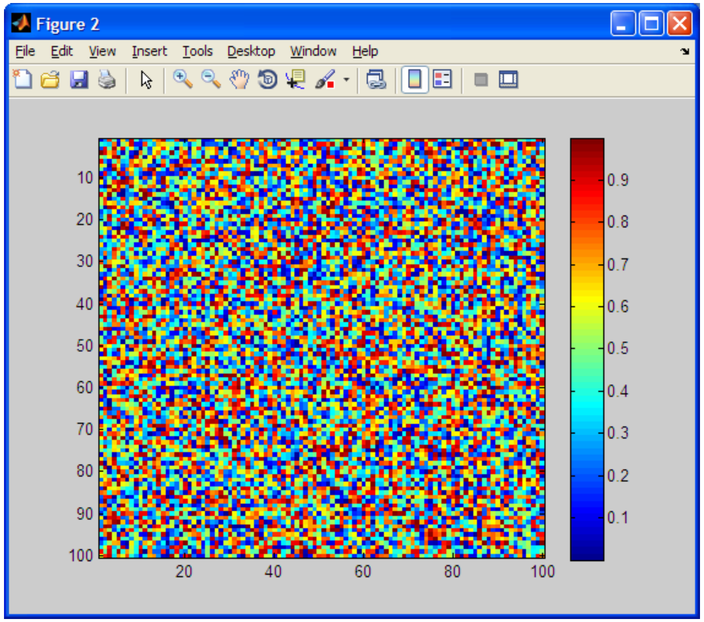


3D plotting – 3rd dimension as color

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

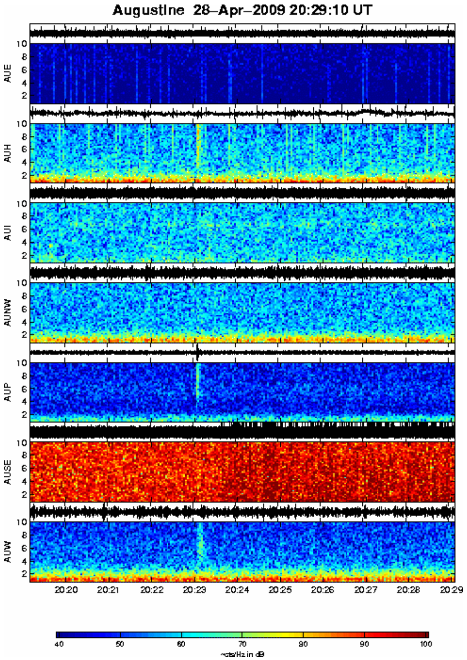
Example:

```
>> 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).



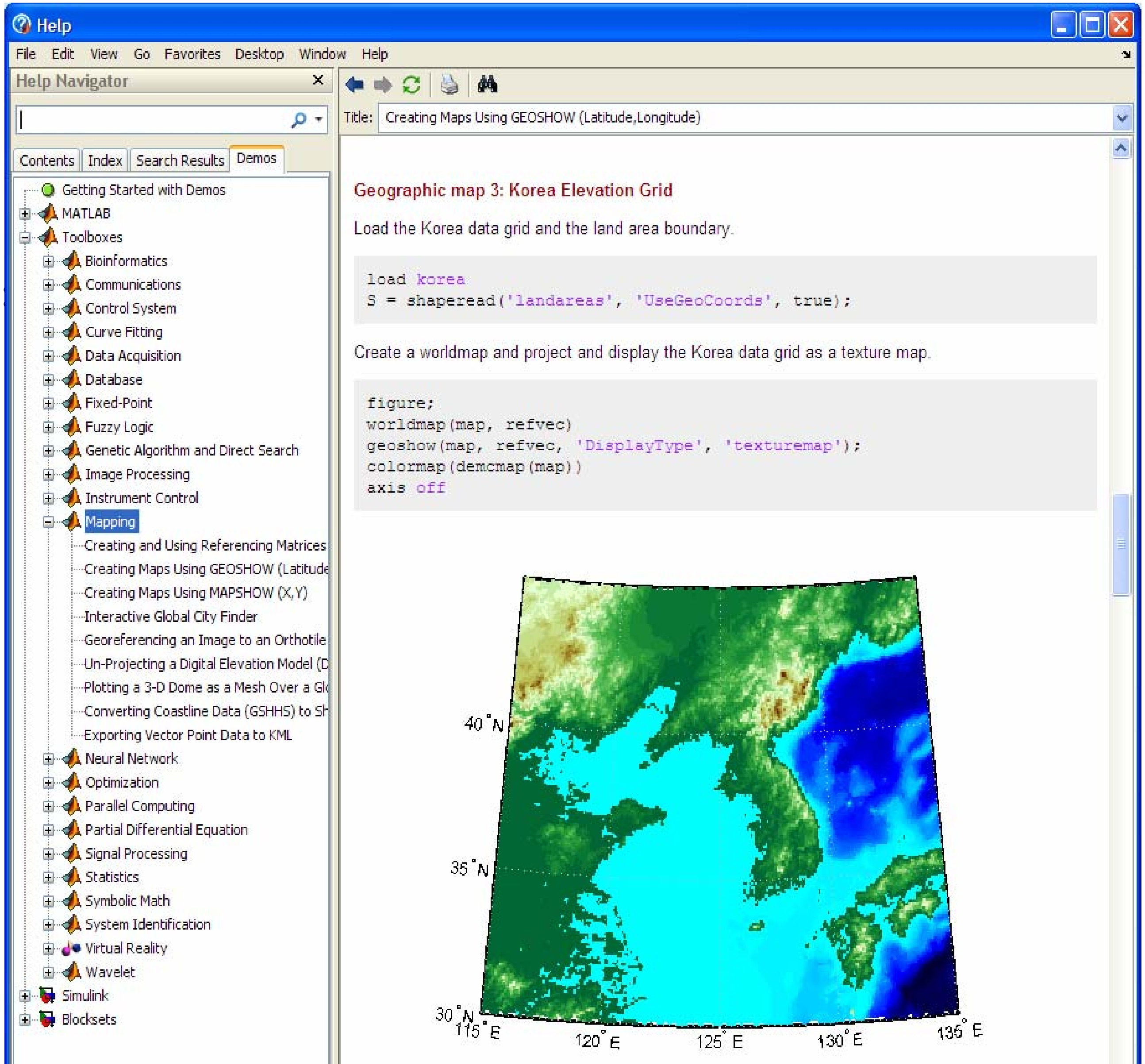
Plotting maps: the Mapping Toolbox

```
>>  help map
>> mapdemos
```

Can write KML (GoogleEarth)

```
>> help kmlwrite
```

Alternative to GMT


Help
File Edit View Go Favorites Desktop Window Help
Help Navigator
Contents
Index
Search Results
Demos
Getting Started with Demos
MATLAB
Toolboxes
Bioinformatics
Communications
Control System
Curve Fitting
Data Acquisition
Database
Fixed-Point
Fuzzy Logic
Genetic Algorithm and Direct Search
Image Processing
Instrument Control
Mapping
Creating and Using Referencing Matrices
Creating Maps Using GEOSHOW (Latitude
Creating Maps Using MAPSHOW (X,Y)
Interactive Global City Finder
Georeferencing an Image to an Orthotile
Un-Projecting a Digital Elevation Model (D
Plotting a 3-D Dome as a Mesh Over a Gl
Converting Coastline Data (GSHHS) to Sh
Exporting Vector Point Data to KML
Neural Network
Optimization
Parallel Computing
Partial Differential Equation
Signal Processing
Statistics
Symbolic Math
System Identification
Virtual Reality
Wavelet
Simulink
Blocksets
Title: Creating Maps Using GEOSHOW (Latitude,Longitude)

Geographic map 3: Korea Elevation Grid

Load the Korea data grid and the land area boundary.

```
load korea
S = shaperead('landareas', 'UseGeoCoords', true);
```

Create a worldmap and project and display the Korea data grid as a texture map.

```
figure;
worldmap(map, refvec)
geoshow(map, refvec, 'DisplayType', 'texturemap');
colormap(demcmap(map))
axis off
```

40°N
35°N
30°N
115°E
120°E
125°E
130°E
135°E

2. Annotating plots

Changing the line style: plot(x,y,s)

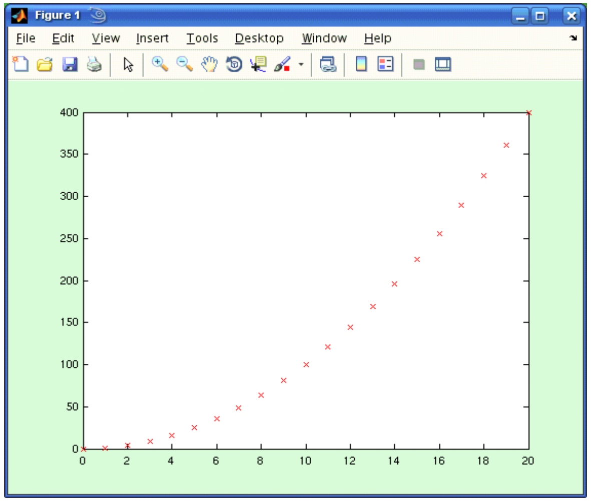
By default, plot(x,y) uses a blue line to connect data points

>> help plot

```
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
```

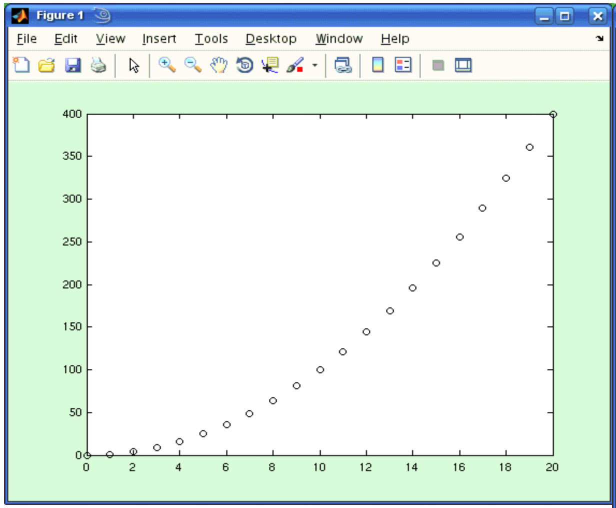
plot(x,y,s)

plot(x,y,'rx')



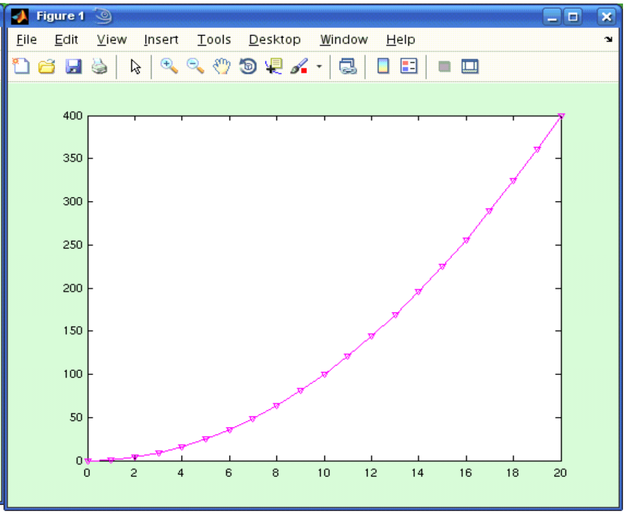
red crosses

plot(x,y,'bo')



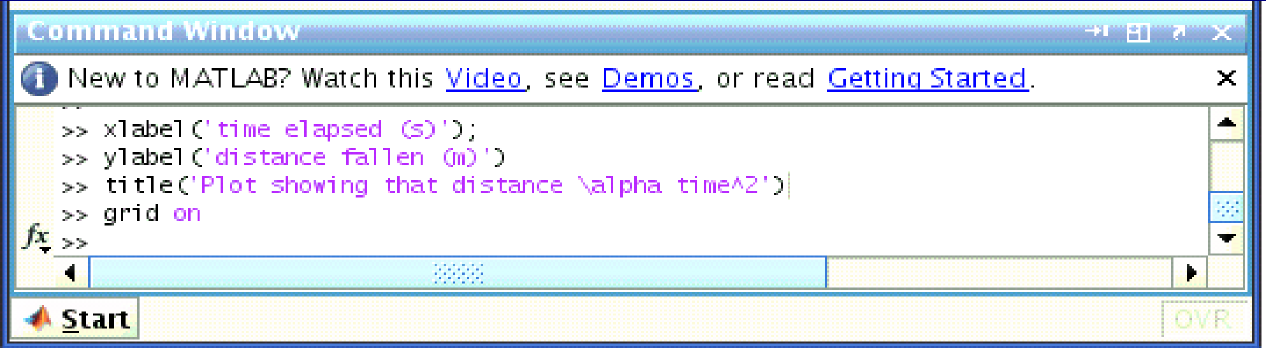
black circles

plot(x, y, 'mv-')



magenta triangles + line

Labelling axes

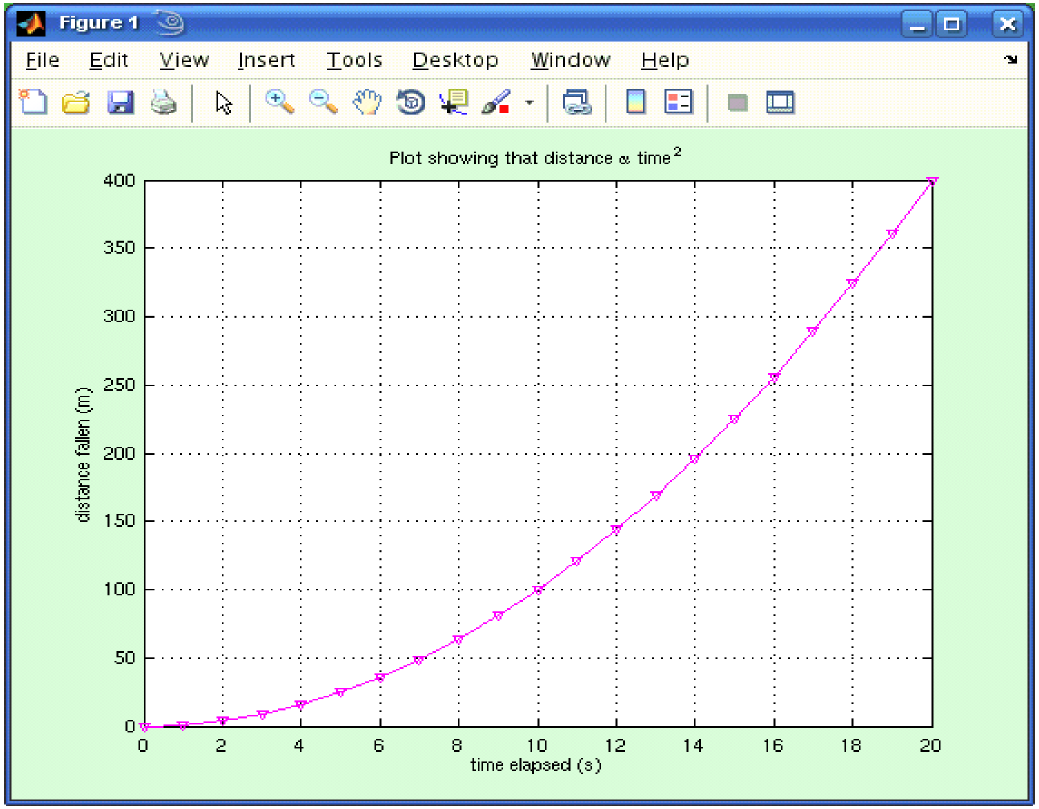


New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
>> 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



Adding text

To add text at the position xpos, ypos to the current axes use:

```
>> text(xpos, ypos, ‘some_string’);
```

Remember you can use sprintf.

```
>> text(2.3, 5.1, sprintf(‘station %s’,station{stationNum}) );
```

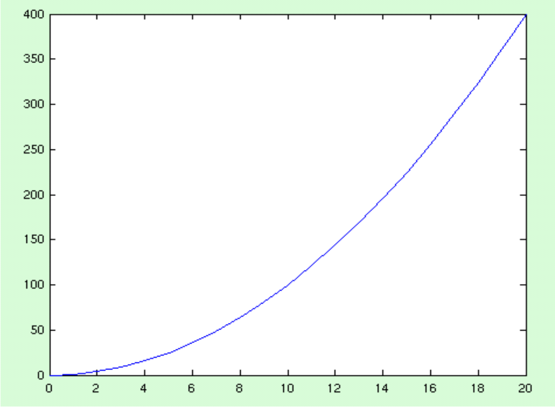
Changing the data range shown

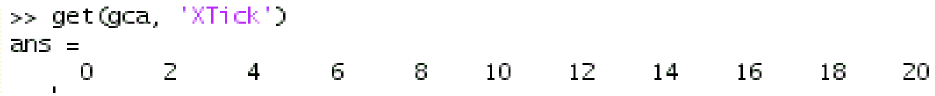
Default: show all the data.

To override 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
```

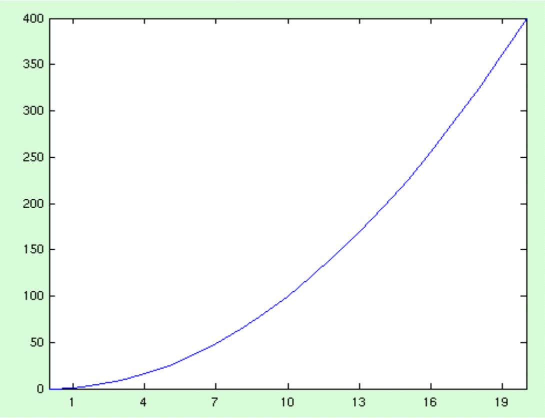
Changing the tick positions/labels



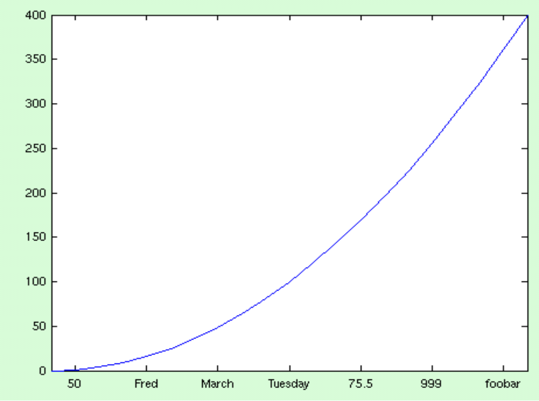


```
>> get(gca, 'XTick')
ans =
        0        2        4        6        8       10       12       14       16       18       20
```

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



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



Plotting against date/time: datenum & datetick

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.
        datetick(‘x’); % can give unexpected results, ask for help.
```

datestr

I often use dates in plot labels, or in file paths/names.

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
>> xlabel( sprintf(‘This plot was generated at %s’, datestr(now, 31) ) );
```

An aside – making dates work for you:

YYYYMMDD, not MMYYDD (U.S.) or DDMMYY (Europe).

3. Multiple plots on a figure

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 axes plot

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)

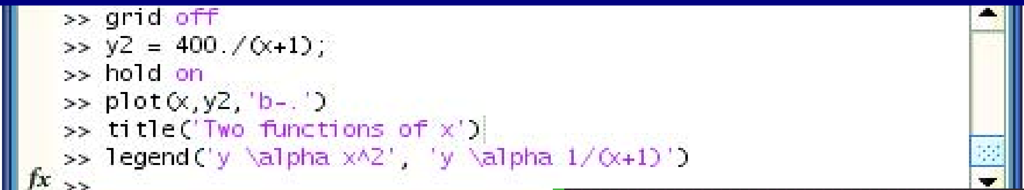
>> get(gca, ‘position’)
This will return the screen position of the current active figure window, which by default is:
[0.13 0.11 0.775 0.815]
Format here is [xorigin yorigin xwidth yheight] in fractions of the figure window width.

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:

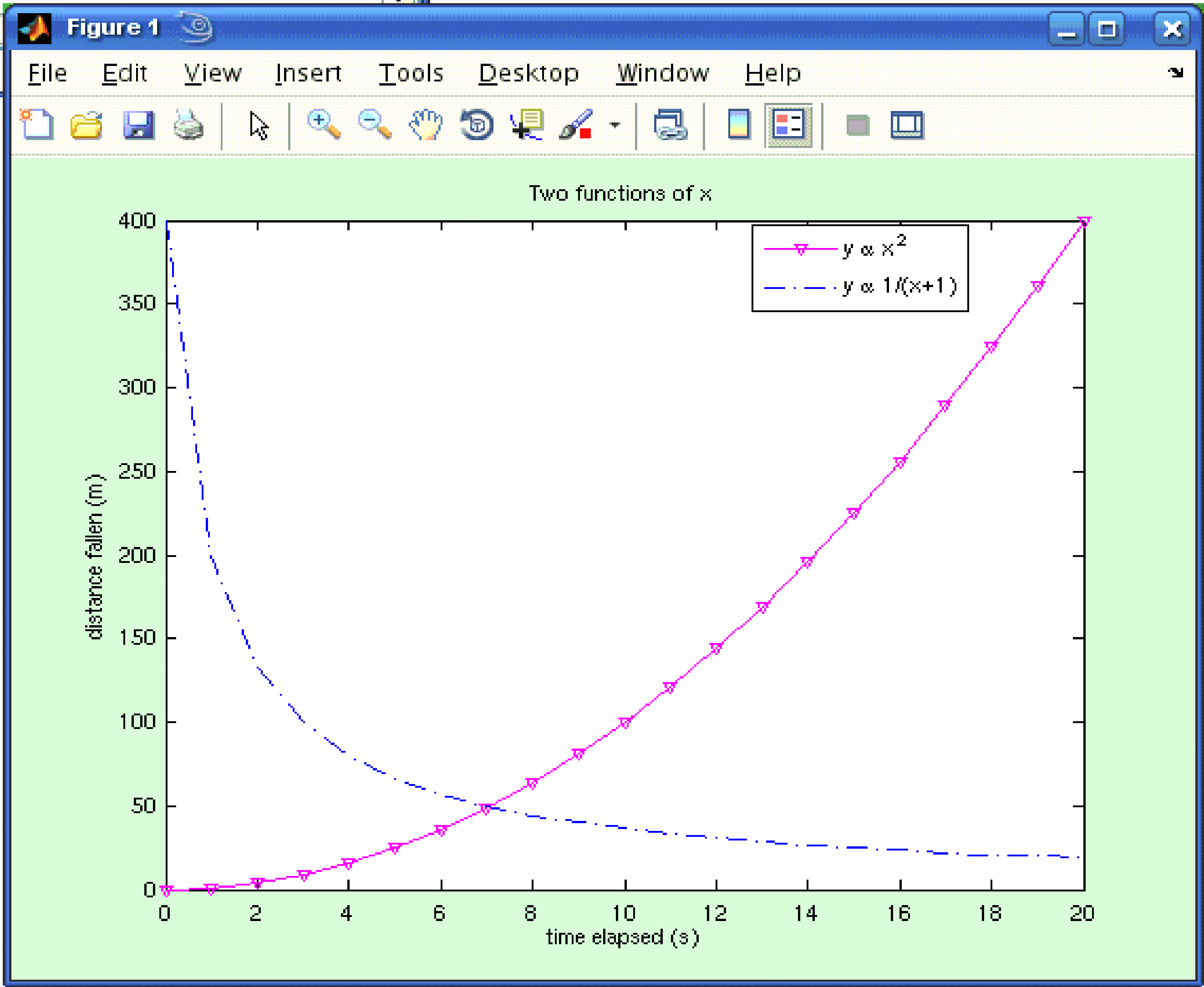
>> axes(‘position’, [0.2 0.3 0.6 0.4]);
Either will create a figure if none already exists. Or modify the current set of axes on the current figure.

Multiple plots on a figure 1: 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)')
>>
```

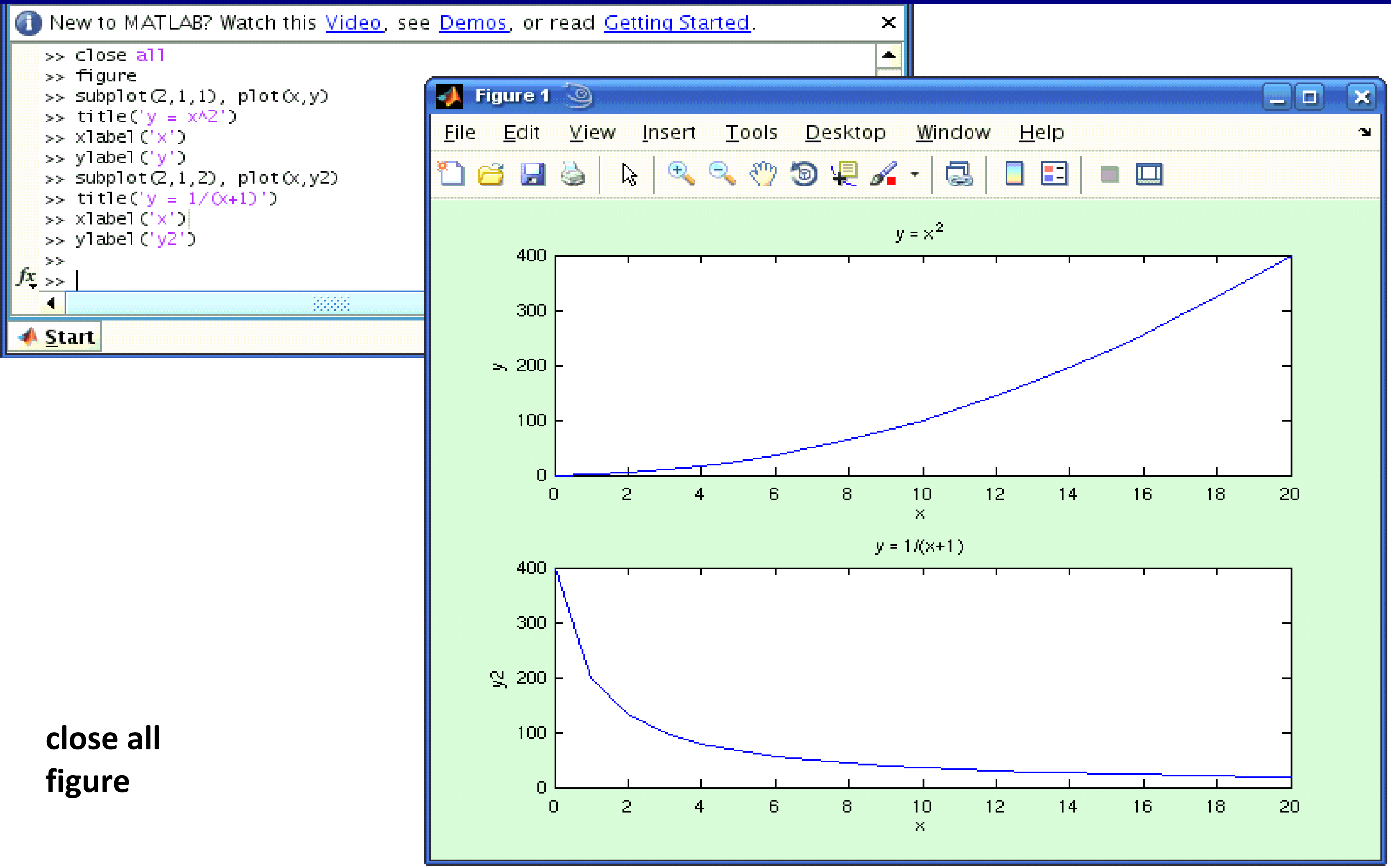
hold on “holds on” to graphs already in the current axes.
Normally they would be erased



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 2: subplot



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

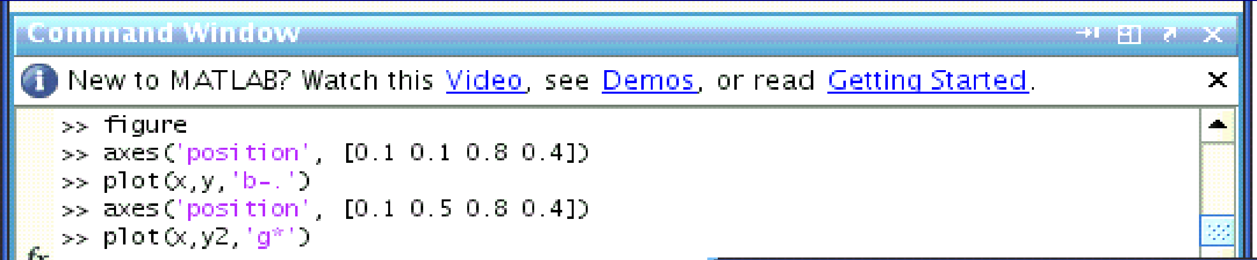
```
>> close all
>> figure
>> subplot(2,1,1), plot(x,y)
>> title('y = x^2')
>> xlabel('x')
>> ylabel('y')
>> subplot(2,1,2), plot(x,y2)
>> title('y = 1/(x+1)')
>> xlabel('x')
>> ylabel('y2')
>>
>> |
```

close all

figure

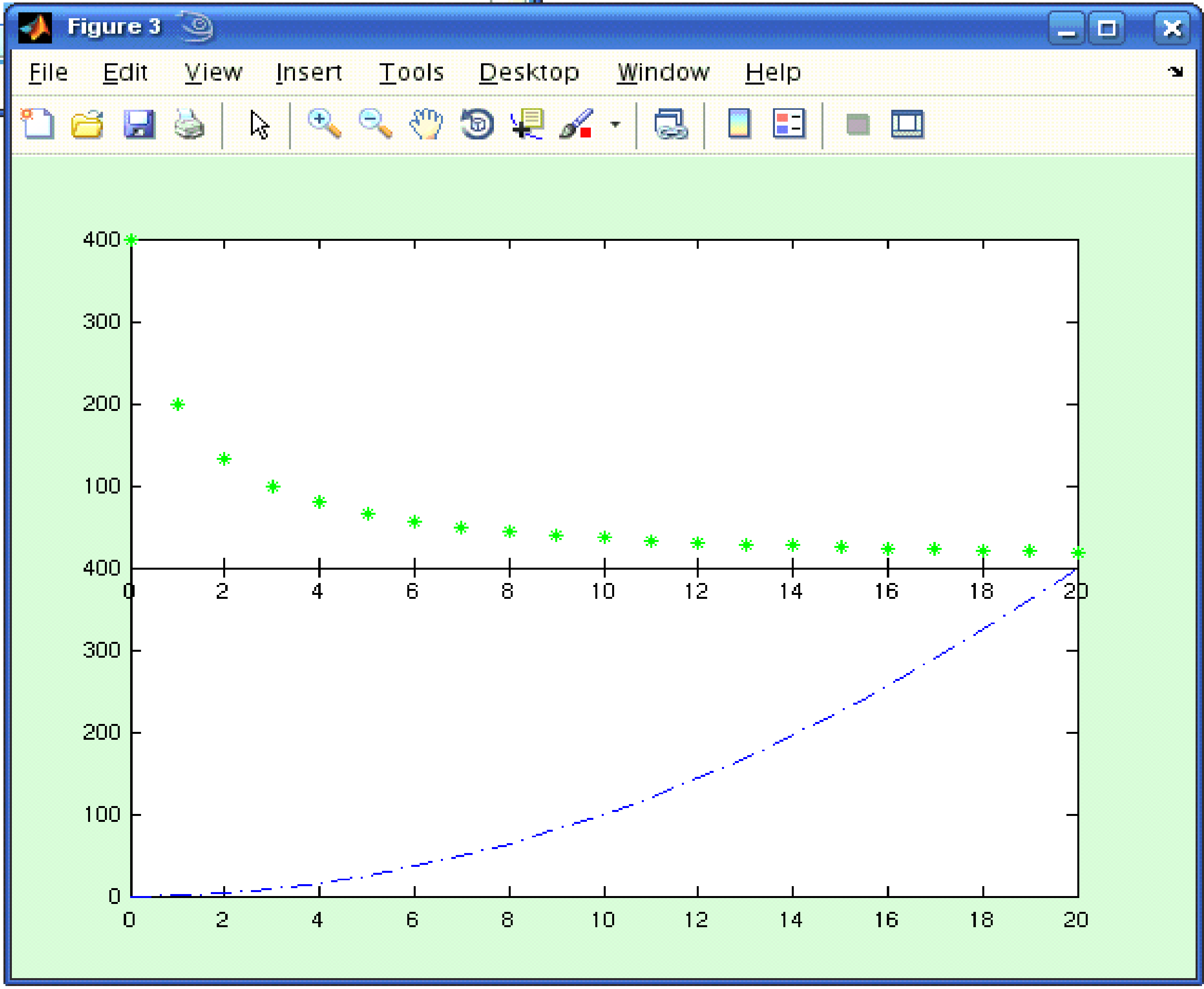
subplot(M, N, plotnum) - an M x N array of plot axes

Multiple plots on a figure 3: axes('position', […])



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*')
>>
```



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

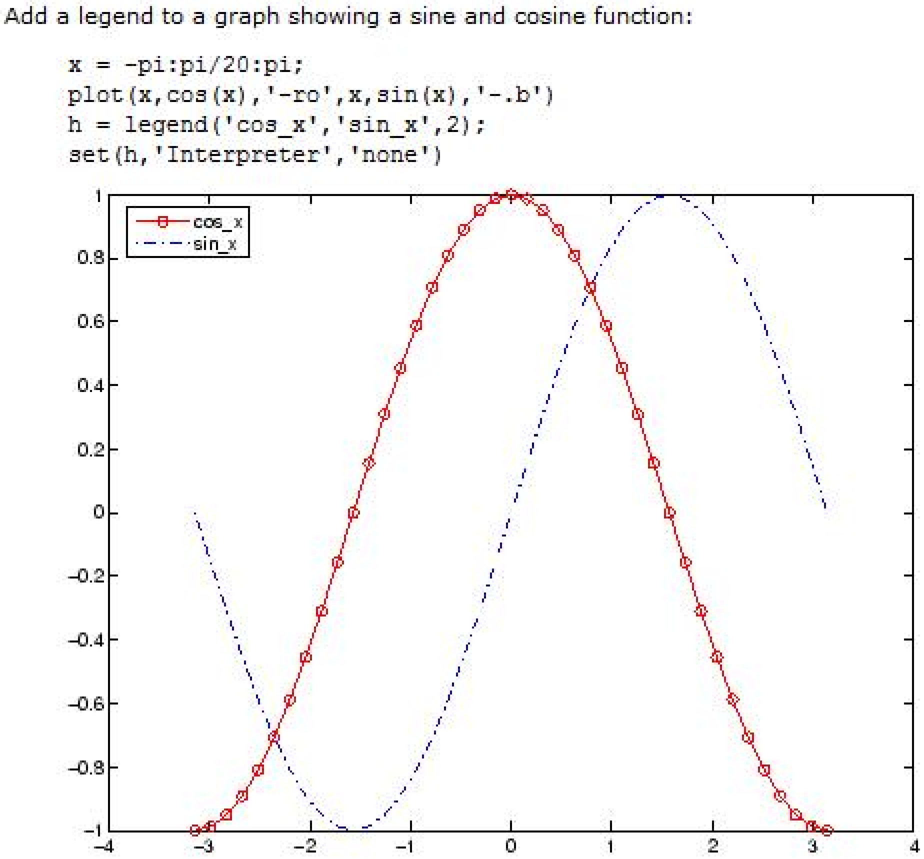
set(gca, 'XTickLabel', {})
- remove x tick labels

Multiple plots on a figure 4: long form of plot command

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 lin styles.

You can then use **legend** to create a key for the different graphs in your figure.



Add a legend to a graph showing a sine and cosine function:

```
x = -pi:pi/20:pi;
plot(x,cos(x),'-ro',x,sin(x),'-.b')
h = legend('cos_x','sin_x',2);
set(h,'Interpreter','none')
```

4. Savings figures to image files

Writing an image file - print

print -f1 -dpng myplotfilename.png - script form

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

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.

Writing an image file - print

Example:

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
```

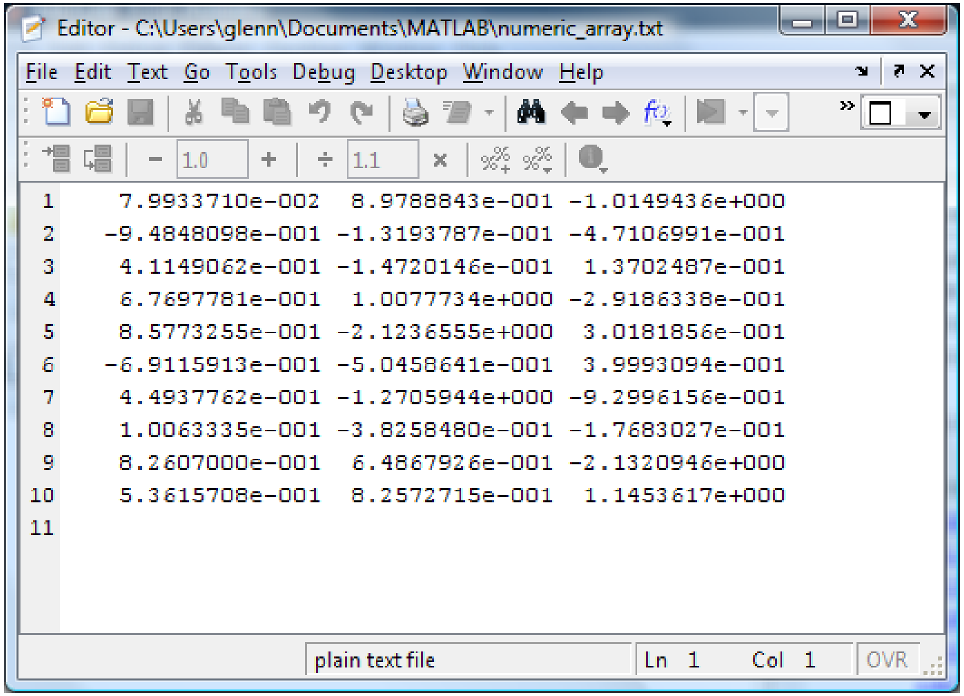
For plotNum = 2, the print line would evaluate to:

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

5. Reading (and writing) data from files

load

- **Load data from an ASCII file into an array (must look like an array)**


Editor - C:\Users\glenn\Documents\MATLAB\numeric_array.txt
File Edit Text Go Tools Debug Desktop Window Help
1.0
1.1
plain text file
Ln 1 Col 1
OVR

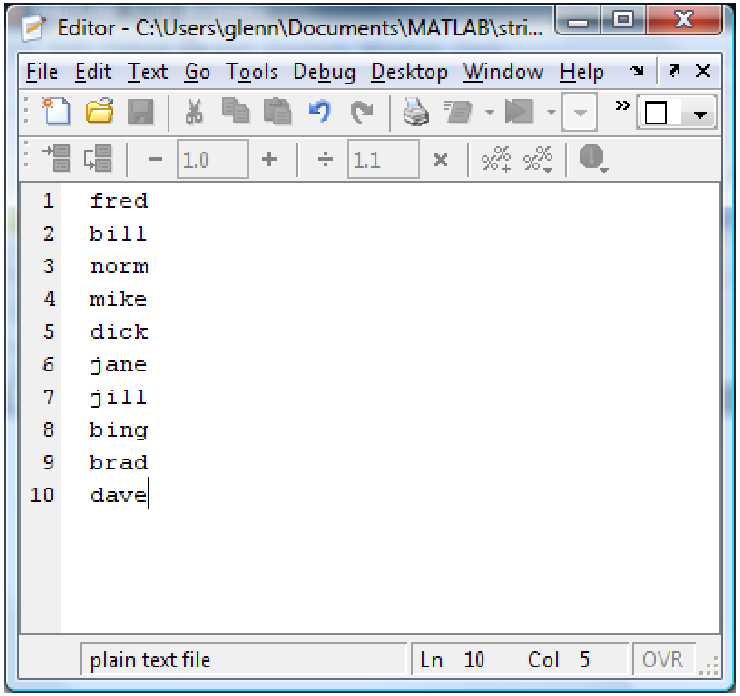
```
1     7.9933710e-002  8.9788843e-001 -1.0149436e+000
2    -9.4848098e-001 -1.3193787e-001 -4.7106991e-001
3     4.1149062e-001 -1.4720146e-001  1.3702487e-001
4     6.7697781e-001  1.0077734e+000 -2.9186338e-001
5     8.5773255e-001 -2.1236555e+000  3.0181856e-001
6    -6.9115913e-001 -5.0458641e-001  3.9993094e-001
7     4.4937762e-001 -1.2705944e+000 -9.2996156e-001
8     1.0063335e-001 -3.8258480e-001 -1.7683027e-001
9     8.2607000e-001  6.4867926e-001 -2.1320946e+000
10    5.3615708e-001  8.2572715e-001  1.1453617e+000
11
```

```
>> 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
```

```
??? Error using ==> load
Number of columns on line 5 of ASCII file numeric_array.txt
must be the same as previous lines.
```

- **Load variables from a MATLAB binary file (*.mat)**

load() wont work at all with alphabetic characters



Editor - C:\Users\glenn\Documents\MATLAB\stri...
File Edit Text Go Tools Debug Desktop Window Help
1.0
1.1

```
1  fred
2  bill
3  norm
4  mike
5  dick
6  jane
7  jill
8  bing
9  brad
10 dave
```

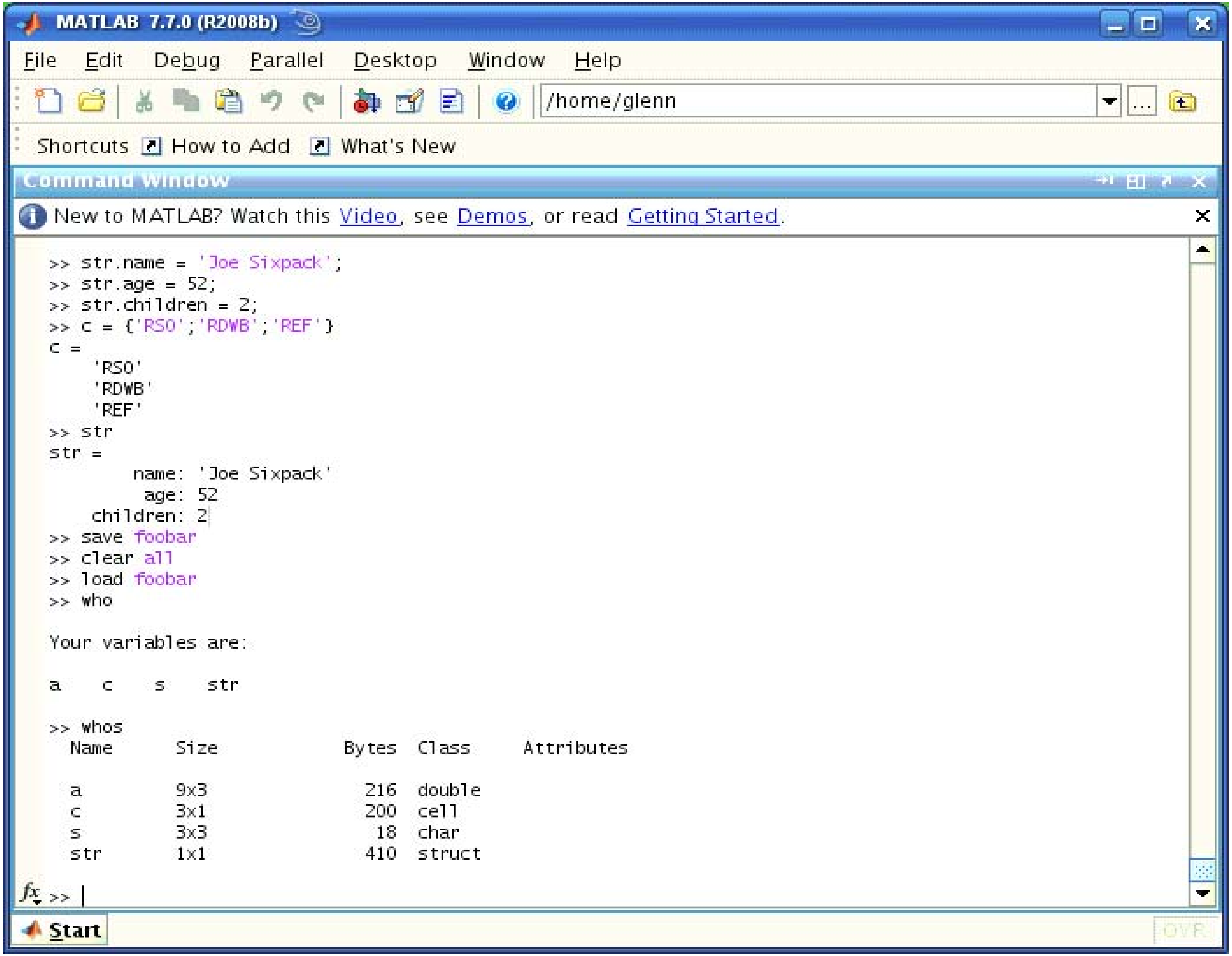
plain text file   Ln 10   Col 5   OVR

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

MATLAB binary files

Only MATLAB can read/write them. Useful for storing (workspace) variables, so you can reload them later. Use **save** and **load**. Support numeric arrays, strings, cell arrays and structs.

```
>> 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
```



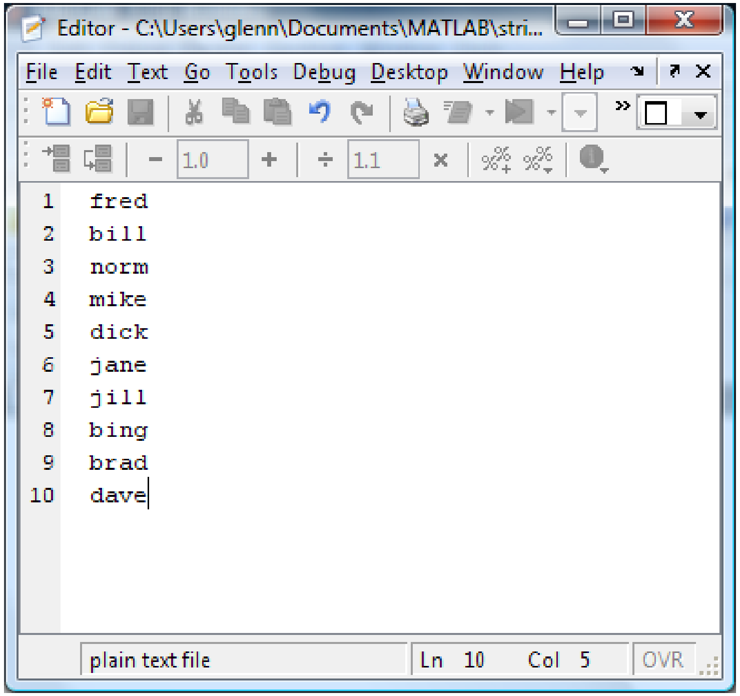
```
>> str.name = 'Joe Sixpack';
>> str.age = 52;
>> str.children = 2;
>> c = {'RSO';'RDWB';'REF'}
c =
    'RSO'
    'RDWB'
    'REF'
>> str
str =
        name: 'Joe Sixpack'
         age: 52
    children: 2
>> save foobar
>> clear all
>> load foobar
>> who

Your variables are:

a    c    s    str

>> whos
  Name      Size            Bytes  Class     Attributes

  a         9x3               216  double
  c         3x1               200  cell
  s         3x3                18  char
  str       1x1               410  struct

>> |
```

importdata

load wont work with strings. A more versatile function is:

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

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



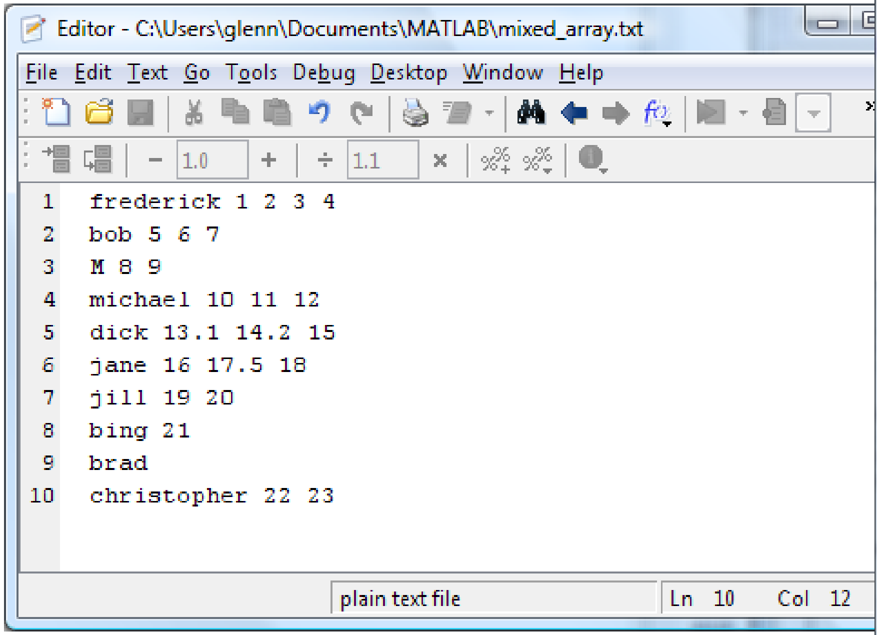
Editor - C:\Users\glenn\Documents\MATLAB\stri...
File Edit Text Go Tools Debug Desktop Window Help
1.0
1.1
plain text file
Ln 10 Col 5
OVR

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

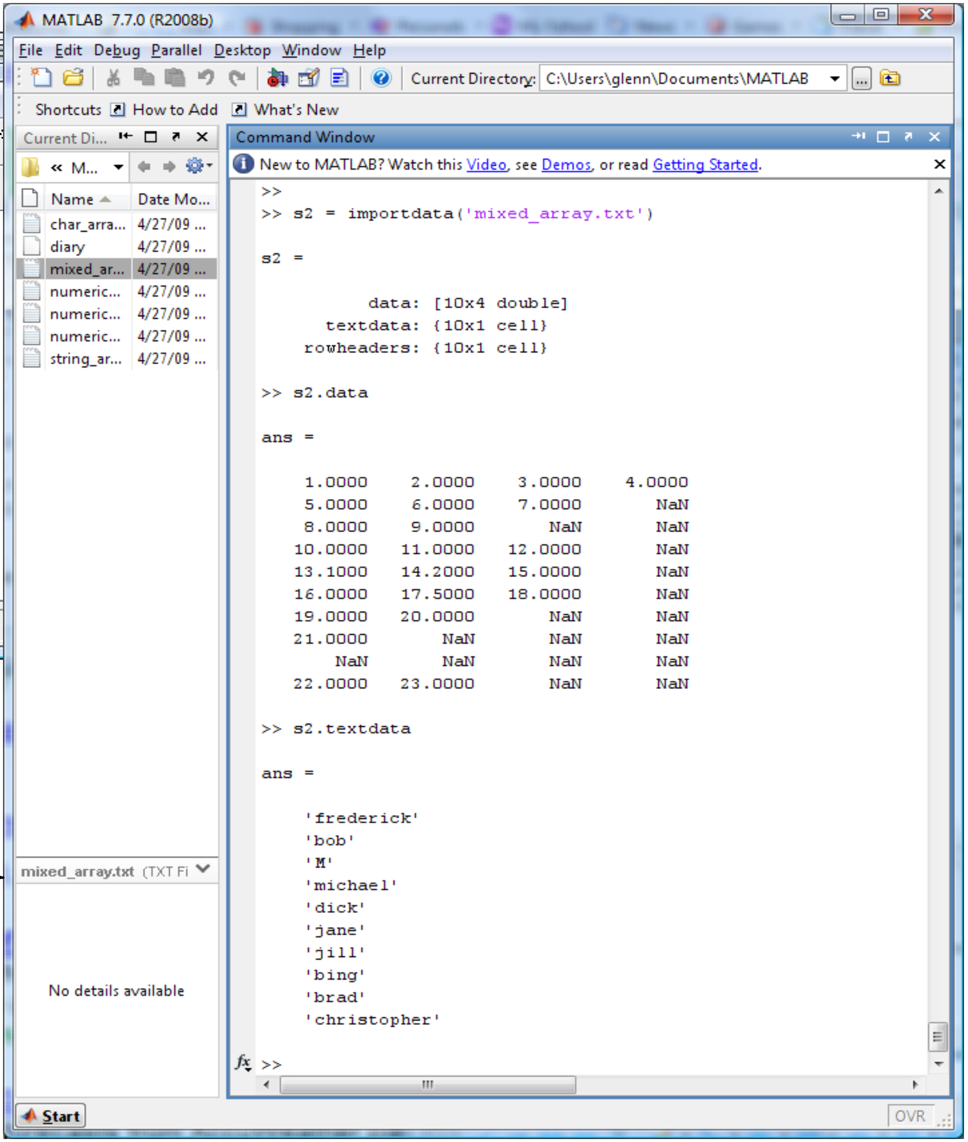
```
>> a=importdata('numeric_array3.txt')

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'

>>
```

Loads string_array.txt into a cell array

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
File Edit Text Go Tools Debug Desktop Window Help
1.0
1.1
plain text file
Ln 10 Col 12

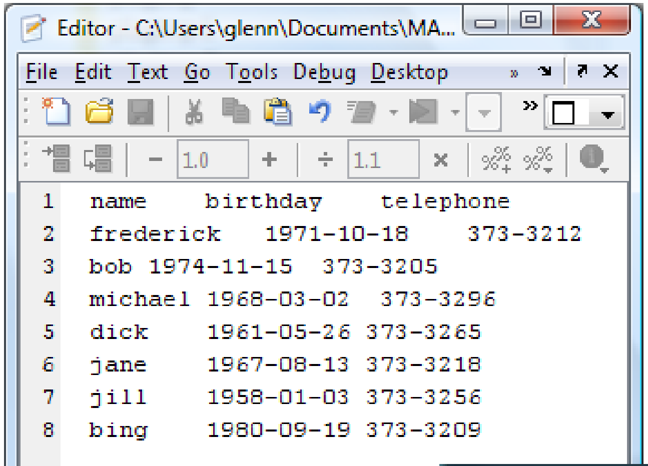
```
frederick 1 2 3 4
bob 5 6 7
M 8 9
michael 10 11 12
dick 13.1 14.2 15
jane 16 17.5 18
jill 19 20
bing 21
brad
christopher 22 23
```



```
>>
>> 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

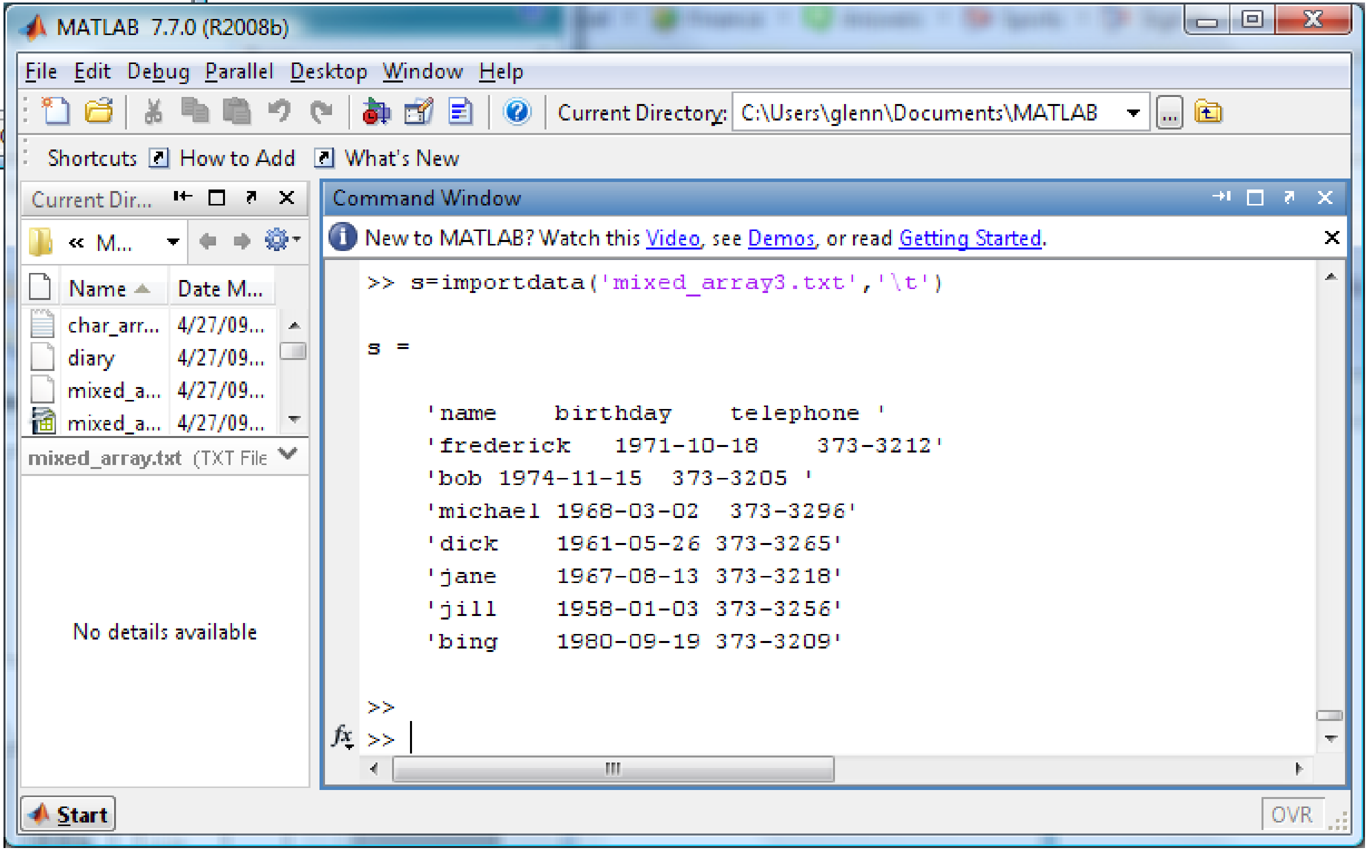
Non-existent values replaced with NaN in numeric array


Editor - C:\Users\glenn\Documents\MA...
File Edit Text Go Tools Debug Desktop
1.0
1.1
ain text file
Ln 3

```
name	birthday	telephone
frederick	1971-10-18	373-3212
bob 1974-11-15  373-3205
michael 1968-03-02  373-3296
dick    1961-05-26 373-3265
jane    1967-08-13 373-3218
jill    1958-01-03 373-3256
bing    1980-09-19 373-3209
```

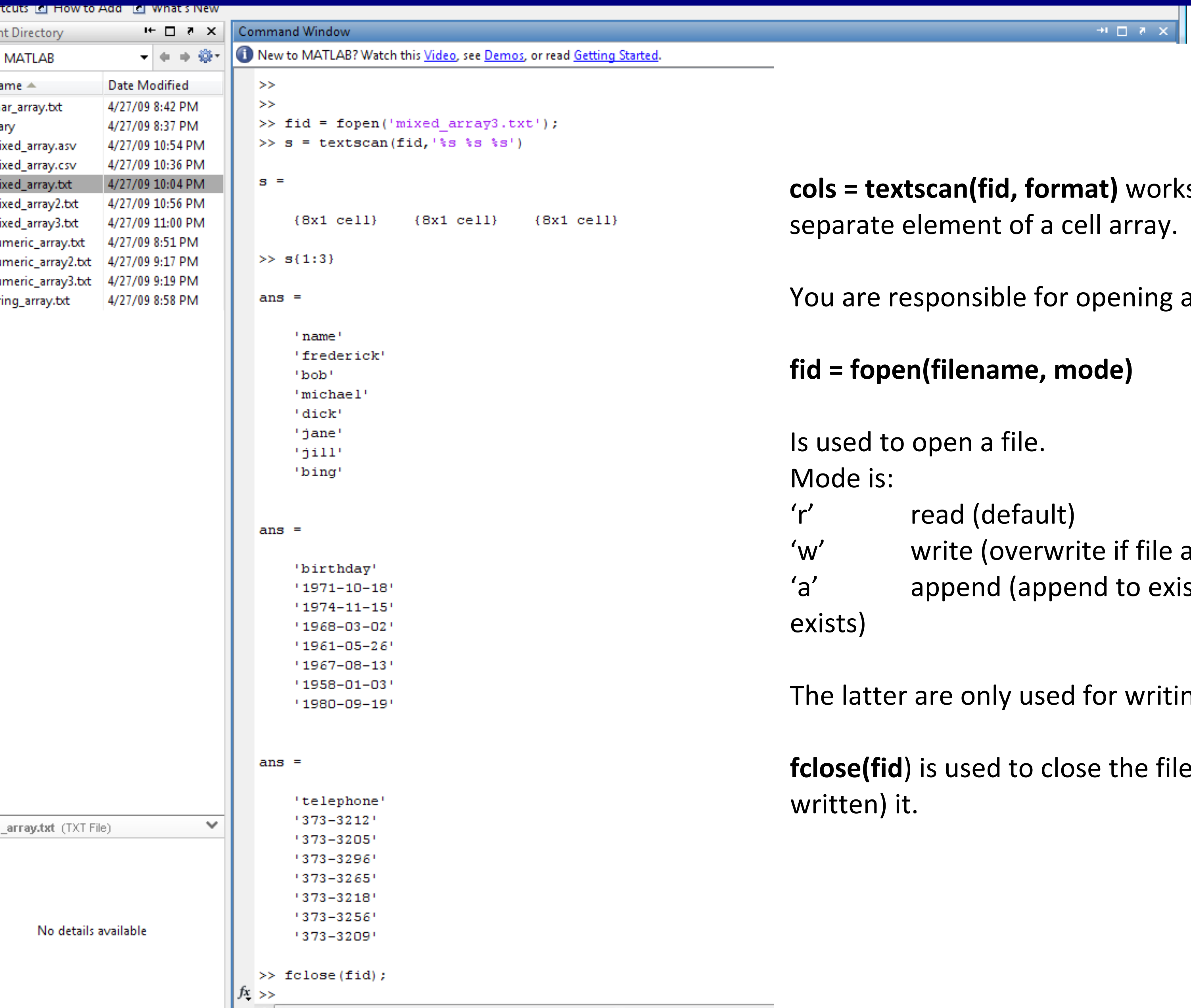
But importdata finally fails to work as desired when we are reading in a simple file made of 3 strings per row.

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


MATLAB 7.7.0 (R2008b)
File Edit Debug Parallel Desktop Window Help
Current Directory: C:\Users\glenn\Documents\MATLAB
Shortcuts How to Add What's New
Current Dir...
« M...
Name
Date M...
char_arr... 4/27/09...
diary 4/27/09...
mixed_a... 4/27/09...
mixed_a... 4/27/09...
mixed_array.txt (TXT File
No details available
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
Start
OVR

```
>> s=importdata('mixed_array3.txt','\t')

s = 

    'name    birthday    telephone '
    'frederick    1971-10-18    373-3212'
    'bob 1974-11-15  373-3205 '
    'michael 1968-03-02  373-3296'
    'dick    1961-05-26 373-3265'
    'jane    1967-08-13 373-3218'
    'jill    1958-01-03 373-3256'
    'bing    1980-09-19 373-3209'

>> 
>> 
```

textscan



ame	Date Modified
ar_array.txt	4/27/09 8:42 PM
ary	4/27/09 8:37 PM
ixed_array.asv	4/27/09 10:54 PM
ixed_array.csv	4/27/09 10:36 PM
ixed_array.txt	4/27/09 10:04 PM
ixed_array2.txt	4/27/09 10:56 PM
ixed_array3.txt	4/27/09 11:00 PM
imeric_array.txt	4/27/09 8:51 PM
imeric_array2.txt	4/27/09 9:17 PM
imeric_array3.txt	4/27/09 9:19 PM
ing_array.txt	4/27/09 8:58 PM

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

s =

    {8x1 cell}    {8x1 cell}    {8x1 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)

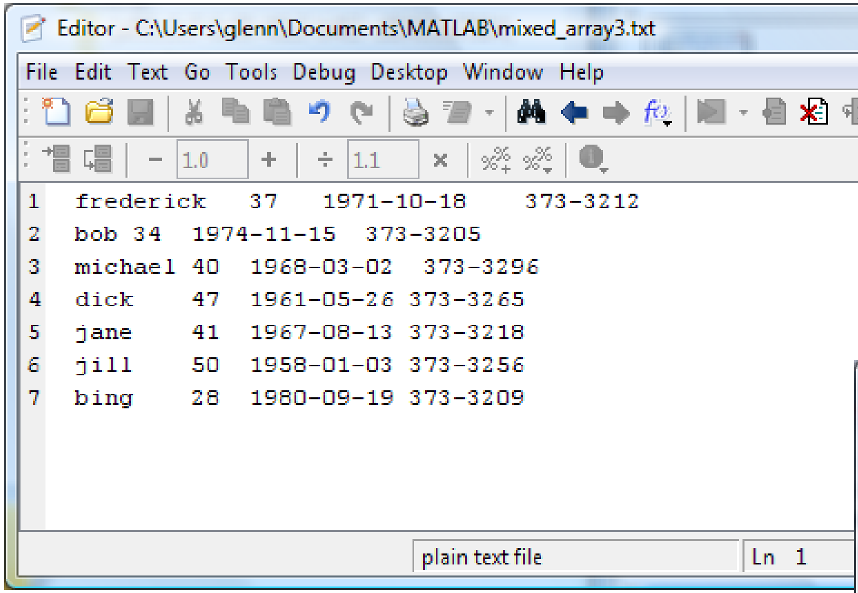
Is used to open a file.
Mode is:
'r' read (default)
'w' write (overwrite if file already exists)
'a' append (append to existing file if it already exists)

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.

Example to plot birthdays against name using:

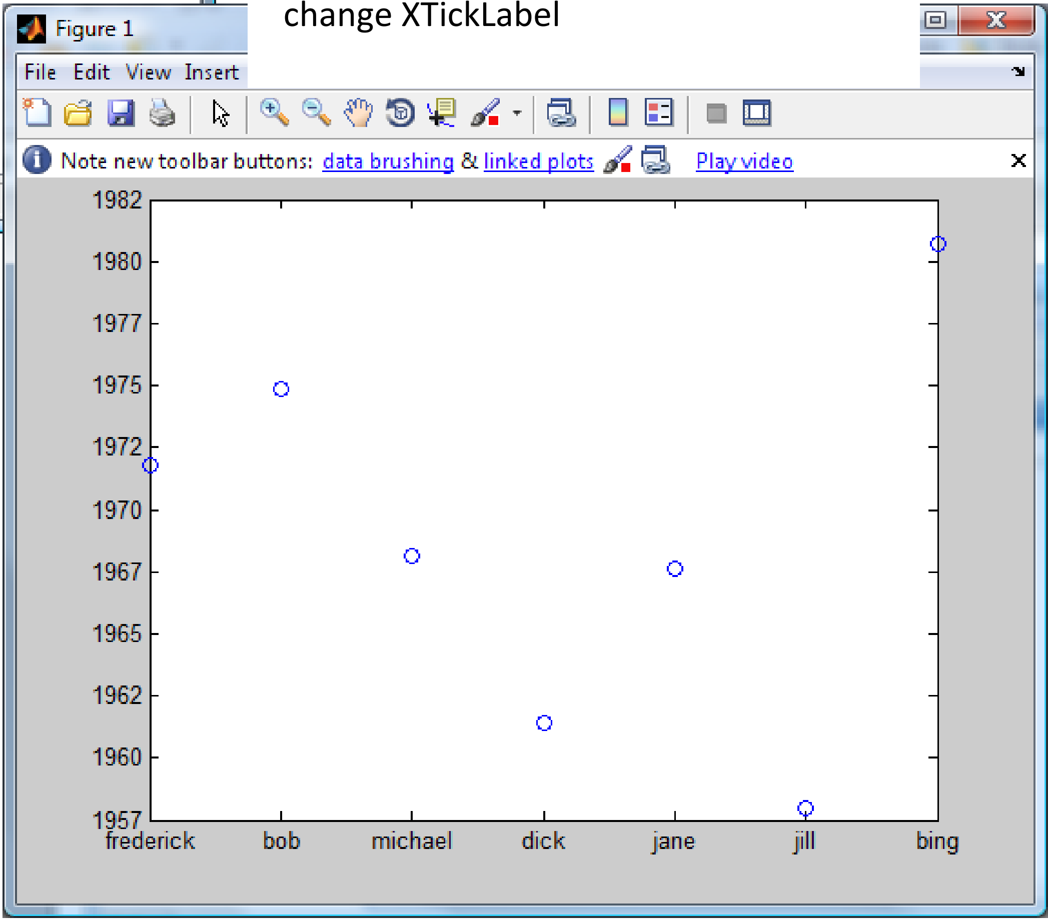
- fopen/textscan/fclose
- struct
- datenum
- plot(y)
- datetick
- change XTickLabel


Editor - C:\Users\glenn\Documents\MATLAB\mixed_array3.txt
File Edit Text Go Tools Debug Desktop Window Help
1.0
1.1
plain text file
Ln 1

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

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
```

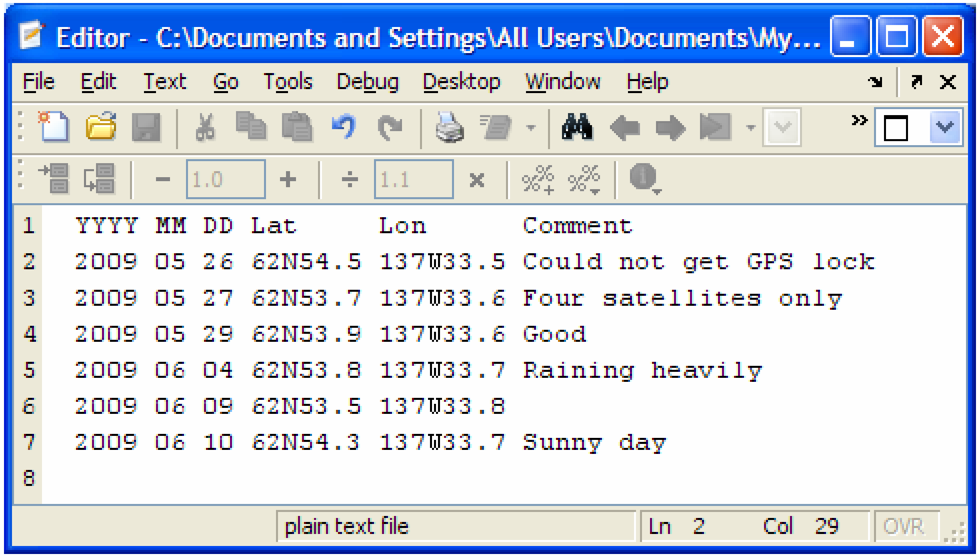


Read a line - fgetl

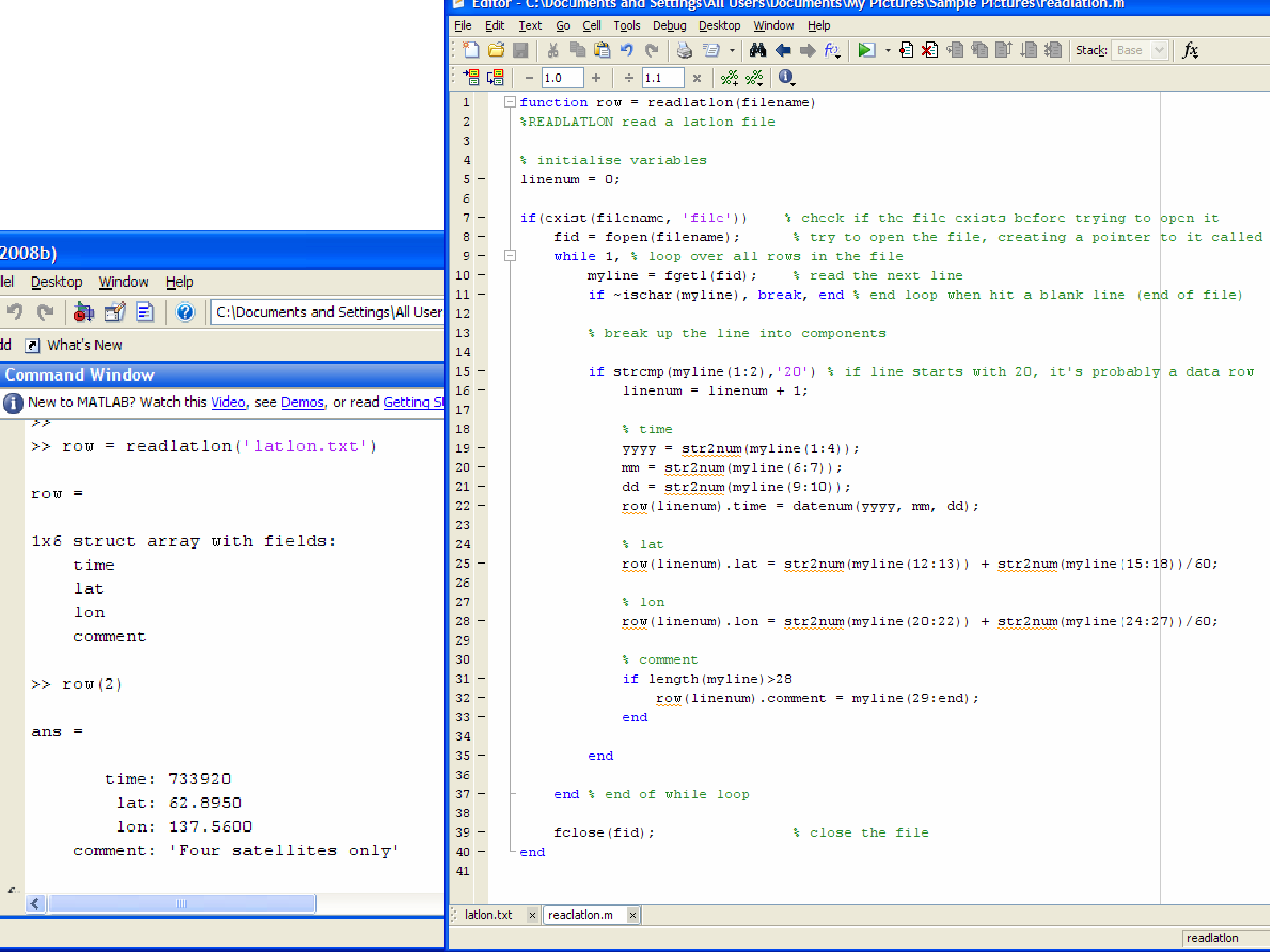
Line can be any length, any format.

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

Useful when each line has fields which appear in fixed positions.


Editor - C:\Documents and Settings\All Users\Documents\My...
File Edit Text Go Tools Debug Desktop Window Help
1.0
1.1
plain text file
Ln 2 Col 29
OVR

```
YYYY MM DD Lat     Lon      Comment
2009 05 26 62N54.5 137W33.5 Could not get GPS lock
2009 05 27 62N53.7 137W33.6 Four satellites only
2009 05 29 62N53.9 137W33.6 Good
2009 06 04 62N53.8 137W33.7 Raining heavily
2009 06 09 62N53.5 137W33.8
2009 06 10 62N54.3 137W33.7 Sunny day
```

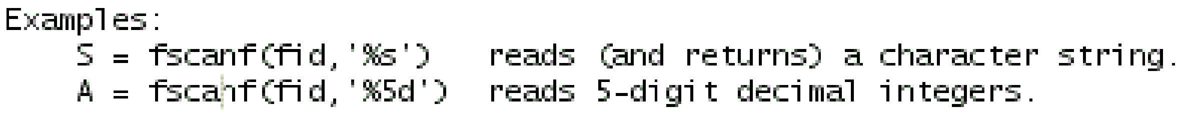


```
function row = readlatlon(filename)
%READLATLON read a latlon file

% 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
    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

        end

    end % end of while loop

    fclose(fid);                 % close the file
end

```

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

```
>>
>> 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'
```

Read a data type - fscanf



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

Writing to a file - fprintf

```
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)

Read an Excel file - xlsread

[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

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

Writing an Excel file - xlswrite

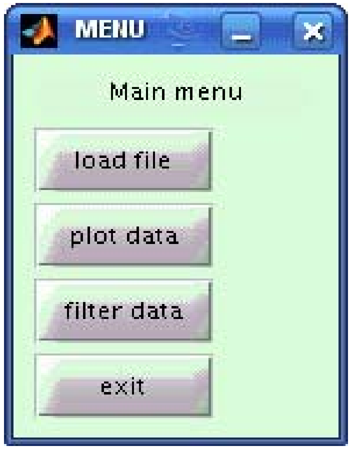
```
xlswrite('myfile.xls', myarray, 'sheet2');
```

myarray - a numeric array or a cell array

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

6. Miscellaneous I/O

Graphical input



```
% A simple menu
choice = 0;
while (choice ~= 4)
        choice = menu('Main menu', 'load file', 'plot data', 'filter data',
'exit')
        switch choice
                case 1, loadFile();
                case 2, plotData();
                case 3, filterData();
        end
end
```

```
% Get filename dialog
[filename, dirname] = uigetfile();
```

```
% Save filename dialog
[filename, dirname] = uiputfile();
```

```
% Getting input coordinates from the mouse
[x, y] = ginput(2);         % input 2 data points
```

- useful for picking P and S arrival times
- or start and end of tremor or swarm episodes
- or start and end of episodes of increased degassing

```
% Designing GUIs
guide;
```

7. 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, importdata, save

Text files:

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

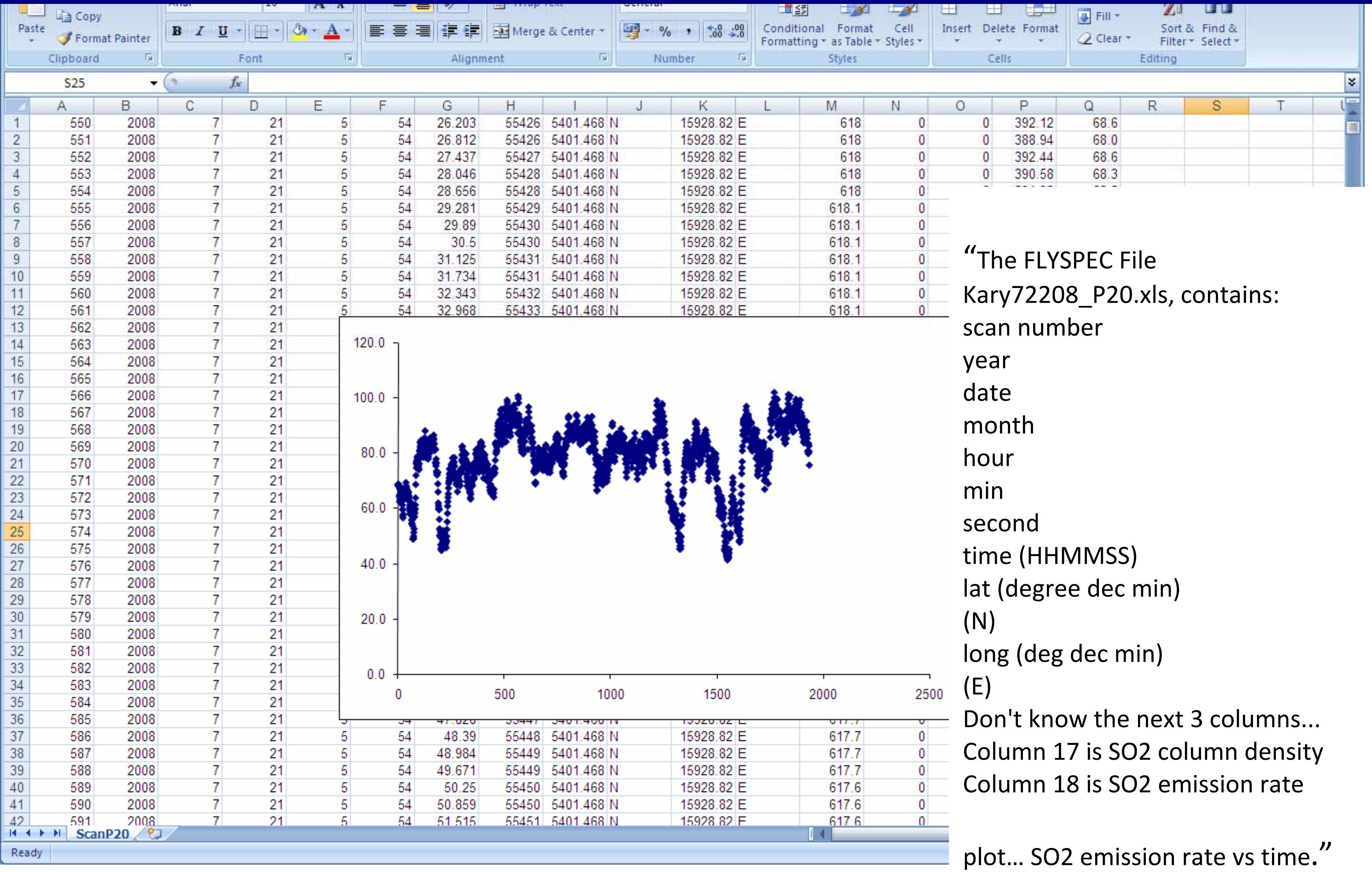
Excel files:

- xlsread, xlswrite

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

8. Examples

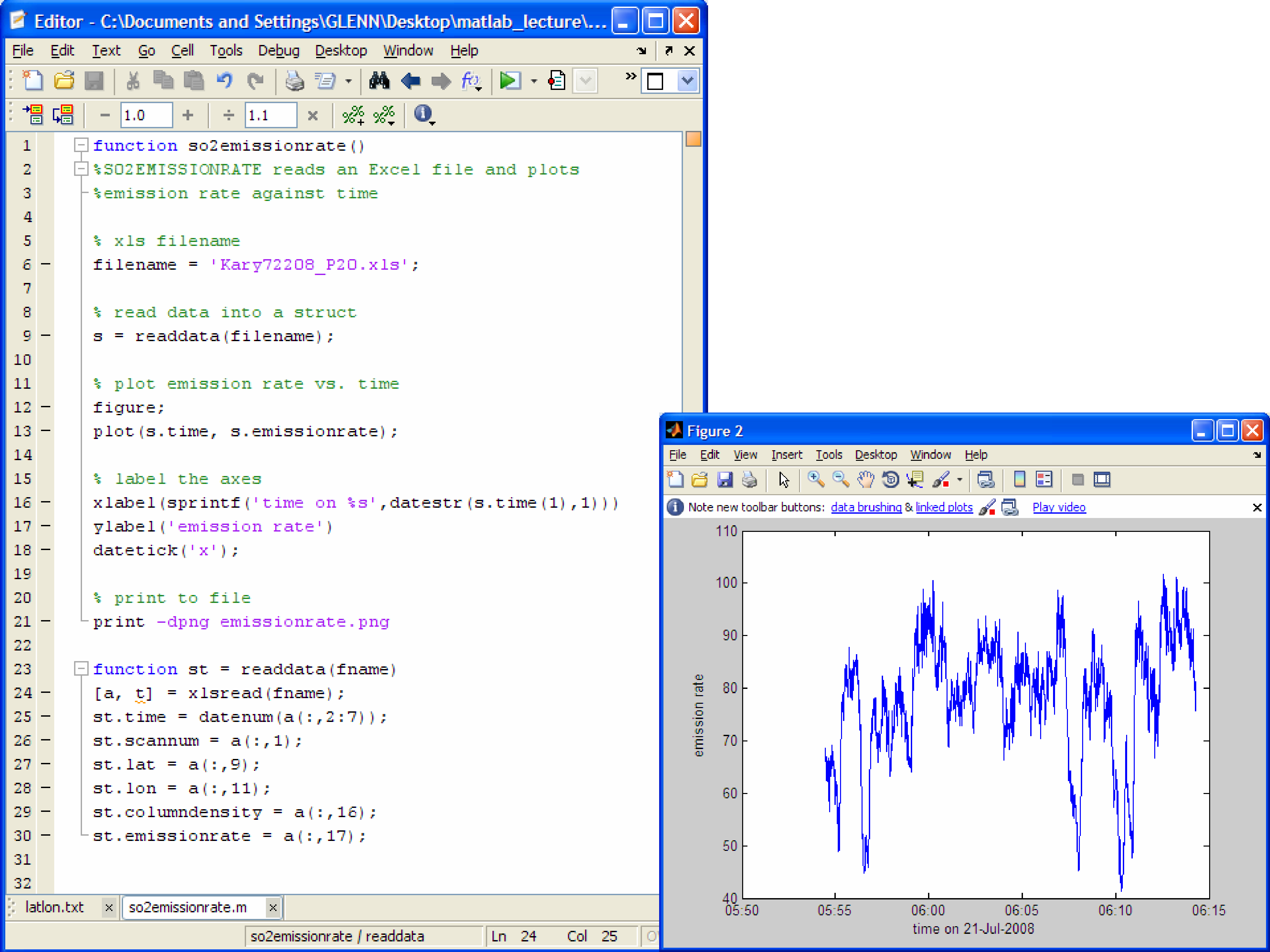
Flyspec data, courtesy of Taryn Lopez



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
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			

"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

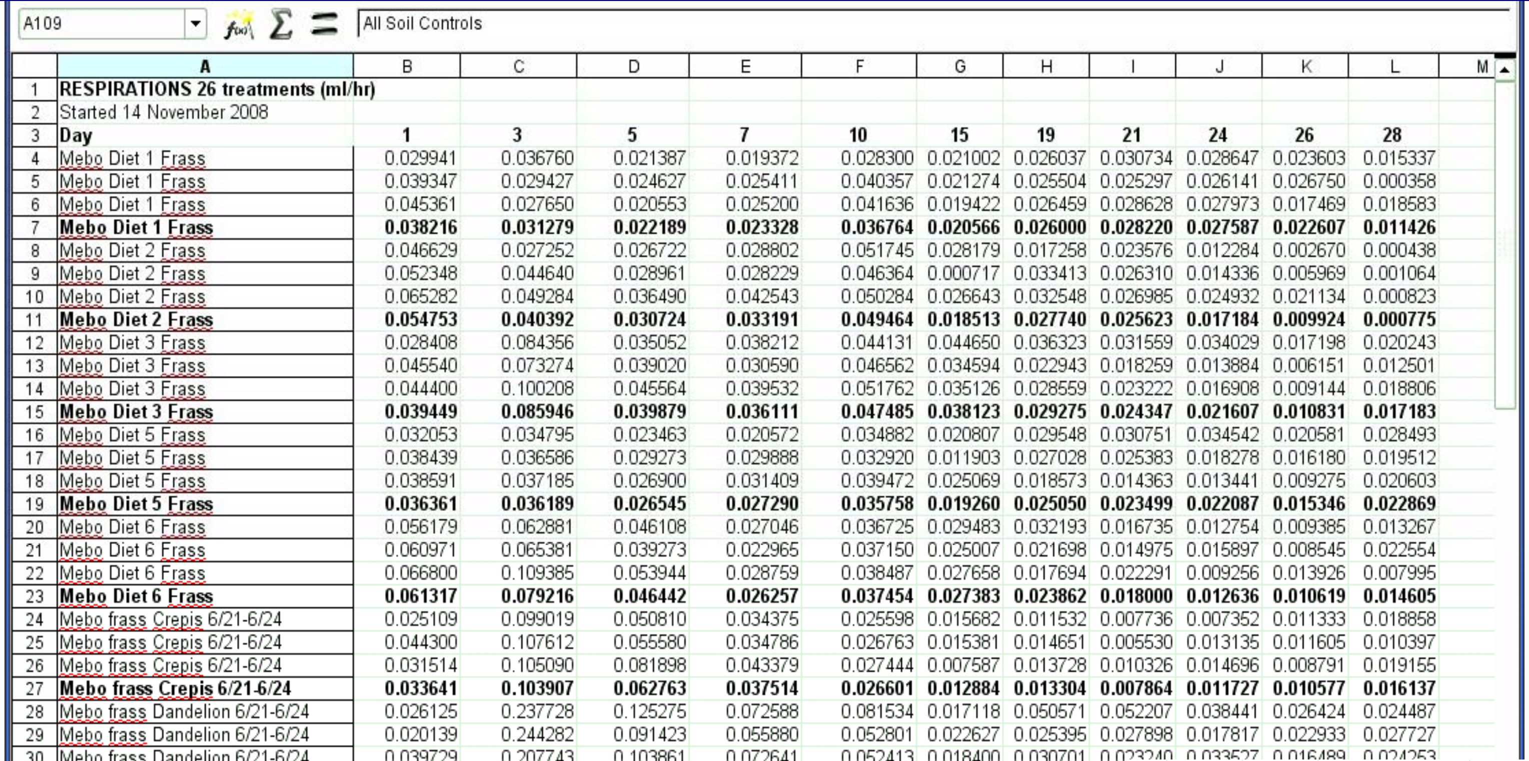
plot… SO2 emission rate vs time."



Editor - C:\Documents and Settings\GLENN\Desktop\matlab_lecture\...
File Edit Text Go Cell Tools Debug Desktop Window Help
1.0
1.1
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
110
100
90
80
70
60
50
40
emission rate
05:50
05:55
06:00
06:05
06:10
06:15
time on 21-Jul-2008

```
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);
```

Grasshopper diet data, courtesy of Ellen Trainor



A109 | All Soil Controls

	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**	**0.0**				
32	Mebo frass Willow 6/21-6/26	0.054596	0.207699	0.079345	0.054584	0.036836	0.015952	0.006870	0.0				
33	Mebo frass Willow 6/21-6/26	0.052565	0.229171	0.120664	0.056337	0.037465	0.012946	0.025919	0.0				
34	Mebo frass Willow 6/21-6/26	0.050980	0.143724	0.146130	0.056944	0.035093	0.015005	0.027627	0.0				
35	**Mebo frass Willow 6/21-6/26**	**0.052713**	**0.193531**	**0.115380**	**0.055955**	**0.036465**	**0.014634**	**0.020138**	**0.0**				
36	Mebo frass Run 2 Brome 7/18-	0.017822	0.082620	0.112102	0.071902	0.039908	0.024361	0.029167	0.0				
37	Mebo frass Run 2 Brome 7/18-	0.027884	0.095826	0.086322	0.044031	0.050002	0.011080	0.029279	0.0				
38	Mebo frass Run 2 Brome 7/18-	0.035452	0.125128	0.092414	0.047351	0.043226	0.012924	0.012425	0.0				
39	**Mebo frass Run 2 Brome 7/18-**	**0.027052**	**0.101191**	**0.096946**	**0.054428**	**0.044379**	**0.016122**	**0.023624**	**0.0**				
40	Mebo frass Run 2 Crepis 7/18-	0.032164	0.156388	0.070297	0.048499	0.039589	0.009032	0.028215	0.0				
41	Mebo frass Run 2 Crepis 7/18-	0.022936	0.153139	0.078992	0.046844	0.050070	0.028356	0.012309	0.0				
42	Mebo frass Run 2 Crepis 7/18-	0.031340	0.161858	0.091402	0.058211	0.054596	0.022125	0.032035	0.0				
43	**Mebo frass Run 2 Crepis 7/18-**	**0.028813**	**0.157128**	**0.080230**	**0.051185**	**0.048085**	**0.019838**	**0.024187**	**0.0**				
44	Mebo frass Run 2 Dandelion 7/18-	0.017573	0.061471	0.142079	0.111889	0.054791	0.035716	0.028574	0.0				
45	Mebo frass Run 2 Dandelion 7/18-	0.020690	0.159492	0.090359	0.070210	0.041087	0.027204	0.027992	0.0				
46	**Mebo frass Run 2 Dandelion 7/18-**	**0.01913**	**0.11048**	**0.11622**	**0.09105**	**0.04794**	**0.03146**	**0.02828**	**0.**				
47	Chcu frass Brome 7/12-7/18	0.043432	0.144770	0.092225	0.063379	0.039101	0.016253	0.013069	0.0				
48	Chcu frass Brome 7/12-7/18	0.041389	0.136011	0.088028	0.060811	0.046290	0.025027	0.026615	0.0				
49	Chcu frass Brome 7/12-7/18	0.040902	0.177715	0.086546	0.068967	0.057643	0.034951	0.037111	0.0				

Chcu Frass Calibrations / Schenk_s Shultz_s Green_s Frass / Meridic Diet Cadavers / Soil Controls / **Sheet1** / She

Sheet 7 / 9 | PageStyle_Sheet1 | 100% | STD

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

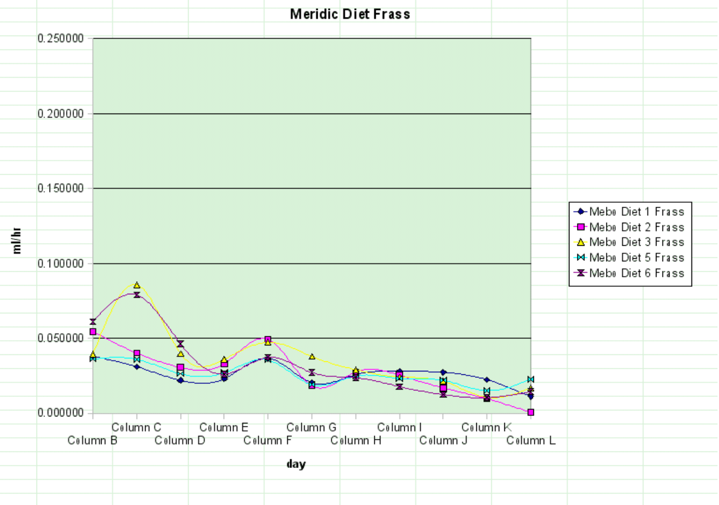
do the same for the next 6 bold rows

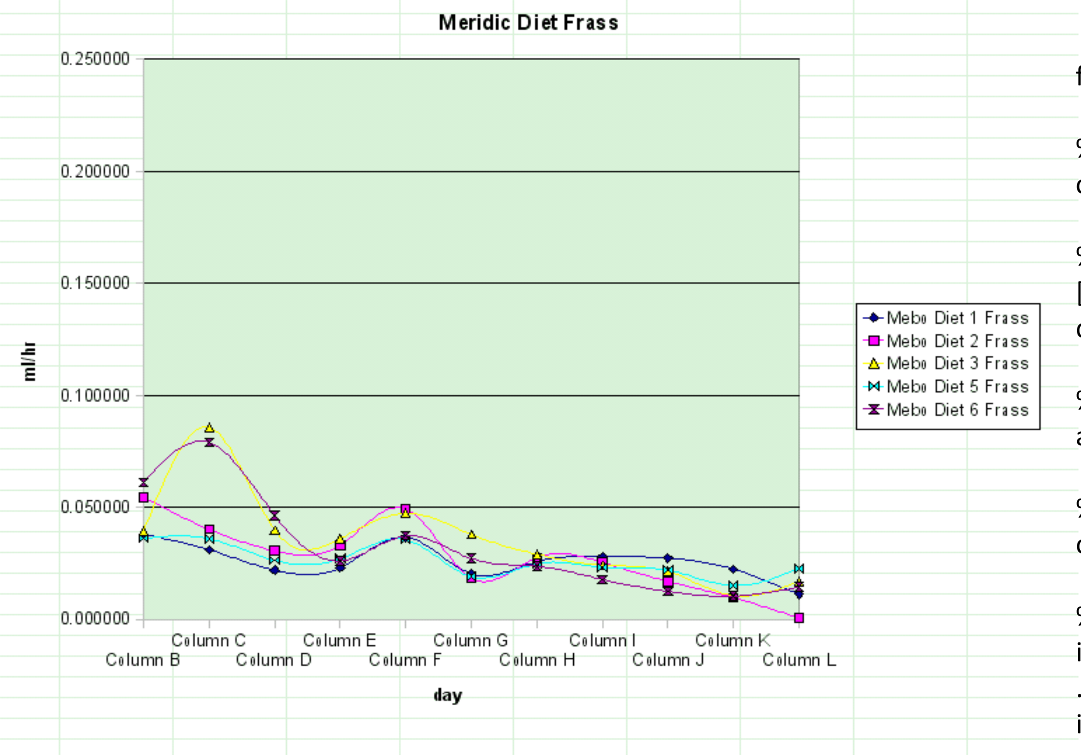
then the next 5

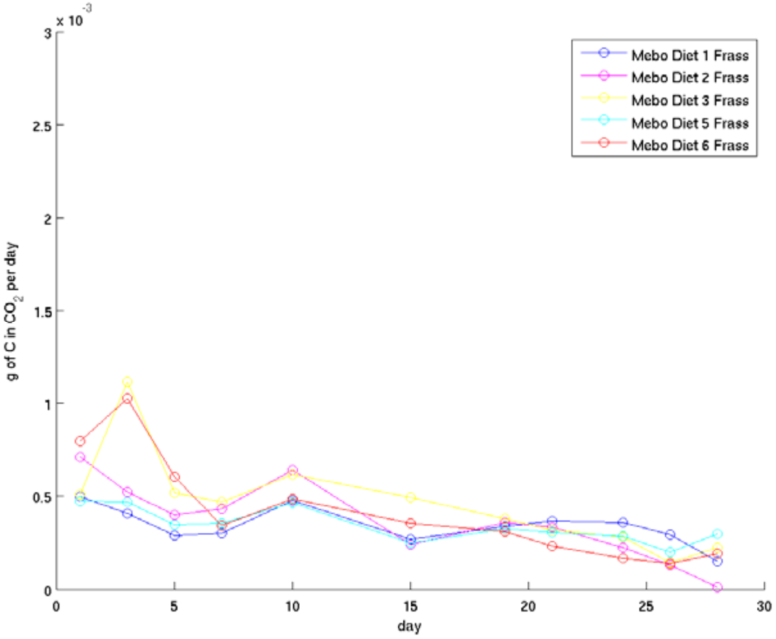
then the next 5

then the next 7

then the next 4







```
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 rows defined in an element of i{ }
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
```

Exercises (optional)

Write scripts (or functions) to do the following:

Exercise 1:

- **(Download the image file http://www.avo.alaska.edu/images/logos/logo_avo_transparent_new.jpg).**
- Load the image into MATLAB (into an array A) with **imread**
- Plot it with **imagesc**
- Find the **size** of the array
- Find the **min**imum and **max**imum values
- Add a **colorbar**
- Add a **title, xlabel, ylabel**
- Move the figure on the screen with **set**(gcf, ‘Position’, …)
- Move the axes with **set**(gca, ‘position’, …)

Exercise 2:

- Store rows 5, 10 and 20 of the array A in new vectors
- In a new **figure, plot** (in 2D) each of those 3 vectors in a different **subplot**
- In a new **figure, plot** (in 2D) each of those 3 vectors on same axes using **hold on**
- **set** the range of data shown (zoom in)
- **set** tick position
- Add **xlabel, ylabel, title** and **legend**.
- **print** to an EPS file

Exercise 3:

- Load an Excel worksheet containing data (**xlsread**)
- **plot** some of the data in MATLAB.
- **print** to a PNG file.
- View the PNG file in your web browser.
- Modify the data in MATLAB.
- Write to new data back to a worksheet in Excel (**xlswrite**)

Send your scripts to gthompson@alaska.edu if you want feedback.