Matlab Introduction

Dr. Antonio A. Trani
Professor
Dept. of Civil and Environmental Engineering

Purpose of this Section

- To illustrate simple uses of the MATLAB^{TM} Technical language
- To help you understand under what circumstances is MATLAB a better choice than spreadsheets and high-level languages
- To understand some of the MATLAB toolboxes used in specialized technical computation
- Just for the fun of learning something new (the most important reason)

^{TM} trademark of the Mathworks (Natick, MA)

What is MATLAB?

- A high-performance language for technical computing (Mathworks, 1998)
- Typical uses of MATLAB:
  - Mathematical computations
  - Algorithmic development
  - Model prototyping (prior to complex model development)
  - Data analysis and exploration of data (visualization)
  - Scientific and engineering graphics for presentation
  - Complex analysis using MATLAB toolboxes (i.e., statistics, neural networks, fuzzy logic, H-infinity control, economics, etc.)

Why is MATLAB Good for Me?

- Because it simplifies the analysis of mathematical models
- It frees you from coding in high-level languages (saves a lot of time - with some computational speed penalties)
- Provides an extensible programming/visualization environment
- Provides professional looking graphs
- The learning curve of this language is moderate (my own bias)
- Our students learn the language in EF, Math and Physics. Perhaps we should exploit this fact in our junior and senior courses

Where is MATLAB in the Scheme of Things?

Complimentary tool to spreadsheets and prog. languages

Tool	My Remarks (subjective)
Spreadsheets (Excel)	• Easy to use • Good for general purpose computation • Nice standard graphics • Good connectivity to other applications • Platform independent
Numeric/Symbolic Tools (MATLAB, Mathematica/Mathcad)	• Moderate learning curve • Good for general and scientific computations • Excellent graphics • Good connectivity to other applications • Platform independent
Compiled Languages (C/C++)	• Require a fairly steep learning curve • Best control over the development cycle • Good graphics if a separate library is available • Generally platform dependent

A Few More Facts About MATLAB

- MATLAB was created to be a numerical computation package (based on the LINPACK routines)
- MATLAB is usually faster than Mathematica and Maple in numeric intensive tasks
- MATLAB has more textbooks than other packages combined (850+ books). Perhaps this speaks on the acceptance by the user community
- Go to www.mathworks.com for a complete set of books on various subjects

Tutorial Outline

- Basics of MATLAB (various modes of operation)
- Input-output commands
- Data analysis functions
- Matrices and vector operations
- Script files and programming issues
- Output graphics and plots (bar, 2D and 3D commands, interactive features)
- Numerical solutions to differential equations (queueing and dynamic system applications)
- Simulink and other MATLAB toolboxes (C compiler, Neural Networks, Statistics, etc.)

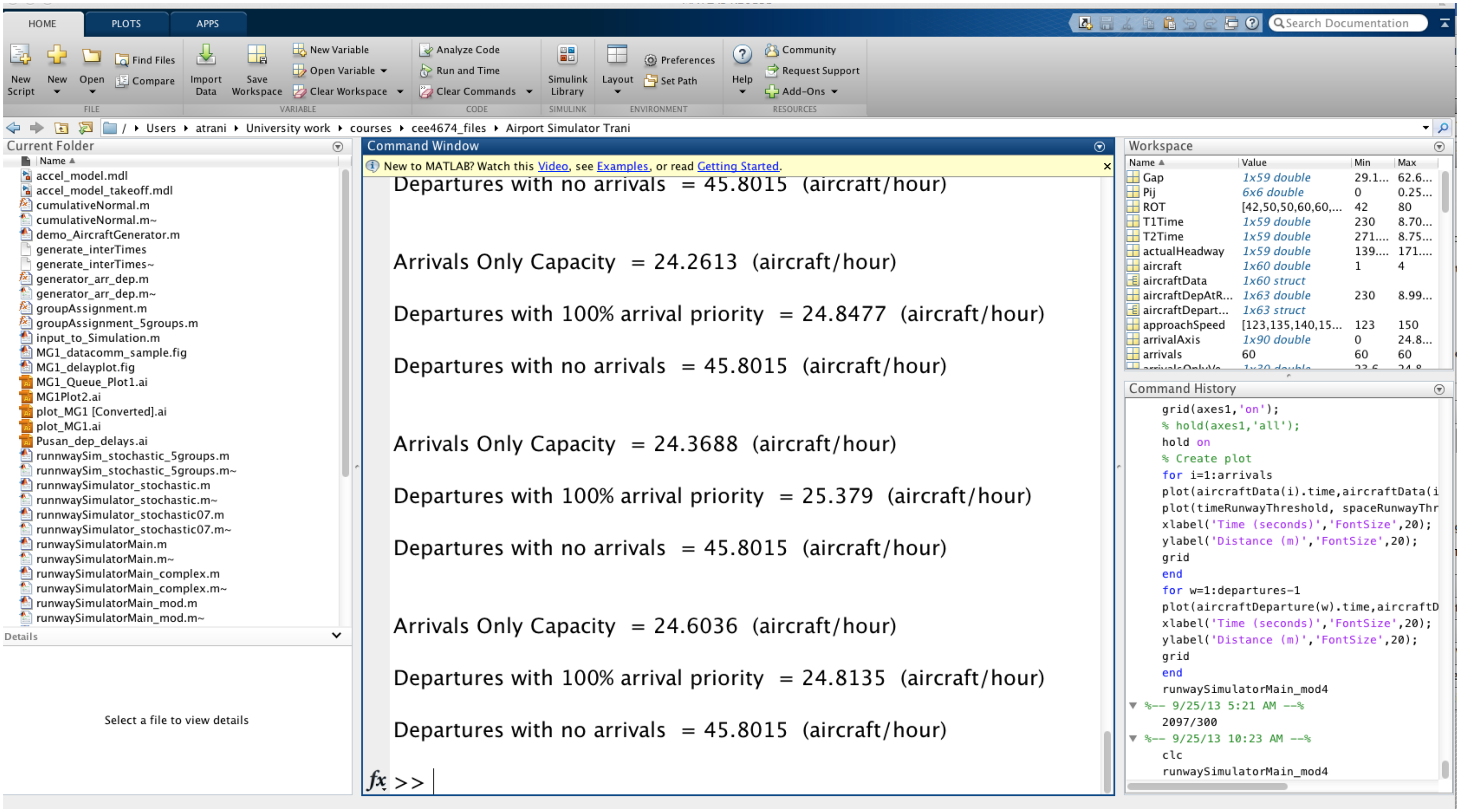
Basics of the Technical Language

- MATLAB is a technical language to ease scientific computations
- The name is derived from MATrix LABoratory
- It provides many of the attributes of spreadsheets and programming languages
- MATLAB is a case sensitive language (a variable named “c” is different than another one called “C”)
- MATLAB can be used in interactive mode or in full compiled version (platform specific mode)
- In interactive mode MATLAB scripts are platform independent (good for cross platform portability)

MATLAB Foundations

- MATLAB works with matrices
- Everything MATLAB understands is a matrix (from text to large cell arrays and structure arrays)
- Various data types exist within MATLAB
  - single precision
  - double precision
  - integer (8 bit)
- Performance of MATLAB scripts can be improved using vector operations (more on this later)
- MATLAB has advanced data structures including object-oriented programming functionality and overloadable operators

The MATLAB Environment



/ ▸ Users ▸ atrani ▸ University work ▸ courses ▸ cee4674_files ▸ Airport Simulator Trani

Current Folder

Name ▲

- accel_model.mdl
- accel_model_takeoff.mdl
- cumulativeNormal.m
- cumulativeNormal.m~
- demo_AircraftGenerator.m
- generate_interTimes
- generate_interTimes~
- generator_arr_dep.m
- generator_arr_dep.m~
- groupAssignment.m
- groupAssignment_5groups.m
- input_to_Simulation.m
- MG1_datacomm_sample.fig
- MG1_delayplot.fig
- MG1_Queue_Plot1.ai
- MG1Plot2.ai
- plot_MG1 [Converted].ai
- plot_MG1.ai
- Pusan_dep_delays.ai
- runnwaySim_stochastic_5groups.m
- runnwaySim_stochastic_5groups.m~
- runnwaySimulator_stochastic.m
- runnwaySimulator_stochastic.m~
- runnwaySimulator_stochastic07.m
- runnwaySimulator_stochastic07.m~
- runwaySimulatorMain.m
- runwaySimulatorMain.m~
- runwaySimulatorMain_complex.m
- runwaySimulatorMain_complex.m~
- runwaySimulatorMain_mod.m
- runwaySimulatorMain_mod.m~

Details

Select a file to view details

Command Window

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

```
Departures with no arrivals  = 45.8015  (aircraft/hour)


Arrivals Only Capacity  = 24.2613  (aircraft/hour)

Departures with 100% arrival priority  = 24.8477  (aircraft/hour)

Departures with no arrivals  = 45.8015  (aircraft/hour)


Arrivals Only Capacity  = 24.3688  (aircraft/hour)

Departures with 100% arrival priority  = 25.379  (aircraft/hour)

Departures with no arrivals  = 45.8015  (aircraft/hour)


Arrivals Only Capacity  = 24.6036  (aircraft/hour)

Departures with 100% arrival priority  = 24.8135  (aircraft/hour)

Departures with no arrivals  = 45.8015  (aircraft/hour)

fx >> 
```

Workspace

Name ▲	Value	Min	Max
Gap	1x59 double	29.1...	62.6...
Pij	6x6 double	0	0.25...
ROT	[42,50,50,60,60,...	42	80
T1Time	1x59 double	230	8.70...
T2Time	1x59 double	271....	8.75...
actualHeadway	1x59 double	139....	171....
aircraft	1x60 double	1	4
aircraftData	1x60 struct		
aircraftDepAtR...	1x63 double	230	8.99...
aircraftDepart...	1x63 struct		
approachSpeed	[123,135,140,15...	123	150
arrivalAxis	1x90 double	0	24.8...
arrivals	60	60	60

Command History

```
    grid(axes1,'on');
    % hold(axes1,'all');
    hold on
    % Create plot
    for i=1:arrivals
    plot(aircraftData(i).time,aircraftData(i
    plot(timeRunwayThreshold, spaceRunwayThr
    xlabel('Time (seconds)','FontSize',20);
    ylabel('Distance (m)','FontSize',20);
    grid
    end
    for w=1:departures-1
    plot(aircraftDeparture(w).time,aircraftD
    xlabel('Time (seconds)','FontSize',20);
    ylabel('Distance (m)','FontSize',20);
    grid
    end
    runwaySimulatorMain_mod4
▼ %-- 9/25/13 5:21 AM --%
    2097/300
▼ %-- 9/25/13 10:23 AM --%
    clc
    runwaySimulatorMain_mod4
```

Basic Components of the MATLAB Environment

MATLAB has the following basic window components:

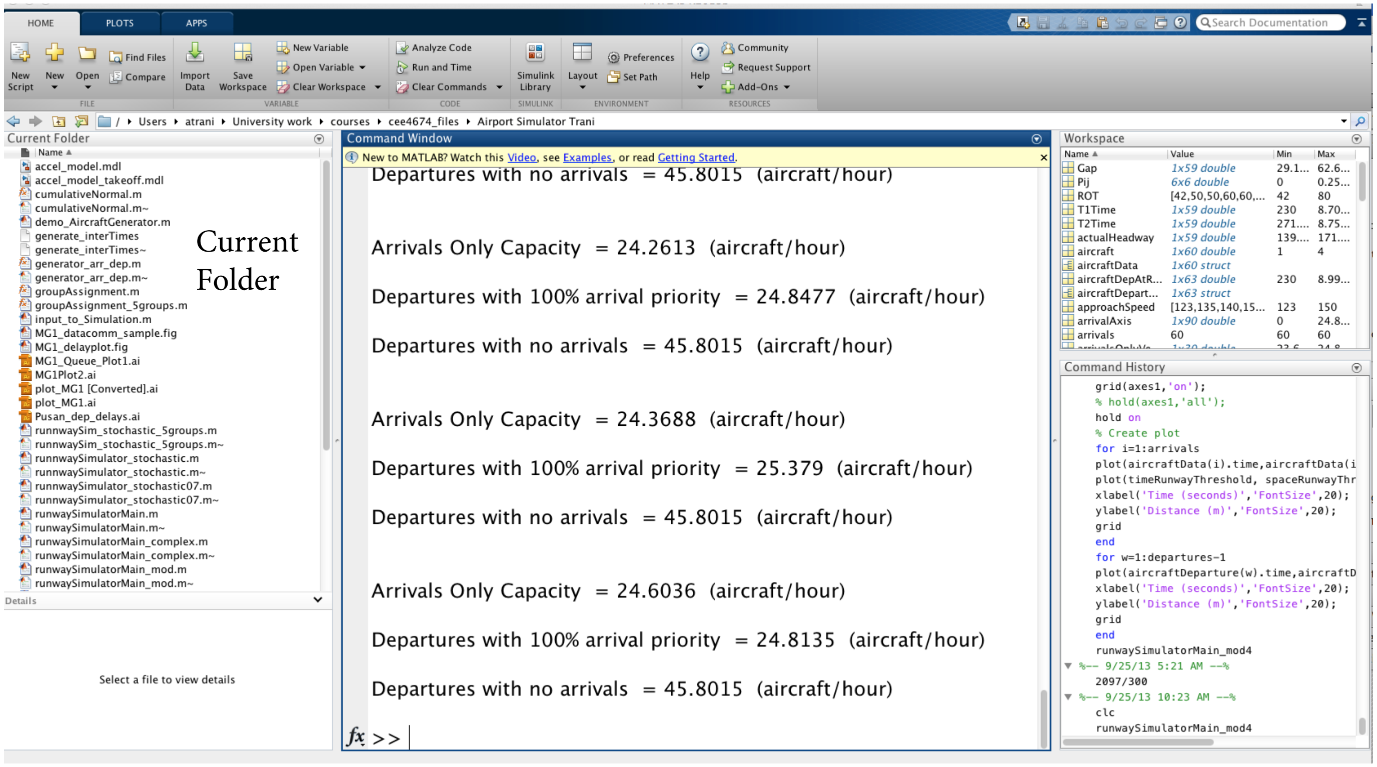
- Launch Pad Window
- to access all MATLAB services and toolboxes
- Command Window
  - to execute commands in the MATLAB environment
- Current Directory Window
  - to quickly access files on the MATLAB path
- Figure Window
  - to display graphical output from MATLAB code

Basic Components of the MATLAB Environment

- Workspace Window
  - to view variable definitions and variable memory allocations
- M-File Editor/Debugger Window
  - to write M-files (includes color-coded syntax features)
  - to debug M-files interactively (break points)
- MATLAB Path Window
  - to add and delete folders to the MATLAB path
- Command History Window
  - displays all commands issued in MATLAB since the last session (good for learning and verification)

Composite MATLAB Window Environment

- A Java-based GUI environment allows you to easily navigate between various windows



/ ▸ Users ▸ atrani ▸ University work ▸ courses ▸ cee4674_files ▸ Airport Simulator Trani

Current Folder

Name ▲

```
accel_model.mdl
accel_model_takeoff.mdl
cumulativeNormal.m
cumulativeNormal.m~
demo_AircraftGenerator.m
generate_interTimes
generate_interTimes~
generator_arr_dep.m
generator_arr_dep.m~
groupAssignment.m
groupAssignment_5groups.m
input_to_Simulation.m
MG1_datacomm_sample.fig
MG1_delayplot.fig
MG1_Queue_Plot1.ai
MG1Plot2.ai
plot_MG1 [Converted].ai
plot_MG1.ai
Pusan_dep_delays.ai
runnwaySim_stochastic_5groups.m
runnwaySim_stochastic_5groups.m~
runnwaySimulator_stochastic.m
runnwaySimulator_stochastic.m~
runnwaySimulator_stochastic07.m
runnwaySimulator_stochastic07.m~
runwaySimulatorMain.m
runwaySimulatorMain.m~
runwaySimulatorMain_complex.m
runwaySimulatorMain_complex.m~
runwaySimulatorMain_mod.m
runwaySimulatorMain_mod.m~
```

Details

Select a file to view details

Command Window

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

```
Departures with no arrivals  = 45.8015  (aircraft/hour)


Arrivals Only Capacity  = 24.2613  (aircraft/hour)

Departures with 100% arrival priority  = 24.8477  (aircraft/hour)

Departures with no arrivals  = 45.8015  (aircraft/hour)


Arrivals Only Capacity  = 24.3688  (aircraft/hour)

Departures with 100% arrival priority  = 25.379  (aircraft/hour)

Departures with no arrivals  = 45.8015  (aircraft/hour)


Arrivals Only Capacity  = 24.6036  (aircraft/hour)

Departures with 100% arrival priority  = 24.8135  (aircraft/hour)

Departures with no arrivals  = 45.8015  (aircraft/hour)

fx >>
```

Workspace

Name ▲	Value	Min	Max
Gap	1x59 double	29.1...	62.6...
Pij	6x6 double	0	0.25...
ROT	[42,50,50,60,60,...	42	80
T1Time	1x59 double	230	8.70...
T2Time	1x59 double	271....	8.75...
actualHeadway	1x59 double	139....	171....
aircraft	1x60 double	1	4
aircraftData	1x60 struct		
aircraftDepAtR...	1x63 double	230	8.99...
aircraftDepart...	1x63 struct		
approachSpeed	[123,135,140,15...	123	150
arrivalAxis	1x90 double	0	24.8...
arrivals	60	60	60

Command History

```
    grid(axes1,'on');
    % hold(axes1,'all');
    hold on
    % Create plot
    for i=1:arrivals
    plot(aircraftData(i).time,aircraftData(i
    plot(timeRunwayThreshold, spaceRunwayThr
    xlabel('Time (seconds)','FontSize',20);
    ylabel('Distance (m)','FontSize',20);
    grid
    end
    for w=1:departures-1
    plot(aircraftDeparture(w).time,aircraftD
    xlabel('Time (seconds)','FontSize',20);
    ylabel('Distance (m)','FontSize',20);
    grid
    end
    runwaySimulatorMain_mod4
▼ %-- 9/25/13 5:21 AM --%
    2097/300
▼ %-- 9/25/13 10:23 AM --%
    clc
    runwaySimulatorMain_mod4
```

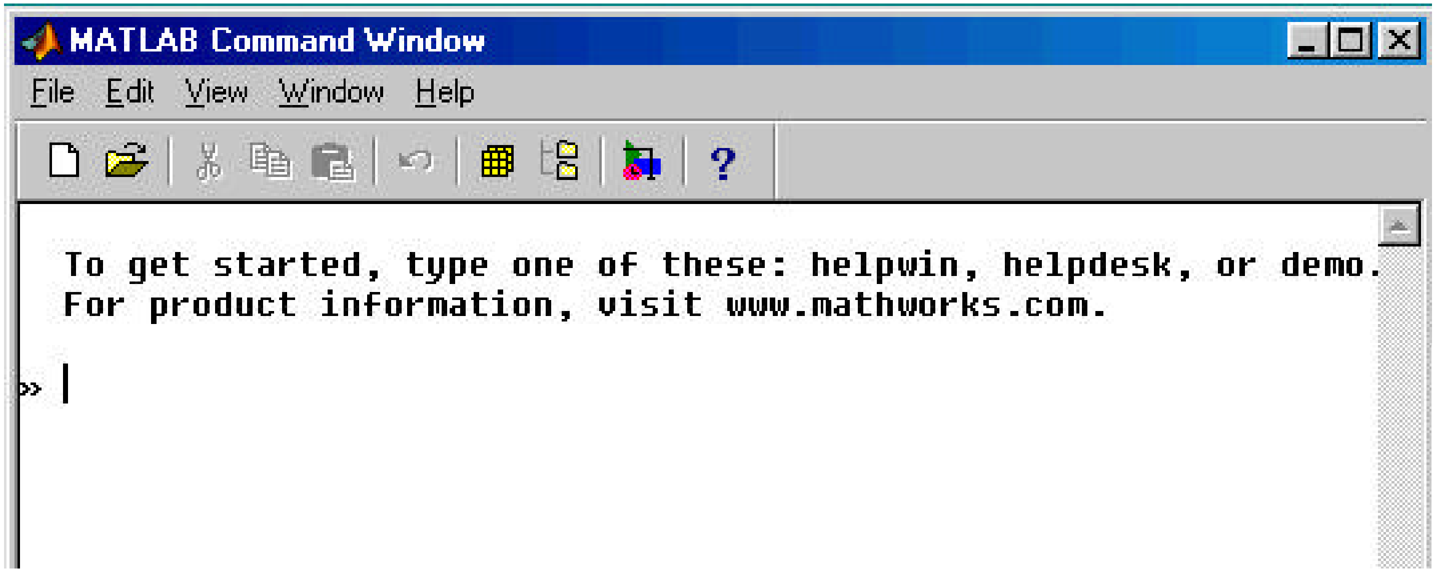
Current Folder

Workspace Window

Command History

MATLAB Command Window

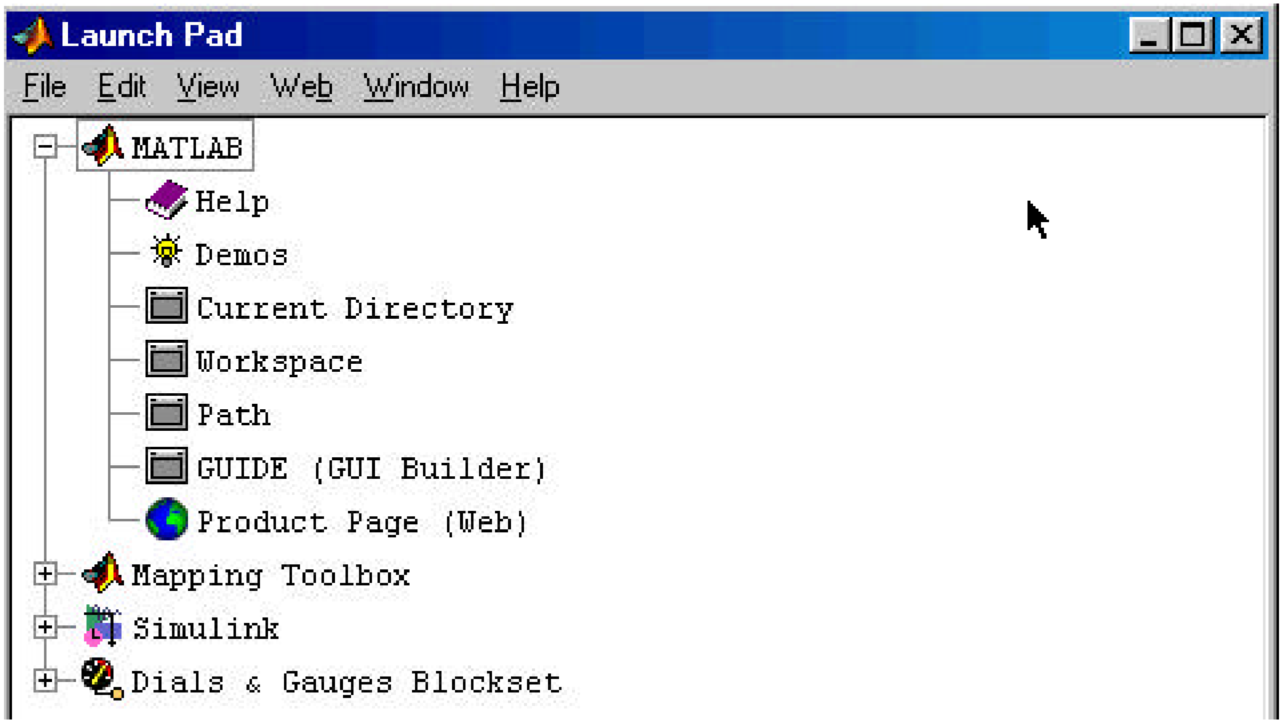
- The command window allows you to interact with MATLAB just as if you type things in a calculator
- Cut and paste operations ease the repetition of tasks
- Use ‘up-arrow’ key to repeat commands (command history)



```
  To get started, type one of these: helpwin, helpdesk, or demo.
  For product information, visit www.mathworks.com.

>> 
```

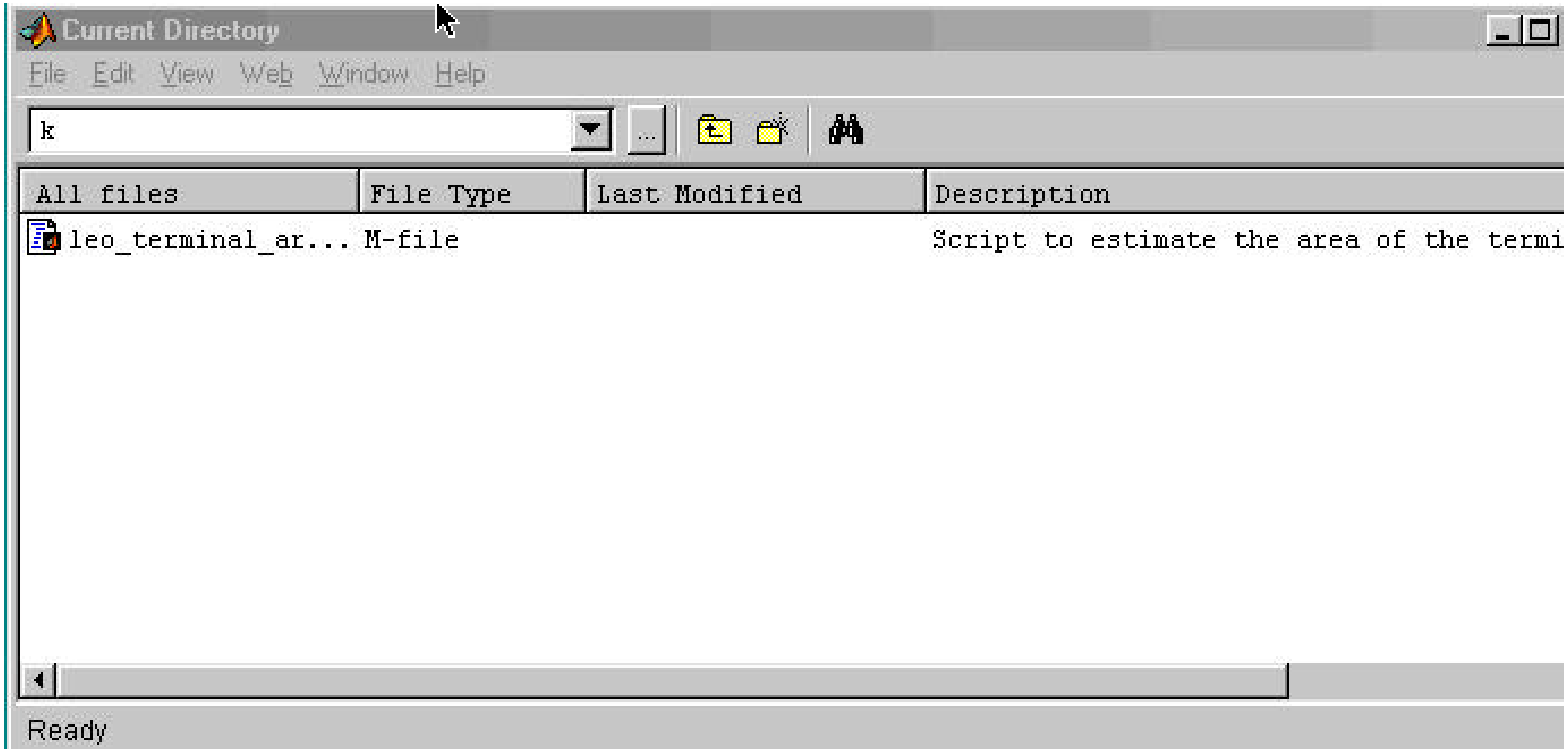
MATLAB Launch Pad Window

- The launch window allows you to quickly select among various MATLAB components and toolboxes
- Shown below are MATLAB and three installed toolboxes in the launch window environment



MATLAB Current Directory Window

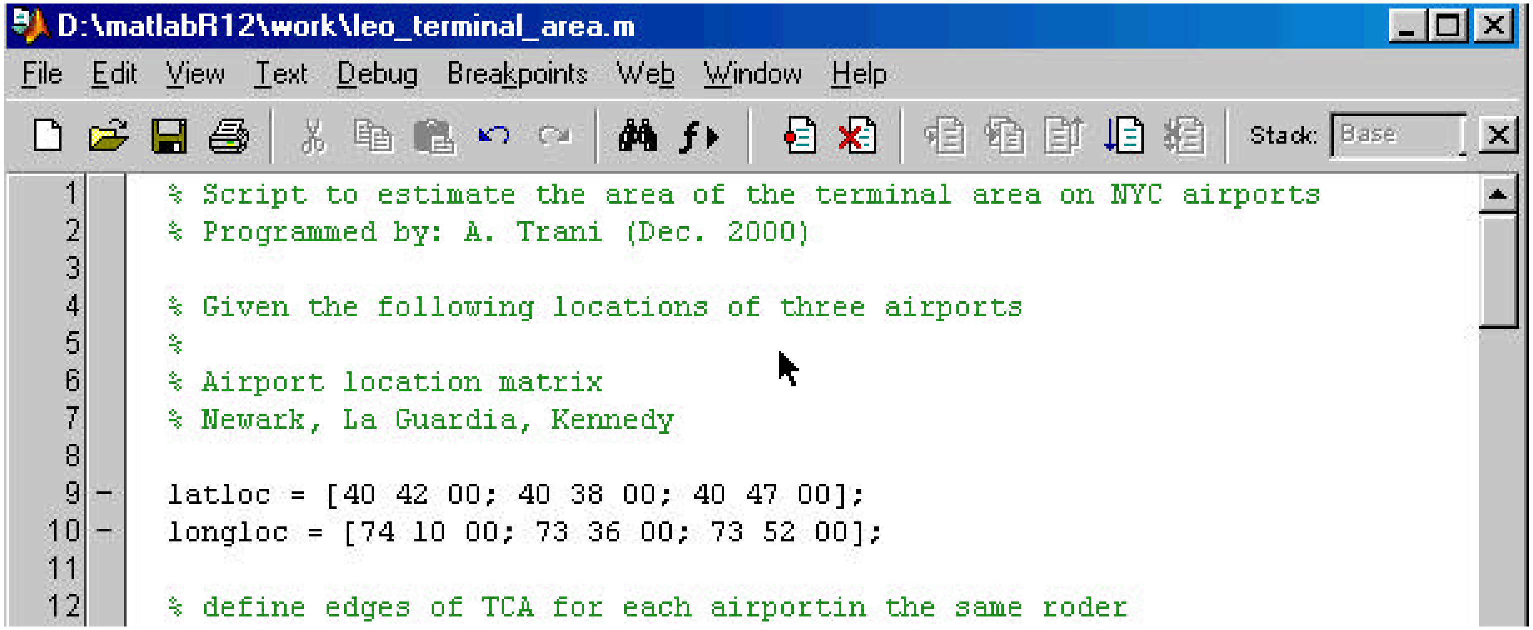
- Provides quick access to all files available in your Path
- Provides a brief description (when files are commented out) of each M-file



All files	File Type	Last Modified	Description
leo_terminal_ar...	M-file		Script to estimate the area of the termi

MATLAB Editor/Debuger Window

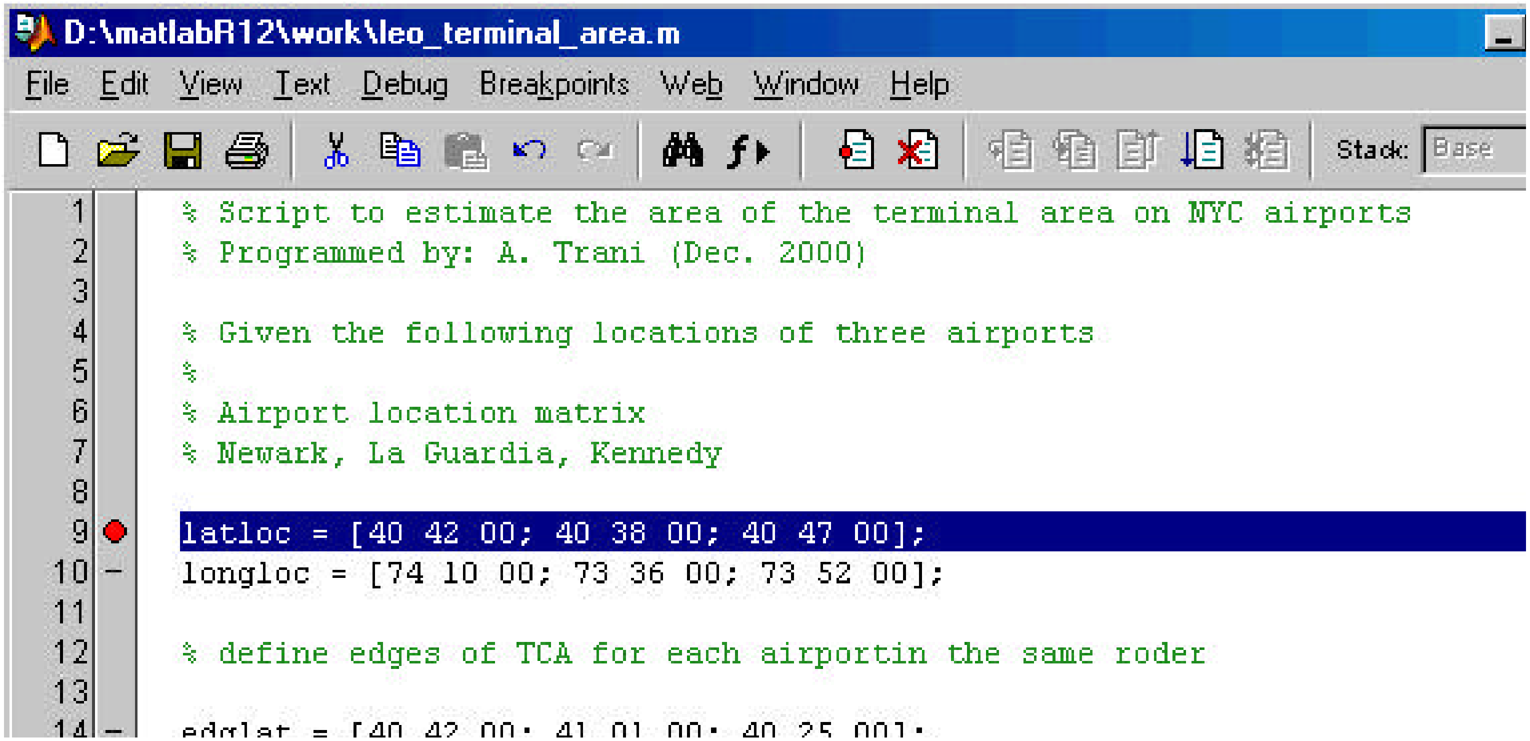
- Provides the same functionality found in most programming language development environments
  - Color codes MATLAB built-in functions (blue color)
  - Easy access to cut, paste, print, and debug operations
  - Checks balance in MATLAB function syntax



```
% Script to estimate the area of the terminal area on NYC airports
% Programmed by: A. Trani (Dec. 2000)

% Given the following locations of three airports
%
% Airport location matrix
% Newark, La Guardia, Kennedy

latloc = [40 42 00; 40 38 00; 40 47 00];
longloc = [74 10 00; 73 36 00; 73 52 00];

% define edges of TCA for each airportin the same roder
```

MATLAB Editor/Debugger

MATLAB has an interactive debugger to help you step through your source code. This debugger has many of the same functional features found in high-level programming languages (i.e., FORTRAN, C/C++, etc.).



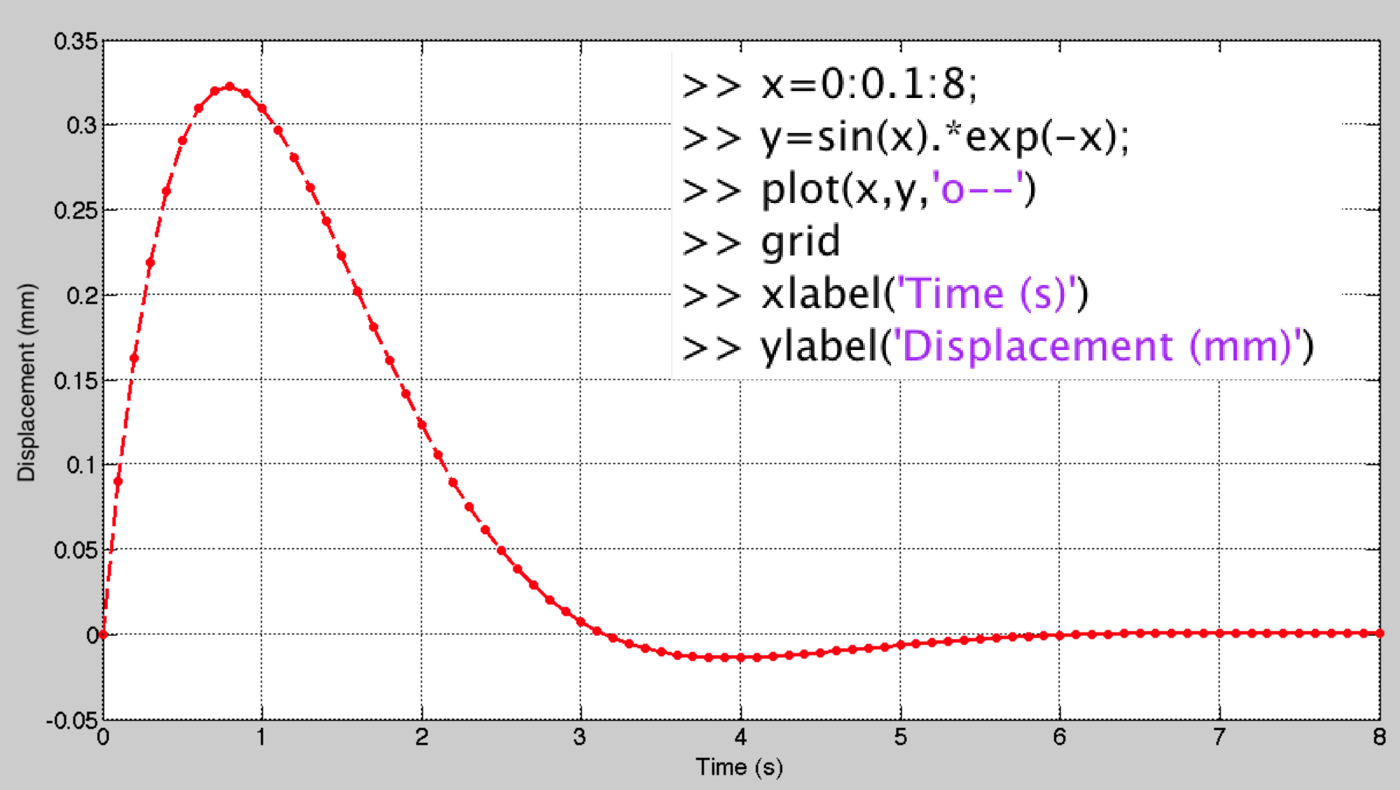
```
% Script to estimate the area of the terminal area on NYC airports
% Programmed by: A. Trani (Dec. 2000)

% Given the following locations of three airports
%
% Airport location matrix
% Newark, La Guardia, Kennedy

latloc = [40 42 00; 40 38 00; 40 47 00];
longloc = [74 10 00; 73 36 00; 73 52 00];

% define edges of TCA for each airportin the same roder

edglat = [40 42 00; 41 01 00; 40 25 00];
```

MATLAB Debugger

- Allows standard programming techniques such:
  - Breakpoints
  - Break on error, warnings and overflows
  - Step in and out of script
  - Function dependencies

MATLAB Figure Window

- Displays the graphic contents of MATLAB code (either from Command Window, an M-file, or output from MEX file)



```
>> x=0:0.1:8;
>> y=sin(x).*exp(-x);
>> plot(x,y,'o--')
>> grid
>> xlabel('Time (s)')
>> ylabel('Displacement (mm)')
```

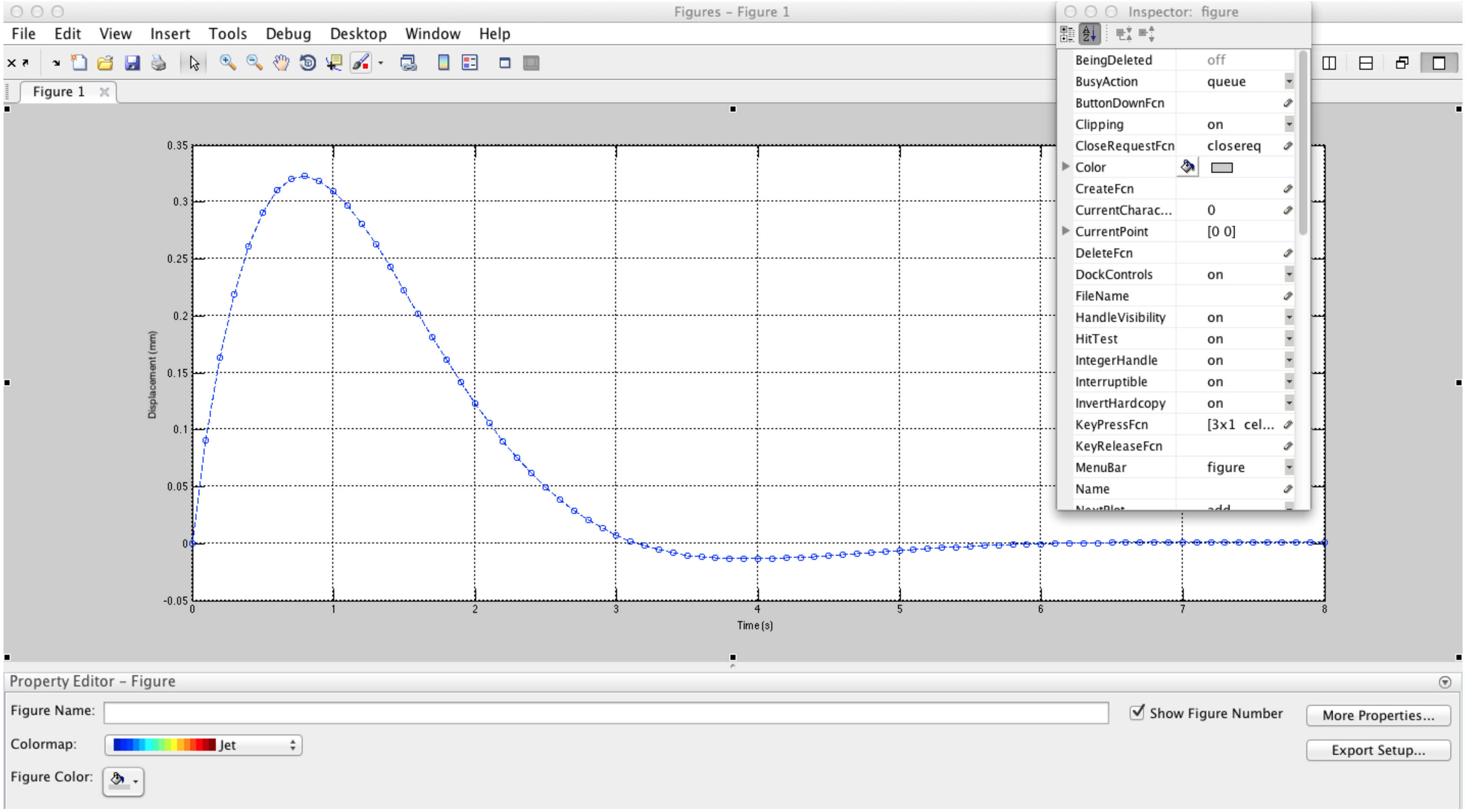
MATLAB Figure Window (cont.)

Figure properties can be changed interactively using the following commands:

- PlotEdit
  - allows interactive changes to plots (add legend, lines, arrows, etc.)
  - This function is automatically invoked in MATLAB 5.3
- PropEdit
  - Allows changes to all Handle Graphic properties in a MATLAB plot
  - Requires knowledge of Handle Graphics (more on this later)

MATLAB Figure Property Editor

- Propedit : Allows you to change properties of a plot



BeingDeleted	off
BusyAction	queue
ButtonDownFcn	
Clipping	on
CloseRequestFcn	closereq
Color	
CreateFcn	
CurrentCharac...	0
CurrentPoint	[0 0]
DeleteFcn	
DockControls	on
FileName	
HandleVisibility	on
HitTest	on
IntegerHandle	on
Interruptible	on
InvertHardcopy	on
KeyPressFcn	[3x1 cel...
KeyReleaseFcn	
MenuBar	figure
Name	
NextPlot	add

Property Editor – Figure

Figure Name:

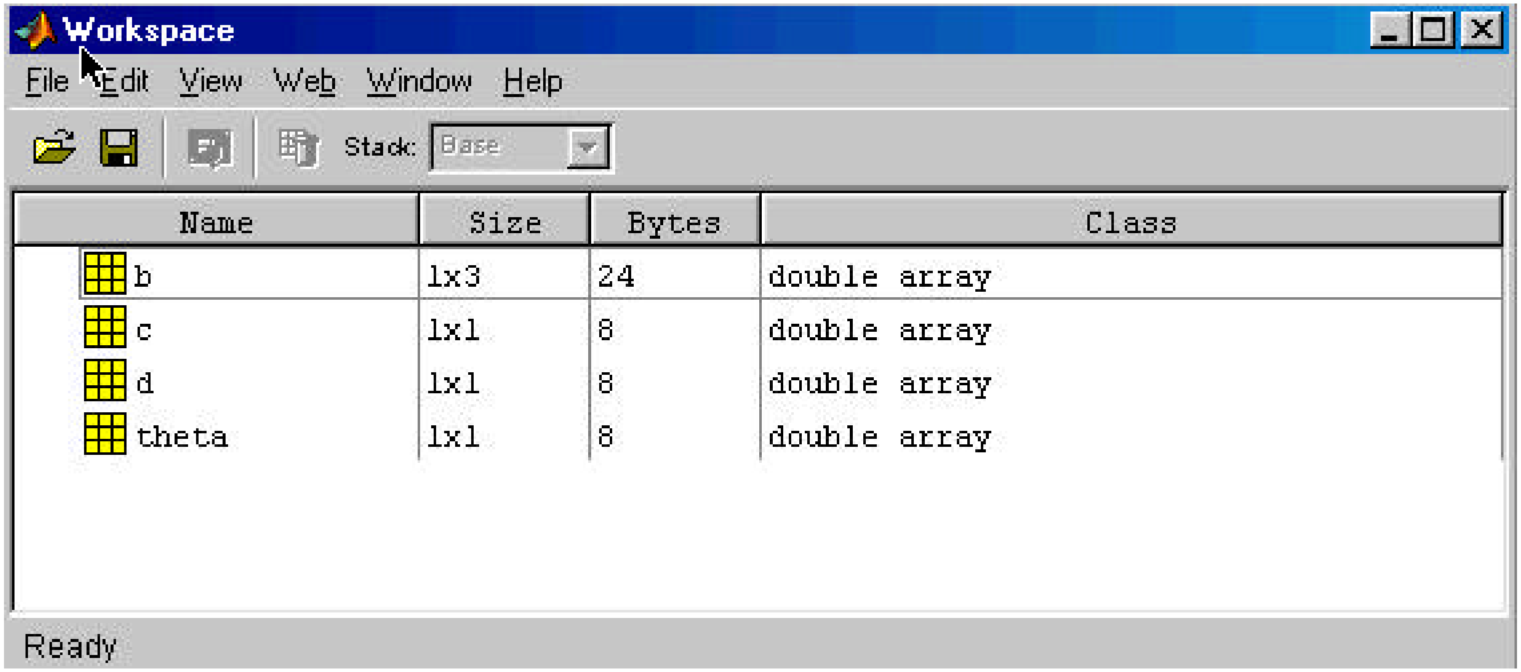
Colormap: Jet

Figure Color:

☑ Show Figure Number

MATLAB Workspace

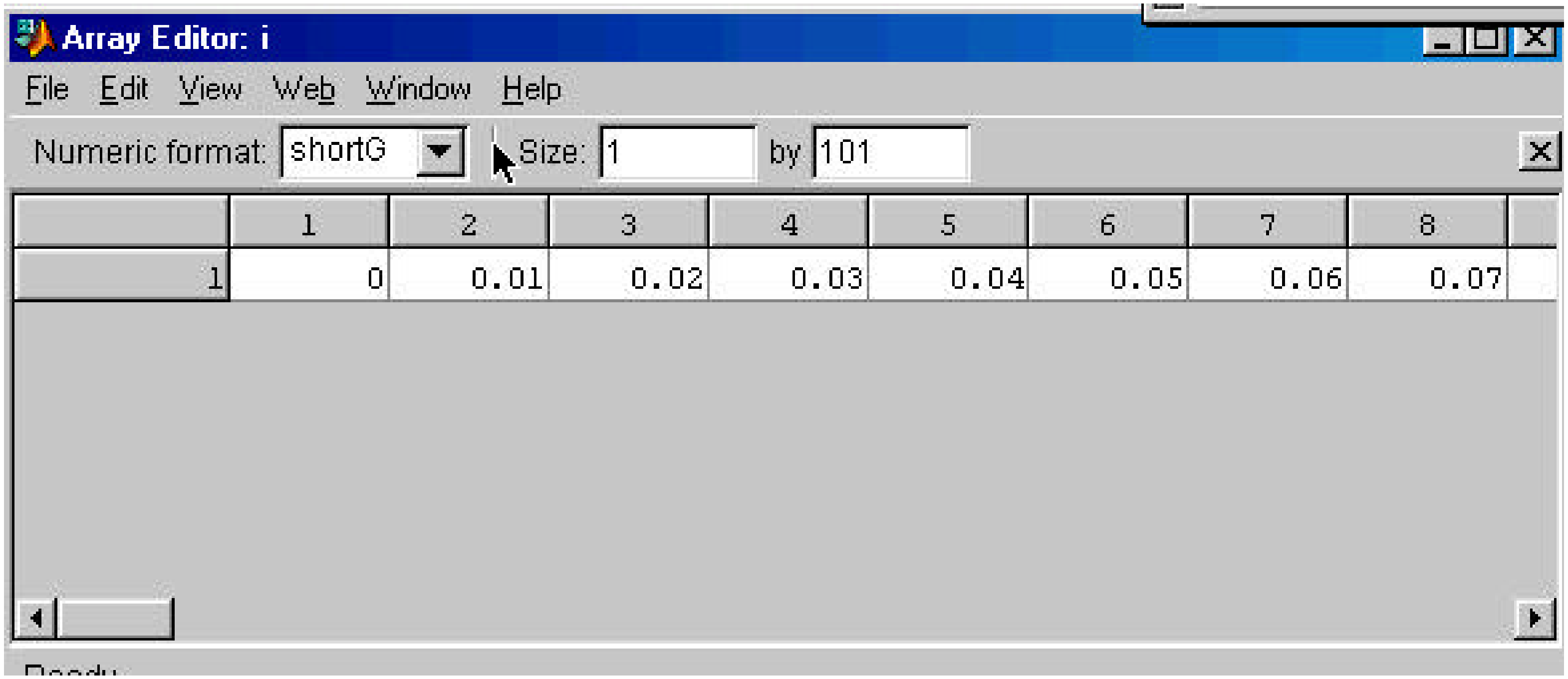
As you develop and execute models in MATLAB the workspace stores all variables names and definitions for you. All variables are usually available to you unless the workspace is clear with the ‘>>clear’ command.



Name	Size	Bytes	Class
b	1x3	24	double array
c	1x1	8	double array
d	1x1	8	double array
theta	1x1	8	double array

Array Editor of Workspace Variables

- The workspace window allows you to inspect (and modify) variables in a spreadsheet-type window
- Cut and paste operations from the clipboard are also permitted from other applications

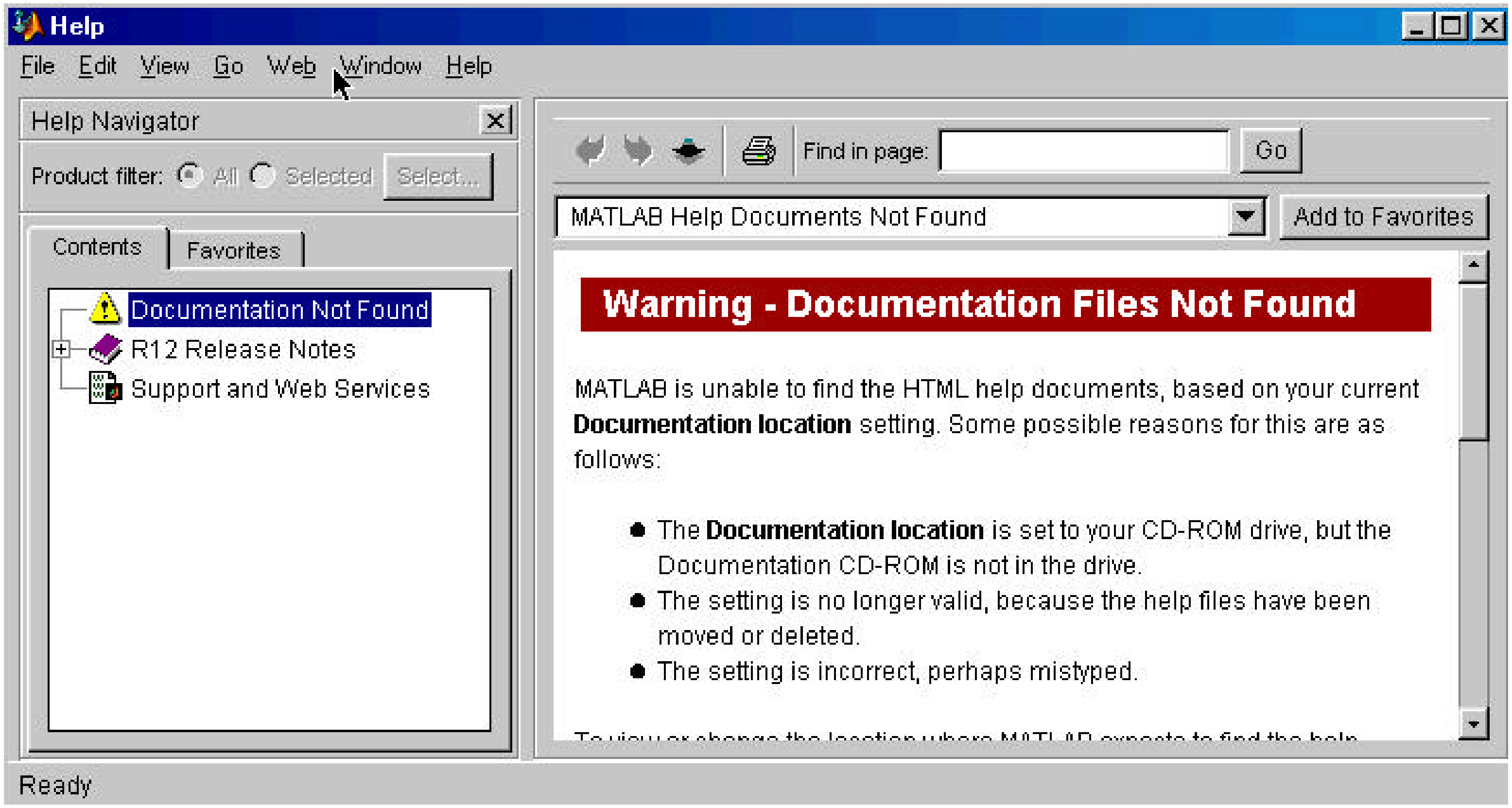


Numeric format: shortG Size: 1 by 101

	1	2	3	4	5	6	7	8
1	0	0.01	0.02	0.03	0.04	0.05	0.06	0.07

Matlab Help Window

- Provides acces to various help files (both internal and on-line files available on the web)



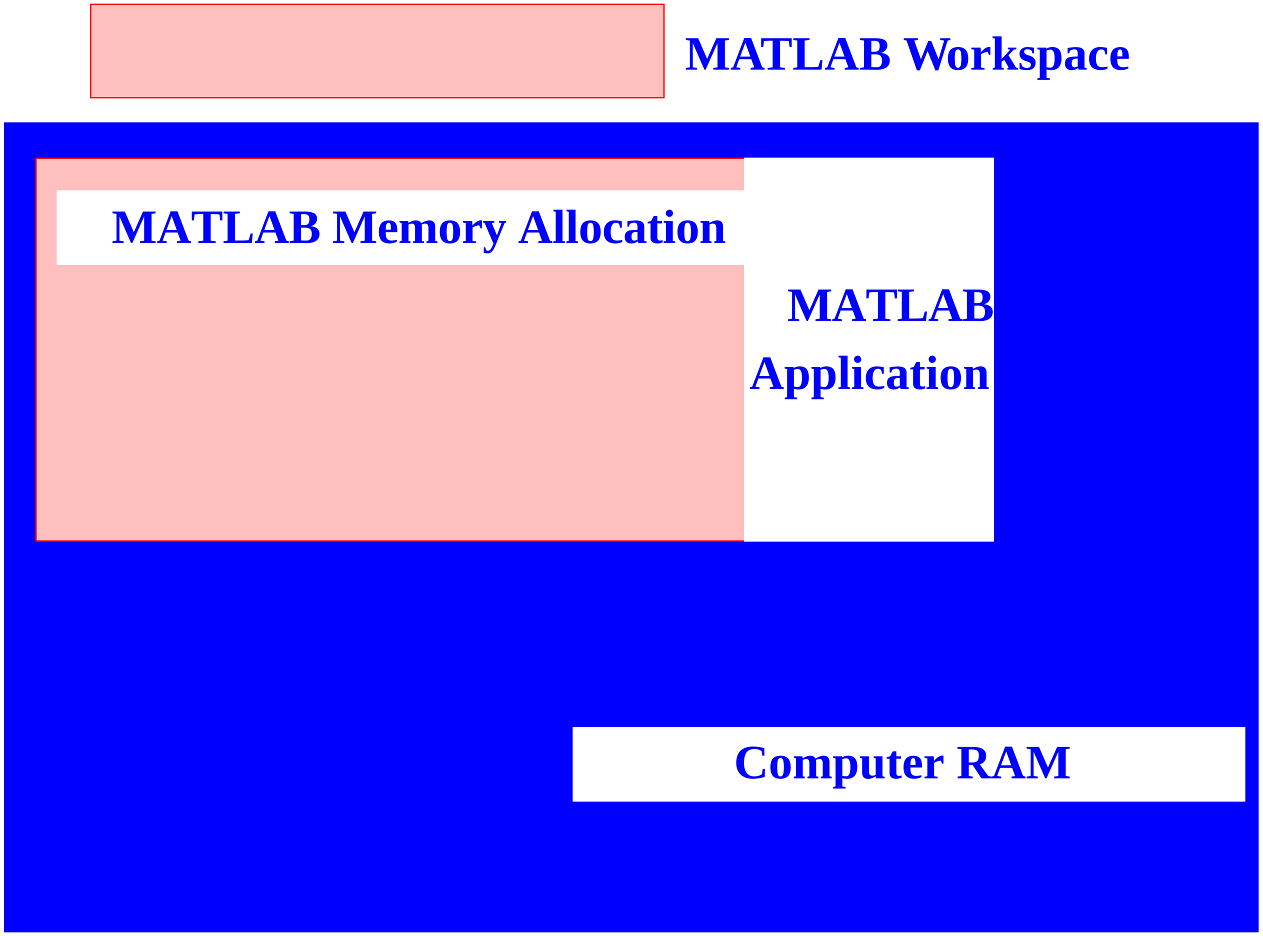
Warning - Documentation Files Not Found

MATLAB is unable to find the HTML help documents, based on your current **Documentation location** setting. Some possible reasons for this are as follows:

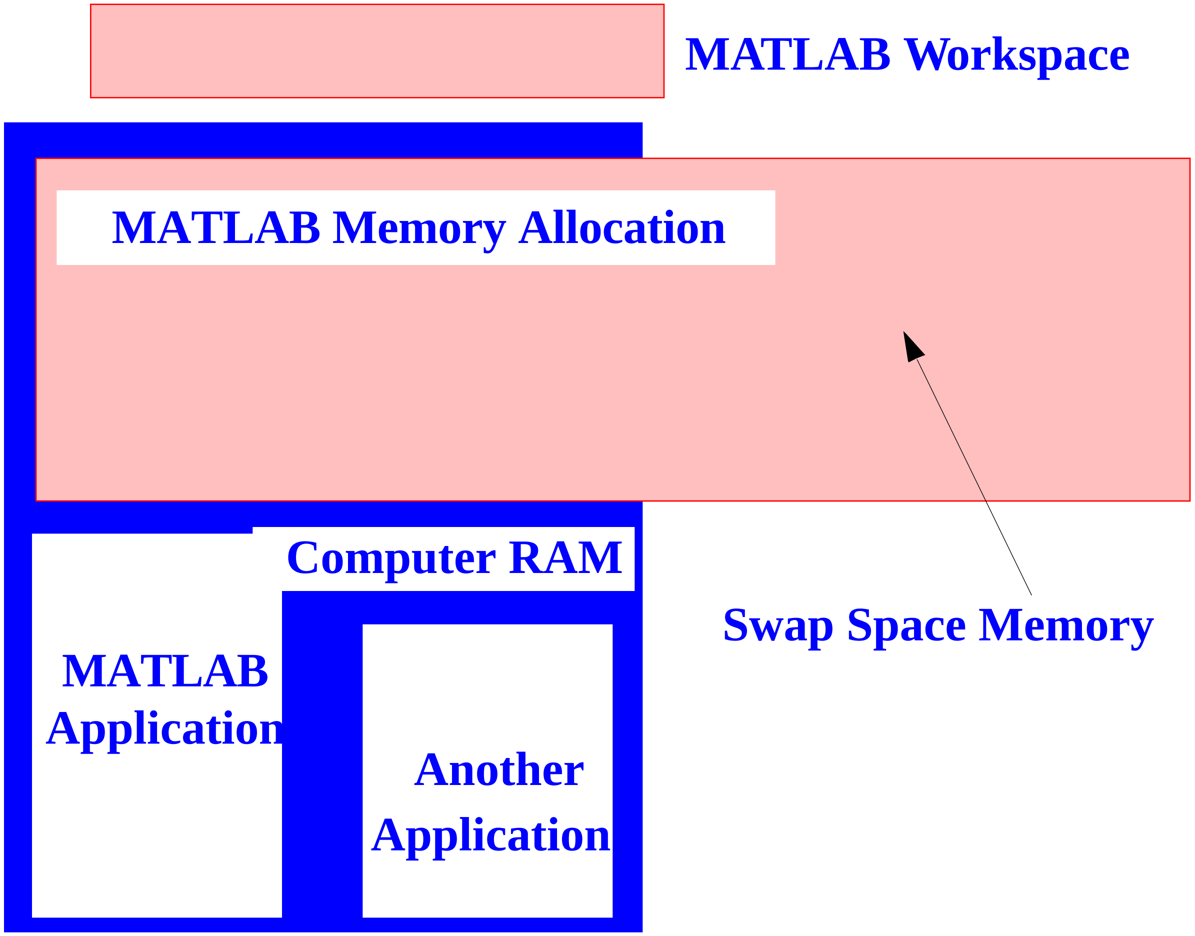
- The **Documentation location** is set to your CD-ROM drive, but the Documentation CD-ROM is not in the drive.
- The setting is no longer valid, because the help files have been moved or deleted.
- The setting is incorrect, perhaps mistyped.

To view or change the location where MATLAB expects to find the help

MATLAB Workspace (Macintosh Model)

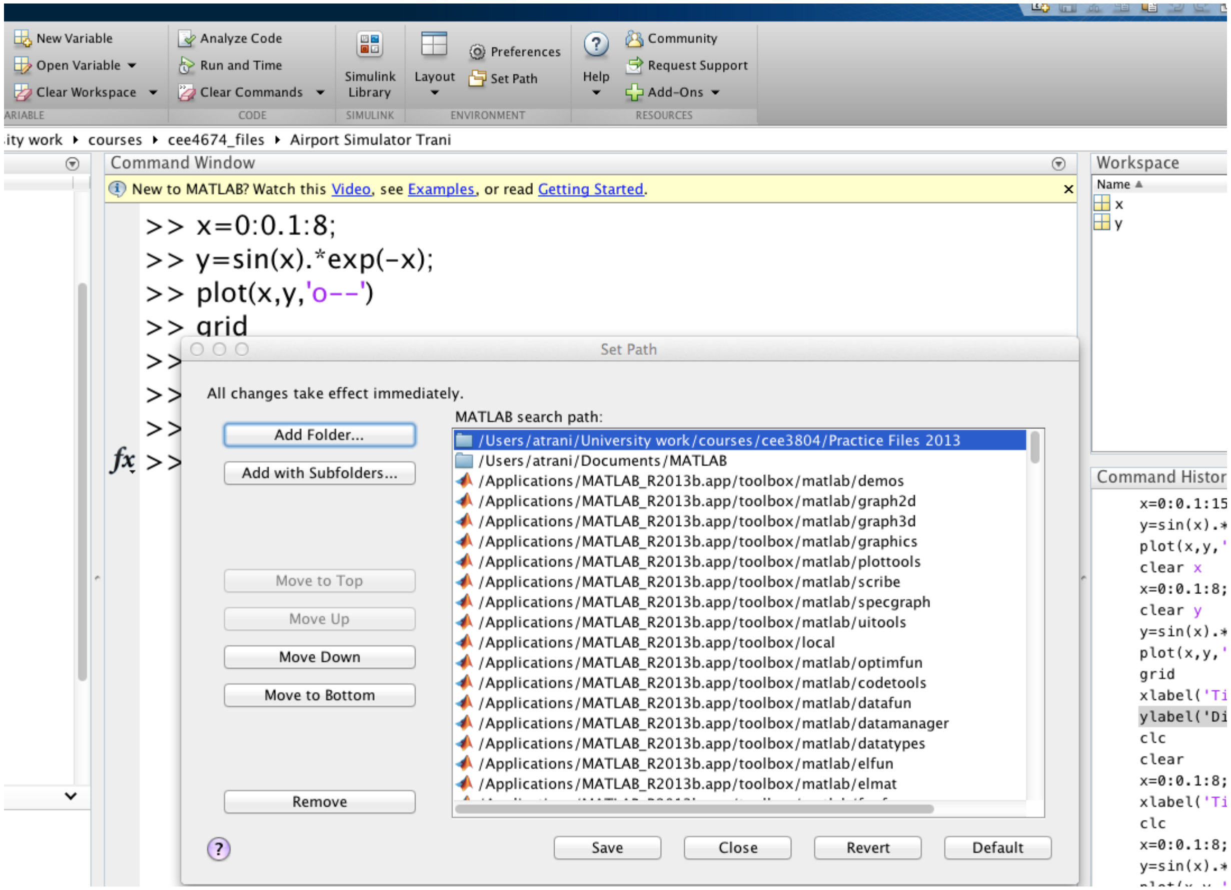


MATLAB Workspace (Windows/UNIX Models)



MATLAB Path Window

- Shows all folders contained in the MATLAB path
- Allows you to include other folders from within MATLAB can be executed



```
>> x=0:0.1:8;
>> y=sin(x).*exp(-x);
>> plot(x,y,'o--')
>> grid
>>
>>
>>
fx >>
```

All changes take effect immediately.

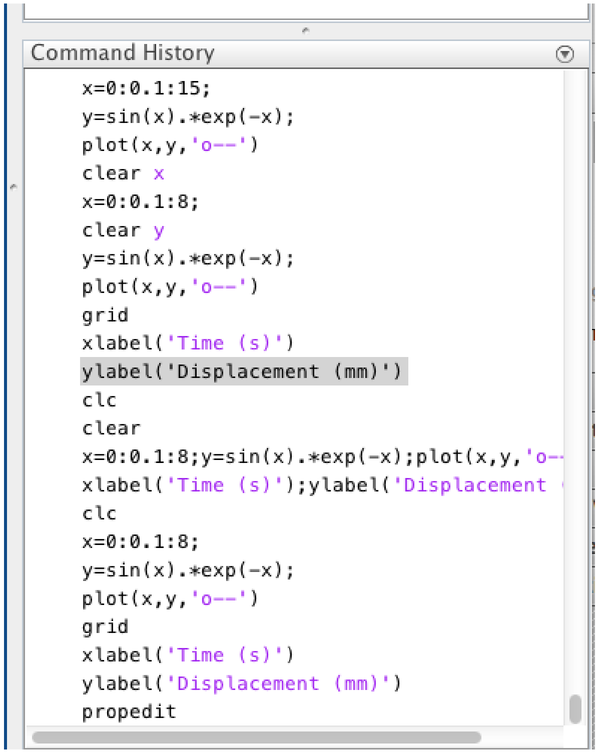
MATLAB search path:

```
/Users/atrani/University work/courses/cee3804/Practice Files 2013
/Users/atrani/Documents/MATLAB
/Applications/MATLAB_R2013b.app/toolbox/matlab/demos
/Applications/MATLAB_R2013b.app/toolbox/matlab/graph2d
/Applications/MATLAB_R2013b.app/toolbox/matlab/graph3d
/Applications/MATLAB_R2013b.app/toolbox/matlab/graphics
/Applications/MATLAB_R2013b.app/toolbox/matlab/plottools
/Applications/MATLAB_R2013b.app/toolbox/matlab/scribe
/Applications/MATLAB_R2013b.app/toolbox/matlab/specgraph
/Applications/MATLAB_R2013b.app/toolbox/matlab/uitools
/Applications/MATLAB_R2013b.app/toolbox/local
/Applications/MATLAB_R2013b.app/toolbox/matlab/optimfun
/Applications/MATLAB_R2013b.app/toolbox/matlab/codetools
/Applications/MATLAB_R2013b.app/toolbox/matlab/datafun
/Applications/MATLAB_R2013b.app/toolbox/matlab/datamanager
/Applications/MATLAB_R2013b.app/toolbox/matlab/datatypes
/Applications/MATLAB_R2013b.app/toolbox/matlab/elfun
/Applications/MATLAB_R2013b.app/toolbox/matlab/elmat
```

```
x=0:0.1:15
y=sin(x).*
plot(x,y,'
clear x
x=0:0.1:8;
clear y
y=sin(x).*
plot(x,y,'
grid
xlabel('Ti
ylabel('Di
clc
clear
x=0:0.1:8;
xlabel('Ti
clc
x=0:0.1:8;
y=sin(x).*
```

MATLAB Command History Window

- Displays all previous commands issued in a MATLAB session
- Good to verify computation sequences and for learning



```
x=0:0.1:15;
y=sin(x).*exp(-x);
plot(x,y,'o--')
clear x
x=0:0.1:8;
clear y
y=sin(x).*exp(-x);
plot(x,y,'o--')
grid
xlabel('Time (s)')
ylabel('Displacement (mm)')
clc
clear
x=0:0.1:8;y=sin(x).*exp(-x);plot(x,y,'o-
xlabel('Time (s)');ylabel('Displacement
clc
x=0:0.1:8;
y=sin(x).*exp(-x);
plot(x,y,'o--')
grid
xlabel('Time (s)')
ylabel('Displacement (mm)')
propedit
```

Interacting with MATLAB

There are several options to interact with MATLAB

Mode	Remarks
Command line	• Interactive mode • Good for quick computations or changes
M-files (script files)	• Semi-interactive mode • Good to prototype small to complex models • Used most of the time • Platform independent
Executable MEX files	• Require a C/C++ compiler • Fastest to execute • Platform specific (target specific)

Interactive Mode (I)

- Use the MATLAB Command Window to interact with MATLAB in “calculator” mode

```
>> a=[3 2 4; 4 5 6; 1 2 3]
```

Try this out

- Multiple commands can be executed using the semi-colon “;” separator between commands

```
>> a=[3 2 4; 4 5 6; 1 2 3] ; b=[3 2 5]’ ; c=a*b
```

This single line defines two matrices (a and b) and computes their product (c)

Interactive Mode (II)

- Use the semi-colon “;” separator to tell the MATLAB to inhibit output to the Command Window

```
>> a=[3 2 4; 4 5 6; 1 2 3]

>> a=[3 2 4; 4 5 6; 1 2 3];
```

Try this and see the difference

- Note that the semi-colon is also used to differentiate between rows in a matrix definition
- All commands that can be executed within the MATLAB Command Window

General Purpose Commands

helpwin	help window with hypertext navigation
demo	runs MATLAB demos from a MATLAB created Graphic User Interface (GUI)
helpdesk	troubleshooting with hypertext navigation
ver	tells you the version of MATLAB being used
who	lists all variables in the current workspace
whos	lists all variables in the workspace including array sizes
clear	clears all variables and functions from memory

General Purpose Commands (cont.)

pack	consolidates workspace memory
load	load workspace variables from disk (from a previous session)
save	saves all variables and functions in the workspace to disk
quit	quits MATLAB session
what	lists MATLAB files in directory
edit	edits a MATLAB M-file
diary	save text of MATLAB session

Operating System Commands that Work in MATLAB

cd	changes directory
copyfile	copy a file
dir	lists files in current directory
pwd	displays the working directory and its full path
delete	delete a file
mkdir	make a directory
dos	execute DOS command and return result
unix	execute UNIX command and return result

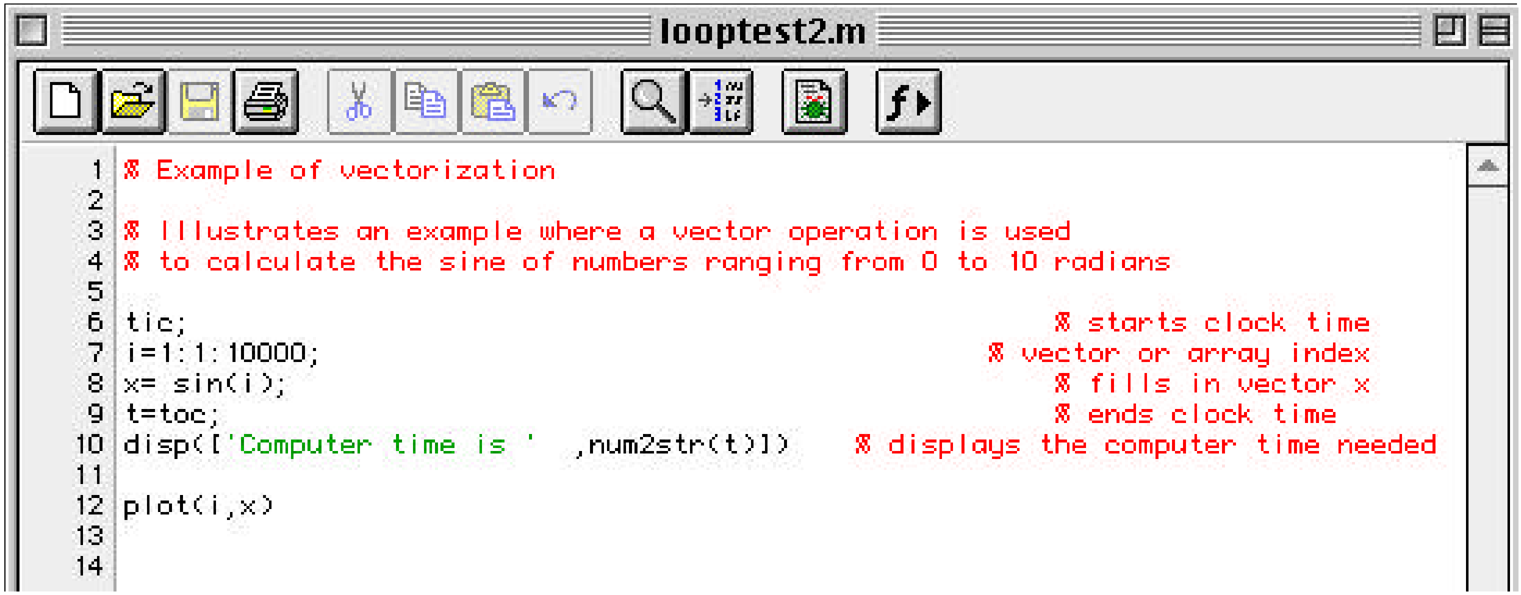
Creating MATLAB Files

Two ways to interact with MATLAB:

- Interactive console mode - allows you to do computations and plots from the command line
- Through M-files - saves your “code” in a text file (with.m termination) allowing you to reuse any function or algorithm in it
- For this tutorial you will be working with M-files most of the time
- Other types of files in MATLAB are MAT (binary) and MEX (executable) files

MATLAB M-Files

- They can be saved, refined and reused as needed
- These files end in “.m” in all platforms
- Use the MATLAB editor to accomplish this task
- Any wordprocessor can also be used (save files as text)



```
% Example of vectorization

% Illustrates an example where a vector operation is used
% to calculate the sine of numbers ranging from 0 to 10 radians

tic;                                                    % starts clock time
i=1:1:10000;                                        % vector or array index
x= sin(i);                                              % fills in vector x
t=toc;                                                  % ends clock time
disp(['Computer time is '  ,num2str(t)])    % displays the computer time needed

plot(i,x)
```

Sample M-File

The following file generates random numbers

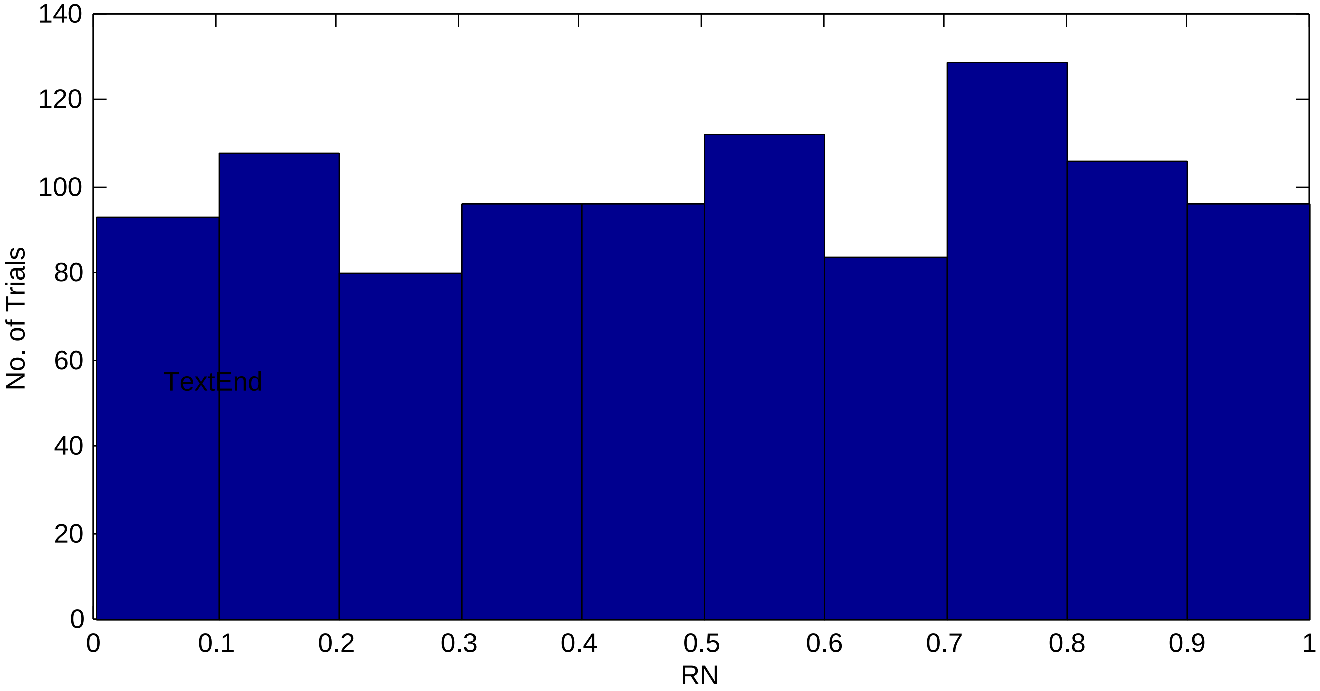
```
% Sample file to generate Random Numbers using
% MATLAB built-in functions

ntrials = 1000;                % No. of trials to be simulated
i=1:1:ntrials;                 % defines a vector with 1k cells
RU(i) = rand(1,ntrials);       % uniform random number
                               % generator
RN(i) = randn (1,ntrials);     % normal random variate
                               % generator
hist(RU)                       % generates a histogram for
                               % variable RU
xlabel('RN')                   % adds the x-label to the plot
ylabel('No. of Trials')        % adds the y-label to the plot
```

Executing the Sample M-File

- Type the previous file using the MATLAB Editor. Name and save the file as randem.m
- To execute the M-file type randem in the Command Window
- Or just go to **Run** from the **Debug** pull-down menu in the Editor/DebugWindow
- Alternatively (in the Mac OS) select the “Save and Execute” under the File menu
- Use the “up-arrow” key to go back to previous commands (cycle back through the MATLAB Command History)

Output of randem.m

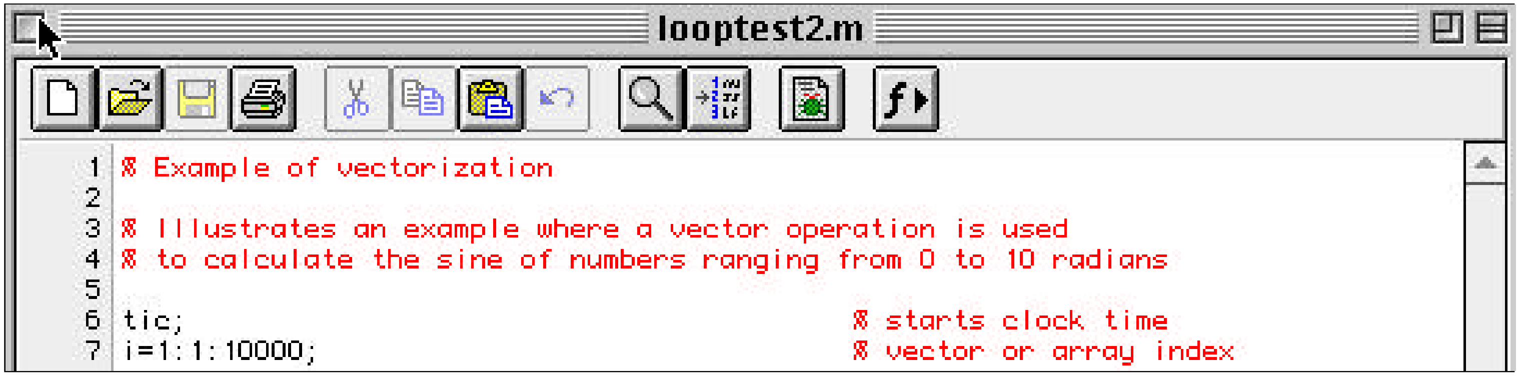
The following figure illustrates the output of randem.m



Adding Comments to Your Code

It is a good practice to add comments to your source code. Use the % operator to introduce comments in MATLAB

- Simplifies our task for code reviewing
- Easy to remember what you did in your code



```
% Example of vectorization

% Illustrates an example where a vector operation is used
% to calculate the sine of numbers ranging from 0 to 10 radians

tic;                                        % starts clock time
i=1:1:10000;                                % vector or array index
```

Few Tasks to Try on Your Own

1) Modify the randem.m M-file and plot a histogram of variable RN

2) Modify randem.m and plot the index variable i versus the values of RN and RU

- Use the plot command as follows:

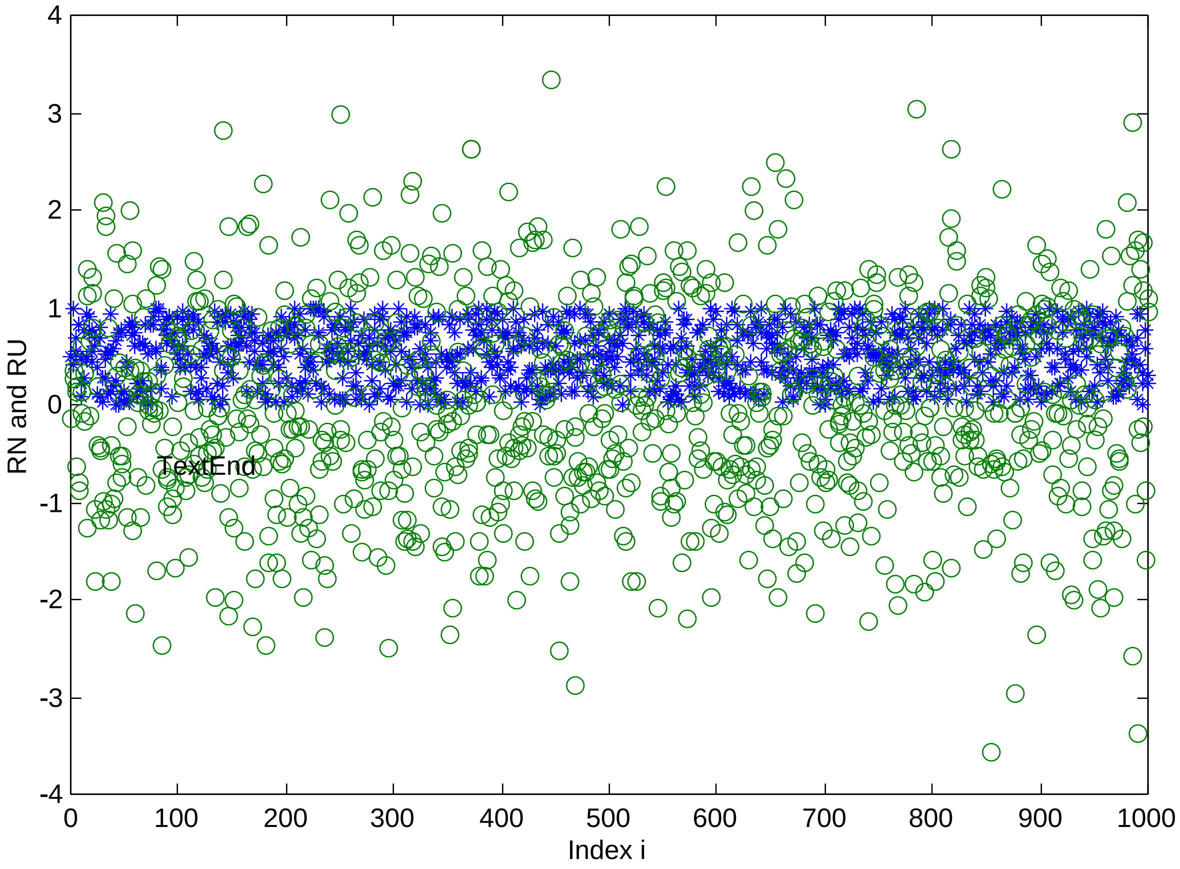
plot(x,y)

- where:
- x is the independent variable (index i in our case)
- y is the dependent variable (values of RU and RN)

3) From the Command Window execute the zoom command and select an area in the plot to view in more detail

Plot of i vs. RN and RU

This plot shows index i versus the values of RU and RN



MATLAB Binary Files

- These files are convenient to store information that needs to be reused
- MATLAB binary files end in .mat
- MATLAB mat files are platform independent
- Use the “save” command at the MATLAB command line.
  - save (saves all workspace variables to matlab.mat)
  - save fname (saves all workspace to fname.mat)
  - save fname x y (saves x and y to fname.mat)
  - save fname x y -ascii (saves x and y in 8-digit text format)
  - save fname x y -ascii -double -tabs (tab delimited format)

Properties of Binary Files

Binary files are compact files only interpreted by MATLAB

- Good to store data to be reused later on
- Easy to transfer among PCs (compact size)
  - This works well across platforms
  - MATLAB 7/8 has good binary files backward compatibility
- Easy to retrieve and work with using the ‘load’command
- Fast retrieval

Loading Binary Files

Binary files can be loaded simply issuing the ‘load’ MATLAB command.

Identified by .mat ending (e.g., traffic.mat)

For example if I want to load a file named traffic.mat (notice the termination) just invoke the load command and do not include the file type termination,

```
>>load traffic
>>who
>> observation density speed volume
>>
```

Note: that in this case the binary file has four variables

Importing Data into MATLAB

There are several ways to enter data in MATLAB:

- Explicitly as elements of a matrix in MATLAB
- Creating data in an M-file
- Loading data from ASCII files
- Use the **Import Wizard** in MATLAB (7.0 version or later)
- Reading data using MATLAB's I/O functions (fopen, fread, etc.)
- Using specialized file reader functions (wk1read, imread, wavread, dlmread)
- Develop an MEX-file to read the data (if FORTRAN or C/C++ routines exist)

Exporting Data from MATLAB

There are several ways to export data from MATLAB:

- Use the diary command (only for small arrays)
- ASCII (use the save command with '-ascii' option)
- Use the function dlmwrite to specify any delimiters needed
- Save data to a file in any specific format (use fopen, fwrite and other MATLAB I/O functions)
- Use specialized MATLAB write functions such as:
  - dlmwrite (user-defined delimeter ascii file)
  - wk1write (spreadsheet format)
  - imwrite and so on

Importing Capabilities (I)

Suppose that we have a data file (called ohare_schedule) containing a typical schedule of daily aircraft operations at Chicago O'Hare Intl Airport. The information provided includes:

1) column 1 = local time (hours)

2) column 2 = number of arrivals per hour

3) column 3 = number of departures per hour

4) column 4 = total operations

This file can be treated as a (24x4) matrix

Sample Data File (ohare_schedule)

The following represents a subset of the ohare_schedule data file

```
0 4 7 11
1 3 2 5
2 2 2 4
3 4 2 6
4 2 8 10
.......
```

Reading the Sample Data File

Method 1 - Use the MATLAB load command

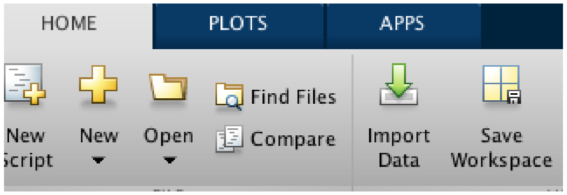
```
>> load ohare_schedule
```

- Loads the data file into the MATLAB Workspace and produces a new array variable called ohare_schedule
- This new array variable has dimensions 24 x 2
- All comment lines (if any) are neglected in the loading process. Only numerical data is read.

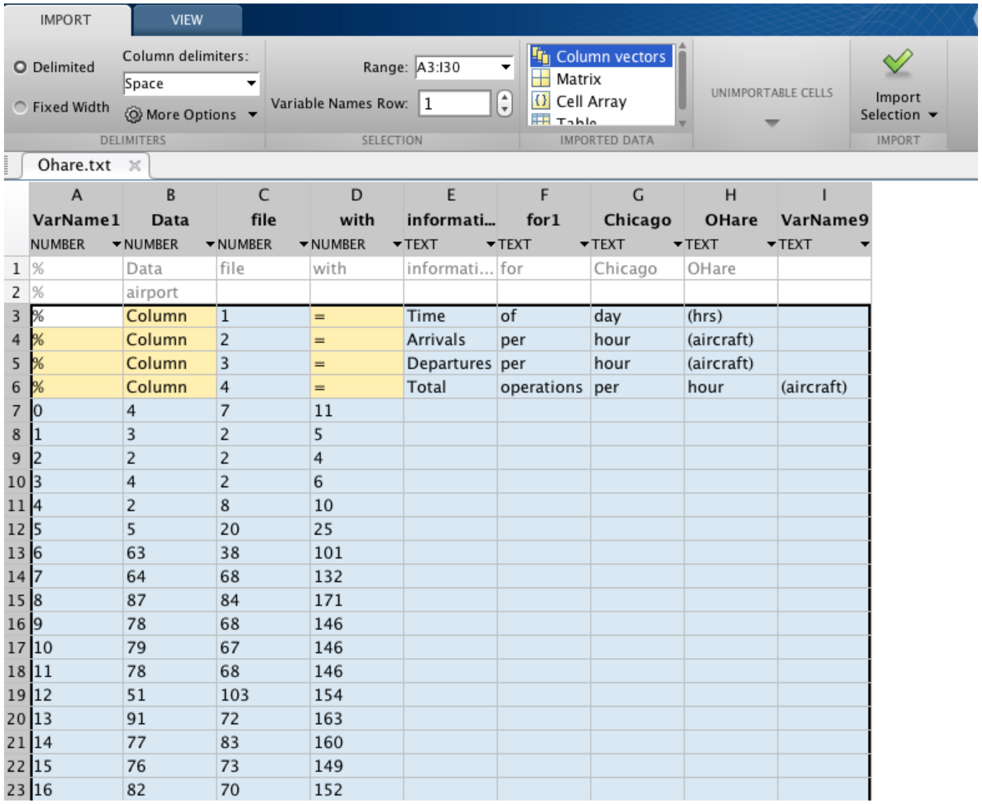
MATLAB Import Screen (version 6.0)

Method 2 - To import data go to the Editor Window

- Select Import from the File pull-down menu



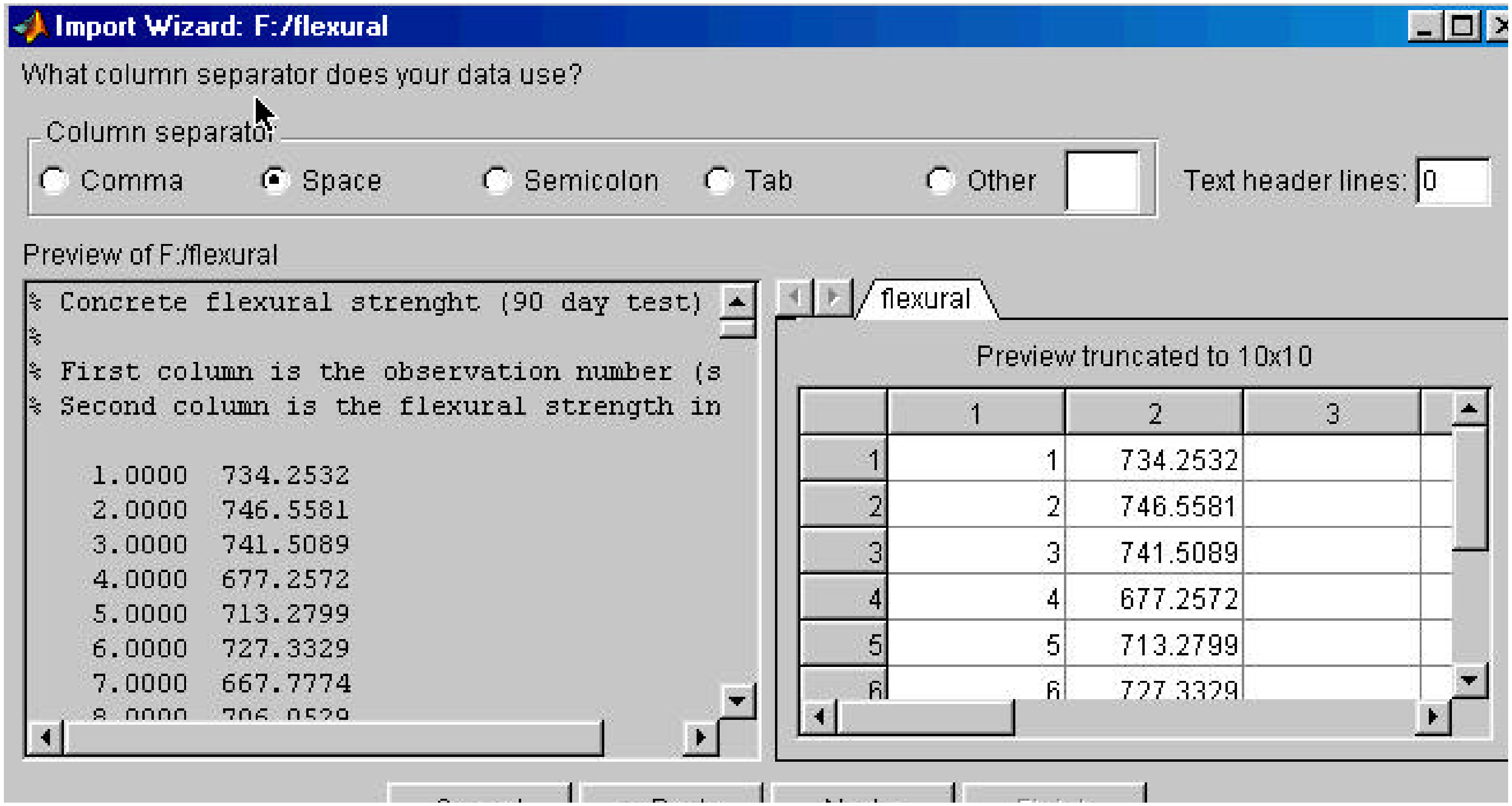
Import Command



	A	B	C	D	E	F	G	H	I
	VarName1	Data	file	with	informati...	for1	Chicago	OHare	VarName9
	NUMBER	NUMBER	NUMBER	NUMBER	TEXT	TEXT	TEXT	TEXT	TEXT
1	%	Data	file	with	informati...	for	Chicago	OHare	
2	%	airport							
3	%	Column	1	=	Time	of	day	(hrs)	
4	%	Column	2	=	Arrivals	per	hour	(aircraft)	
5	%	Column	3	=	Departures	per	hour	(aircraft)	
6	%	Column	4	=	Total	operations	per	hour	(aircraft)
7	0	4	7	11					
8	1	3	2	5					
9	2	2	2	4					
10	3	4	2	6					
11	4	2	8	10					
12	5	5	20	25					
13	6	63	38	101					
14	7	64	68	132					
15	8	87	84	171					
16	9	78	68	146					
17	10	79	67	146					
18	11	78	68	146					
19	12	51	103	154					
20	13	91	72	163					
21	14	77	83	160					
22	15	76	73	149					
23	16	82	70	152					

MATLAB Import Wizard

- Useful tool to import data with various types of variables
- Similar to Excel's import window



What column separator does your data use?

Column separator

○ Comma ● Space ○ Semicolon ○ Tab ○ Other

Text header lines: 0

Preview of F:/flexural

```
% Concrete flexural strenght (90 day test)
%
% First column is the observation number (s
% Second column is the flexural strength in

   1.0000  734.2532
   2.0000  746.5581
   3.0000  741.5089
   4.0000  677.2572
   5.0000  713.2799
   6.0000  727.3329
   7.0000  667.7774
   8.0000  706.0529
```

Preview truncated to 10x10

	1	2	3
1	1	734.2532	
2	2	746.5581	
3	3	741.5089	
4	4	677.2572	
5	5	713.2799	
6	6	727.3329	

Reading the Sample Data File

Method 3 - Use MATLAB fopen and fscan functions

The following script will read the text file ‘ohare_schedule’ using ‘fopen’ and ‘fread’ functions.

```
% Format for data input is a 4-column data file

format long
fid = fopen ('ohare_schedule',’rt’) % ‘rt’ = read text file
y = fscanf(fid, '%g', [4,inf]);     % reads in 4 columns
y = y';

[nrow,ncol] = size(y);              % extracts array size
```

Manipulating Array Data with MATLAB

- Suppose we would like to maintain the results from the data file ‘ohare_schedule’ in four one-dimensional arrays called ‘hour’,’arrivals’,’departures’, and ‘total_ops’.
- Here we use an explicit for-loop to insert values of array ‘y’ into column vectors’hour’,’arrivals’,’departures’, and ‘total_ops’

```
% read data in vector form for each variable
for i=1:1:nrow;
    hour(i)          = y(i,1);
    arrivals(i)      = y(i,2);
    departures(i)    = y(i,3);
    total_ops(i)     = y(i,4);
end
```

Manipulating Array Data with MATLAB (II)

- An easier procedure to assign and create four 1-D arrays is to use an implicit declaration in MATLAB
- Here we use a vector operation (takes less time)

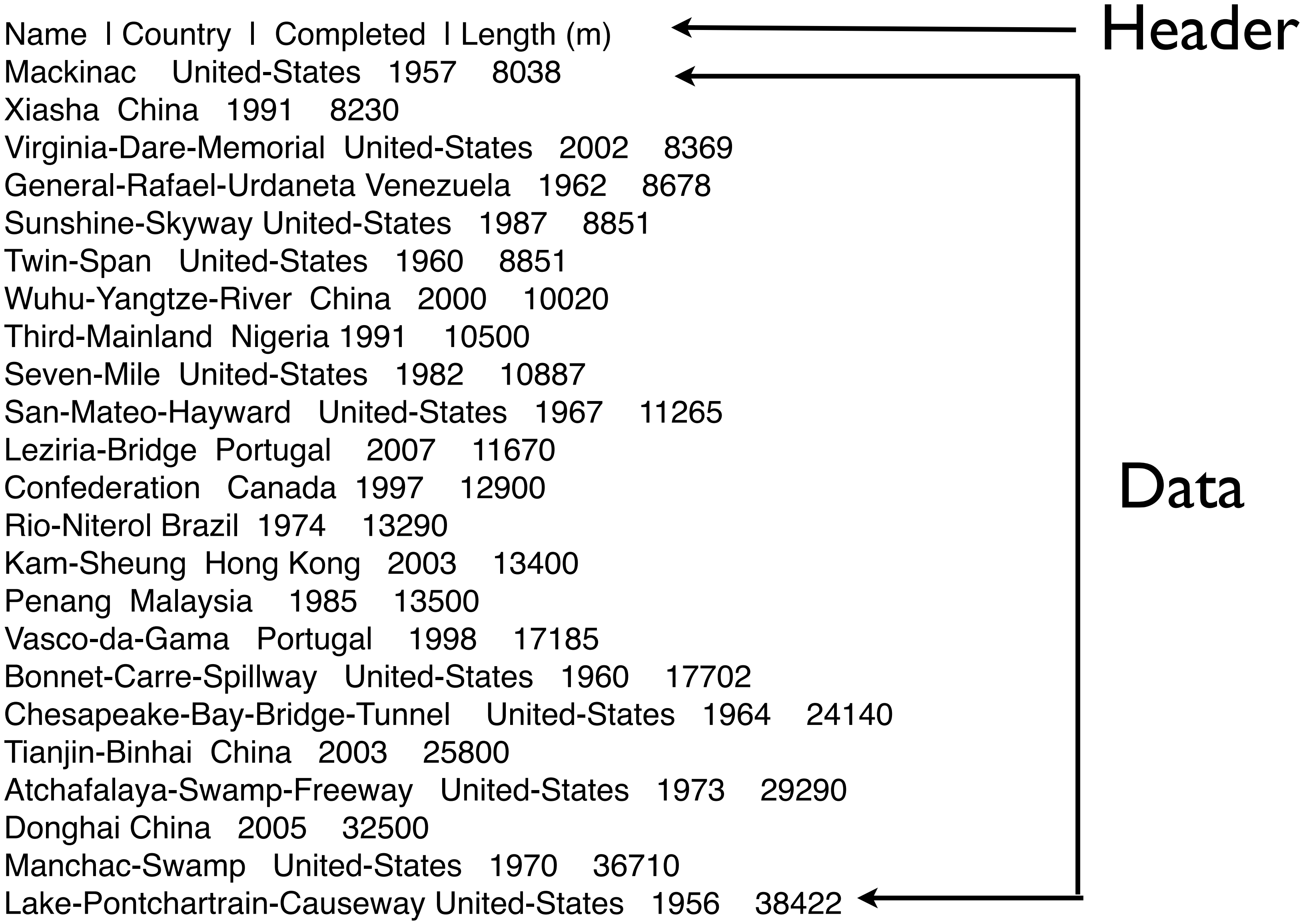
```
% implicit assignment form

hour       = y(: , 1);
arrivals   = y(: , 2);
departures = y(: , 3);
total_ops  = y(: , 4);
```

Reading Data Files

- **Method 4** - Using the **Textscan** Command
- Here is a sample script to read a text file containing data on bridges of the world

```
fid = fopen(‘bridges_of_the_world’)

readHeader = textscan(fid, ‘%s’, 4, ‘delimiter’, ‘|’);

readData = textscan(fid, '%s %s %f %f');

fclose(fid);
```

Data File (bridges_of_the_world)



```
Name  | Country  |  Completed  | Length (m)
Mackinac    United-States   1957    8038
Xiasha  China   1991    8230
Virginia-Dare-Memorial  United-States   2002    8369
General-Rafael-Urdaneta Venezuela   1962    8678
Sunshine-Skyway United-States   1987    8851
Twin-Span   United-States   1960    8851
Wuhu-Yangtze-River  China   2000    10020
Third-Mainland  Nigeria 1991    10500
Seven-Mile  United-States   1982    10887
San-Mateo-Hayward   United-States   1967    11265
Leziria-Bridge  Portugal    2007    11670
Confederation   Canada  1997    12900
Rio-Niterol Brazil  1974    13290
Kam-Sheung  Hong Kong   2003    13400
Penang  Malaysia    1985    13500
Vasco-da-Gama   Portugal    1998    17185
Bonnet-Carre-Spillway   United-States   1960    17702
Chesapeake-Bay-Bridge-Tunnel    United-States   1964    24140
Tianjin-Binhai  China   2003    25800
Atchafalaya-Swamp-Freeway   United-States   1973    29290
Donghai China   2005    32500
Manchac-Swamp   United-States   1970    36710
Lake-Pontchartrain-Causeway United-States   1956    38422
```

Explanations of the Matlab Script

fid = **fopen**(‘bridges_of_the_world’)

- fid - file ID assigned by Matlab
- fopen - “opens” (or reads) the text file called ‘bridges_of_the_world’

readHeader = **textscan**(fid, ‘%s’, 4, ‘delimiter’, ‘|’);

- variable readHeader will store the contents of the first row in the file (‘bridges_of_the_world’)
- textscan reads the first row of the file using ‘%s’,4 (four string variables) with ‘delimiter’ = ‘|’

```
Name  I Country  I  Completed  I Length (m)
Mackinac    United-States   1957    8038
Xiasha  China   1991    8230
Virginia-Dare-Memorial  United-States   2002    8369
```

Explanations of the Matlab Script

readData = **textscan**(fid, '%s %s %f %f');

- variable readData will store the contents of the information starting in the second row (until the end) in the file (‘bridges_of_the_world’)

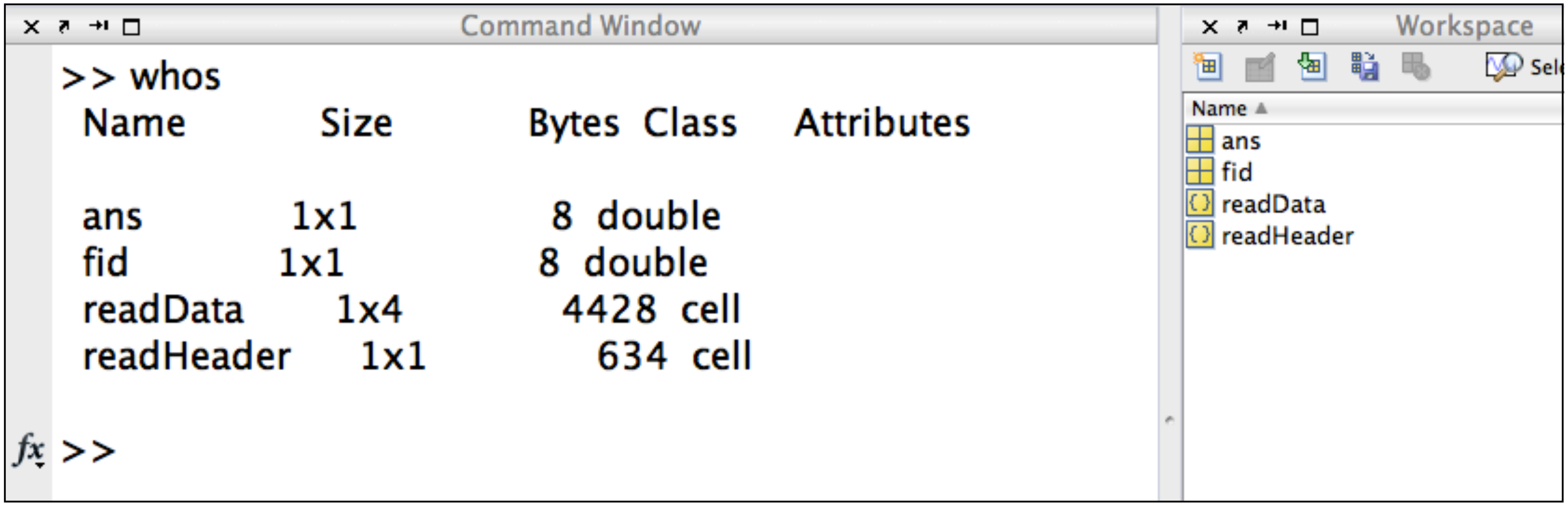
- textscan reads the row data using ‘%s %s’ two string variables and two ‘%f %f’ numerical variables (f stands for floating point)

fclose(fid);

- fclose(fid) closes the file (fid) opened at the beginning of the script

```
Name  I Country  I  Completed  I Length (m)
Mackinac    United-States   1957    8038
Xiasha  China   1991    8230
Virginia-Dare-Memorial  United-States   2002    8369
```

What is Produced by the Matlab Script?



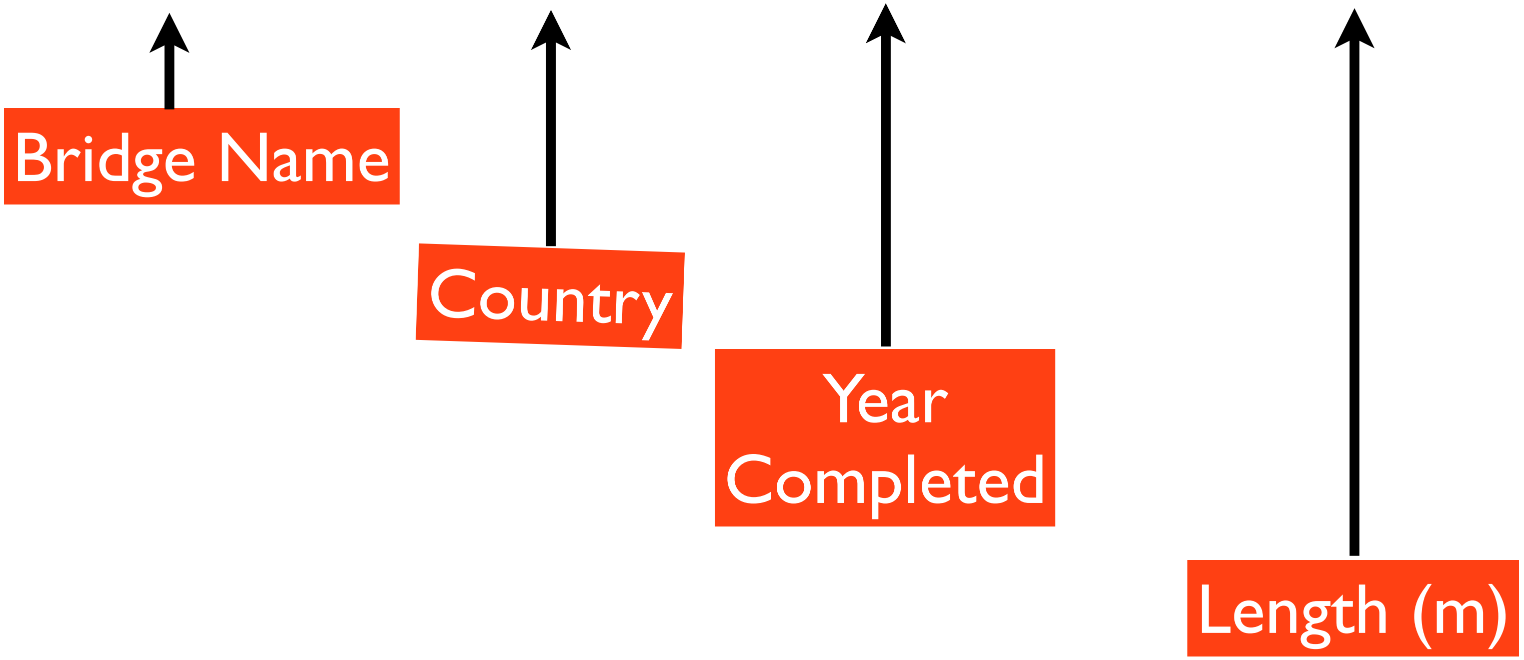
```
>> whos
  Name            Size            Bytes  Class    Attributes

  ans             1x1                 8  double
  fid             1x1                 8  double
  readData        1x4              4428  cell
  readHeader      1x1               634  cell

>>
```

- Four variables (2 are temporary - "ans" and "fid")
- Two variables with the information in the file (***readHeader*** and ***readData***)
- Both variables are **cell arrays (more on this)**

What is a Cell Array?

- A special structure in Matlab to store dissimilar data types (i.e., strings and numeric data)

```
>> readData

readData = {14x1 cell}   {14x1 cell}   [13x1 double]   [13x1 double]
```



Bridge Name

Country

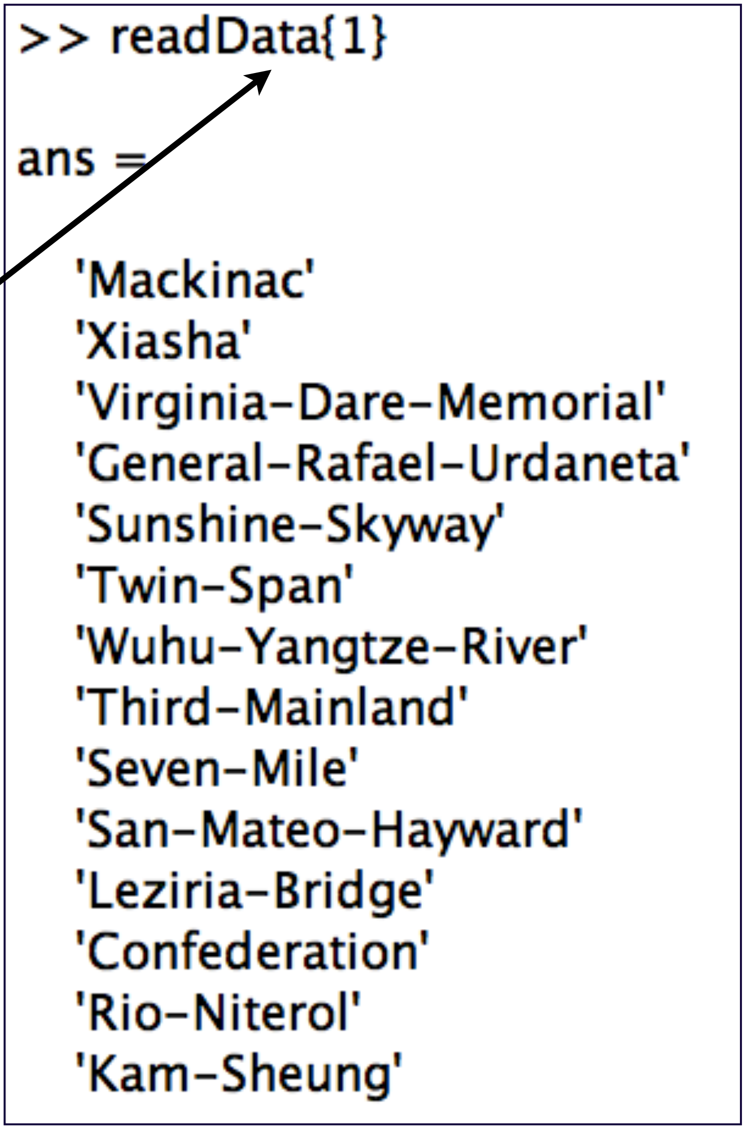
Year Completed

Length (m)

Addressing the Contents of a Cell Array

- Cell arrays are referenced using curly brackets (first) then using standard brackets - to address individual elements of the cell array

- readData{1} references the first column of the array

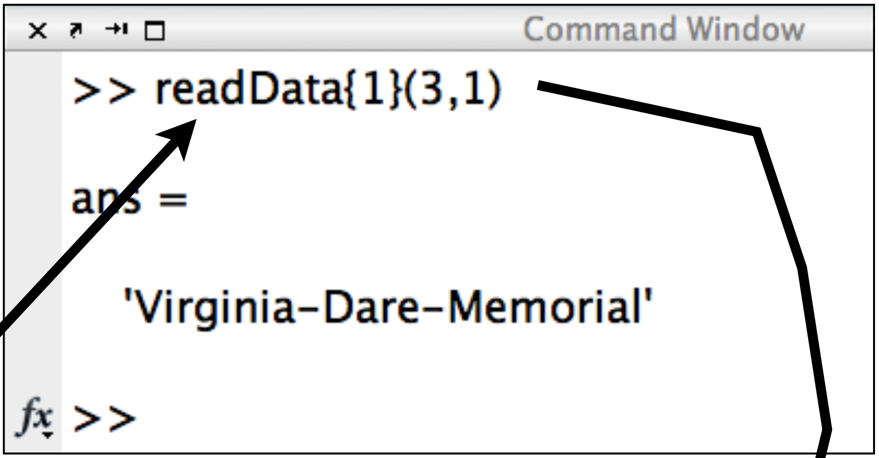


```
>> readData{1}

ans =

    'Mackinac'
    'Xiasha'
    'Virginia-Dare-Memorial'
    'General-Rafael-Urdaneta'
    'Sunshine-Skyway'
    'Twin-Span'
    'Wuhu-Yangtze-River'
    'Third-Mainland'
    'Seven-Mile'
    'San-Mateo-Hayward'
    'Leziria-Bridge'
    'Confederation'
    'Rio-Niterol'
    'Kam-Sheung'
```

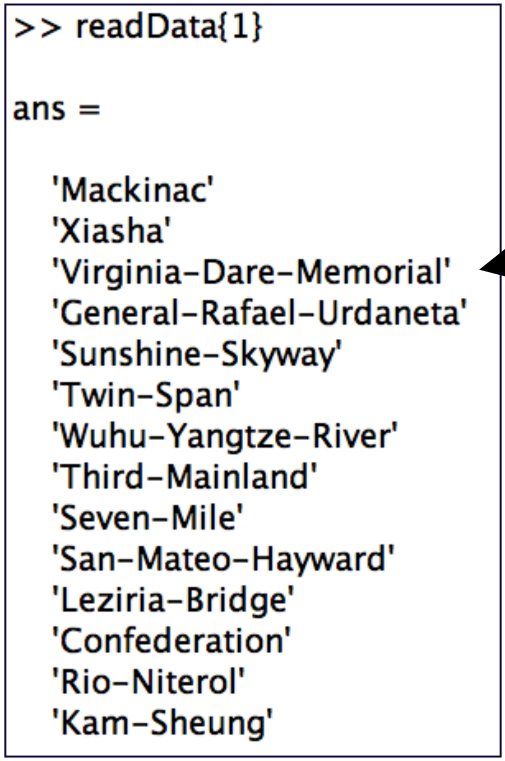
Addressing the Contents of a Cell Array

- Cell arrays are referenced using curly brackets (first) then using standard brackets - to address individual elements of the cell array

- readData{1}(3,1) references the third row element of the cell array



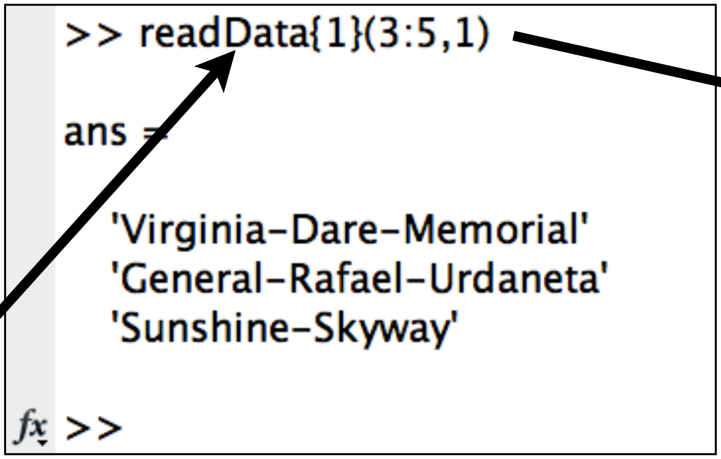
```
>> readData{1}(3,1)

ans =

  'Virginia-Dare-Memorial'

fx >>
```



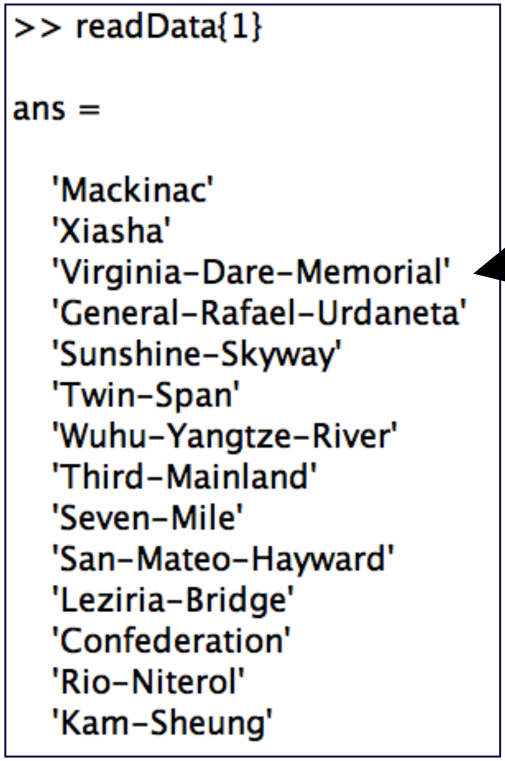
```
>> readData{1}

ans =

    'Mackinac'
    'Xiasha'
    'Virginia-Dare-Memorial'
    'General-Rafael-Urdaneta'
    'Sunshine-Skyway'
    'Twin-Span'
    'Wuhu-Yangtze-River'
    'Third-Mainland'
    'Seven-Mile'
    'San-Mateo-Hayward'
    'Leziria-Bridge'
    'Confederation'
    'Rio-Niterol'
    'Kam-Sheung'
```

Addressing the Contents of a Cell Array

- Cell arrays are referenced using curly brackets (first) then using standard brackets - to address individual elements of the cell array
- readData{1}(3:5,1) references the third, fourth and fifth row elements of the cell array



```
>> readData{1}(3:5,1)

ans =

    'Virginia-Dare-Memorial'
    'General-Rafael-Urdaneta'
    'Sunshine-Skyway'

fx >>
```

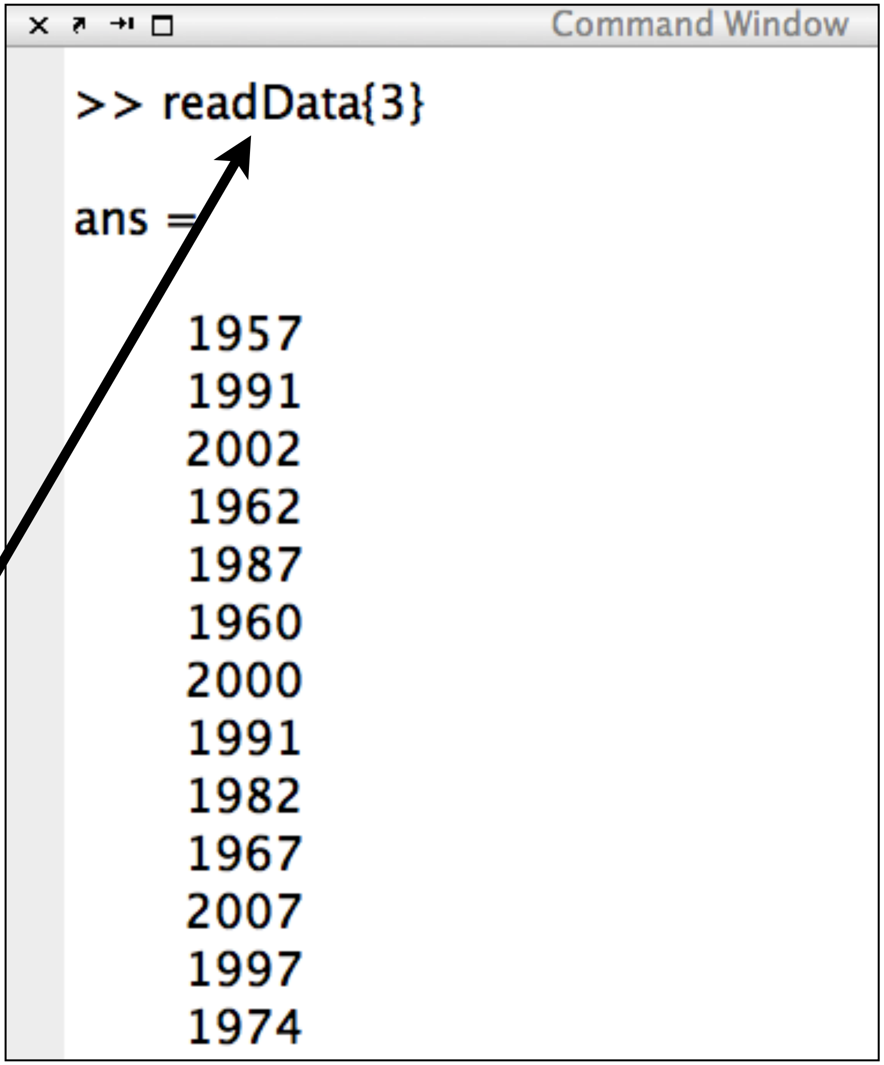


```
>> readData{1}

ans =

    'Mackinac'
    'Xiasha'
    'Virginia-Dare-Memorial'
    'General-Rafael-Urdaneta'
    'Sunshine-Skyway'
    'Twin-Span'
    'Wuhu-Yangtze-River'
    'Third-Mainland'
    'Seven-Mile'
    'San-Mateo-Hayward'
    'Leziria-Bridge'
    'Confederation'
    'Rio-Niterol'
    'Kam-Sheung'
```

Addressing the Contents of a Cell Array

- Cell arrays are referenced using curly brackets (first) then using standard brackets - to address individual elements of the cell array

- readData{3} references all the elements of the third column of the cell array

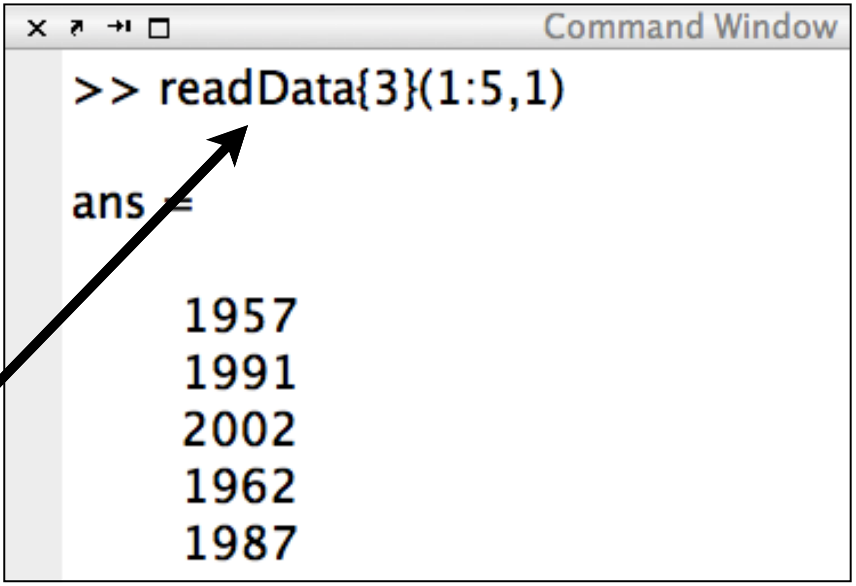


```
>> readData{3}

ans =

        1957
        1991
        2002
        1962
        1987
        1960
        2000
        1991
        1982
        1967
        2007
        1997
        1974
```

Addressing the Contents of a Cell Array

- Cell arrays are referenced using curly brackets (first) then using standard brackets - to address individual elements of the cell array

- readData{3}(1:5,1) references the first five row elements of the third column of the cell array



```
>> readData{3}(1:5,1)

ans =

        1957
        1991
        2002
        1962
        1987
```

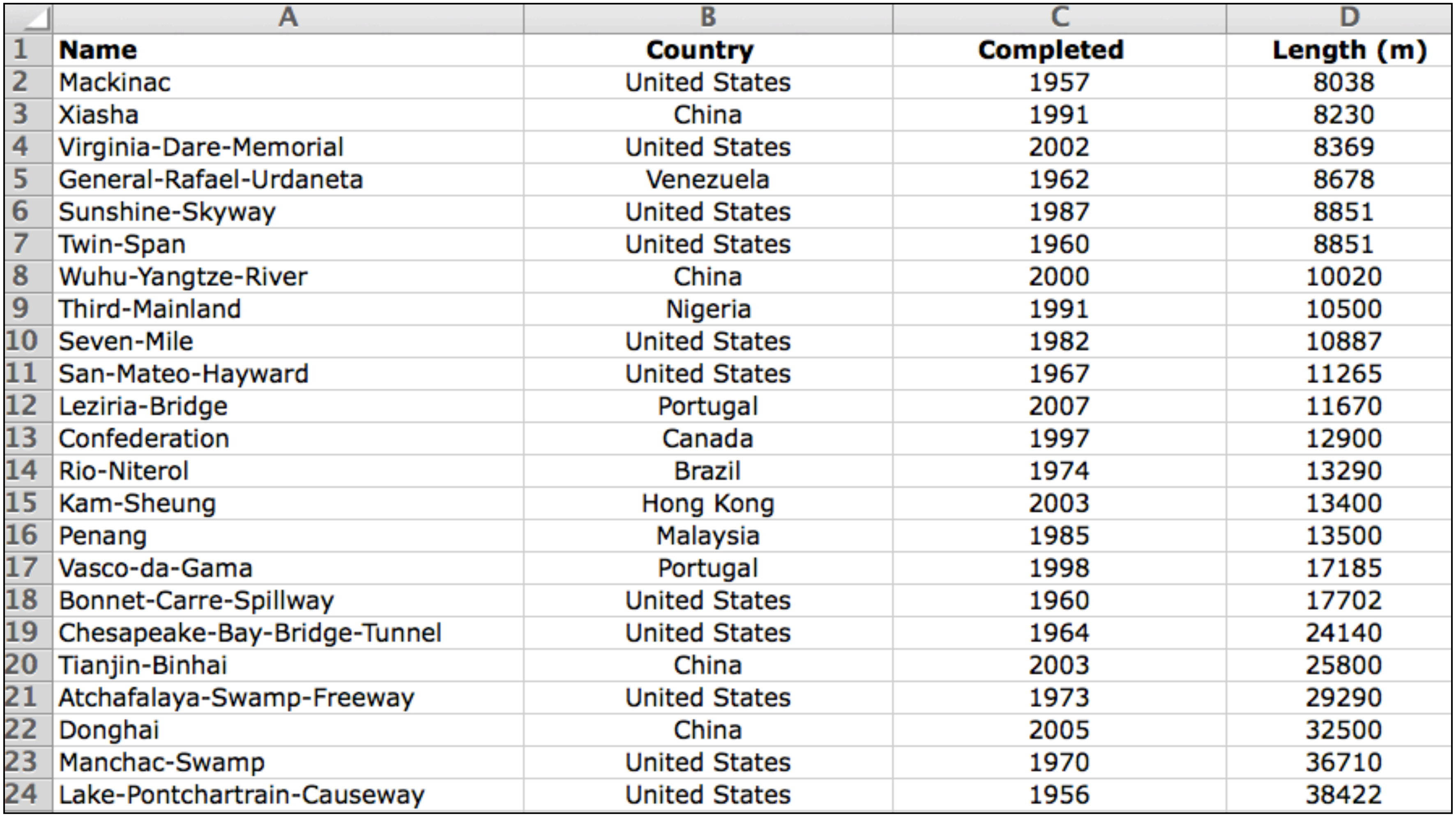
Reading Excel Data Files with Matlab

- **Method 5** - Using the **xlsread** Command
- Here is a sample script to read a data file containing data on bridges of the world

```
[num,txt,raw] = xlsread
('bridges_of_the_world_short.xls','Bridge data');
```

- Reads the Excel worksheet named ‘Bridge data’ contained in file called 'bridges_of_the_world_short.xls'
- Assigns all numeric data to variable **‘num’**
- Assigns all text data to variable called **‘txt’**
- All other unassigned data is stored in variable **‘raw’**

Excel File to be Read

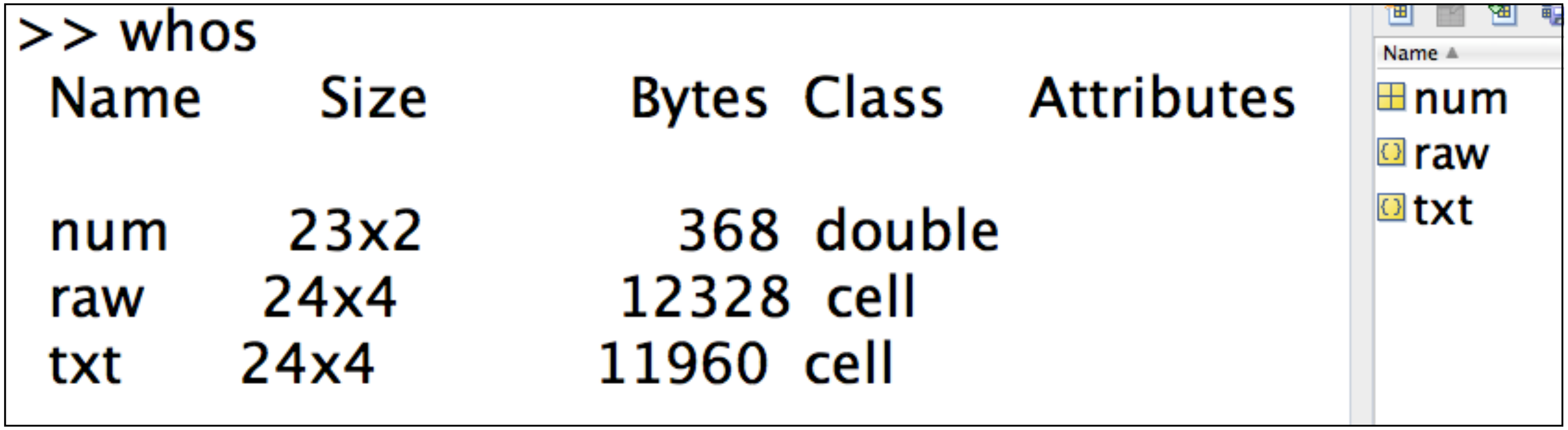


	A	B	C	D
1	**Name**	**Country**	**Completed**	**Length (m)**
2	Mackinac	United States	1957	8038
3	Xiasha	China	1991	8230
4	Virginia-Dare-Memorial	United States	2002	8369
5	General-Rafael-Urdaneta	Venezuela	1962	8678
6	Sunshine-Skyway	United States	1987	8851
7	Twin-Span	United States	1960	8851
8	Wuhu-Yangtze-River	China	2000	10020
9	Third-Mainland	Nigeria	1991	10500
10	Seven-Mile	United States	1982	10887
11	San-Mateo-Hayward	United States	1967	11265
12	Leziria-Bridge	Portugal	2007	11670
13	Confederation	Canada	1997	12900
14	Rio-Niterol	Brazil	1974	13290
15	Kam-Sheung	Hong Kong	2003	13400
16	Penang	Malaysia	1985	13500
17	Vasco-da-Gama	Portugal	1998	17185
18	Bonnet-Carre-Spillway	United States	1960	17702
19	Chesapeake-Bay-Bridge-Tunnel	United States	1964	24140
20	Tianjin-Binhai	China	2003	25800
21	Atchafalaya-Swamp-Freeway	United States	1973	29290
22	Donghai	China	2005	32500
23	Manchac-Swamp	United States	1970	36710
24	Lake-Pontchartrain-Causeway	United States	1956	38422

Bridges_of_the_world_short.xls

What Happens after Executing the One Line Script?

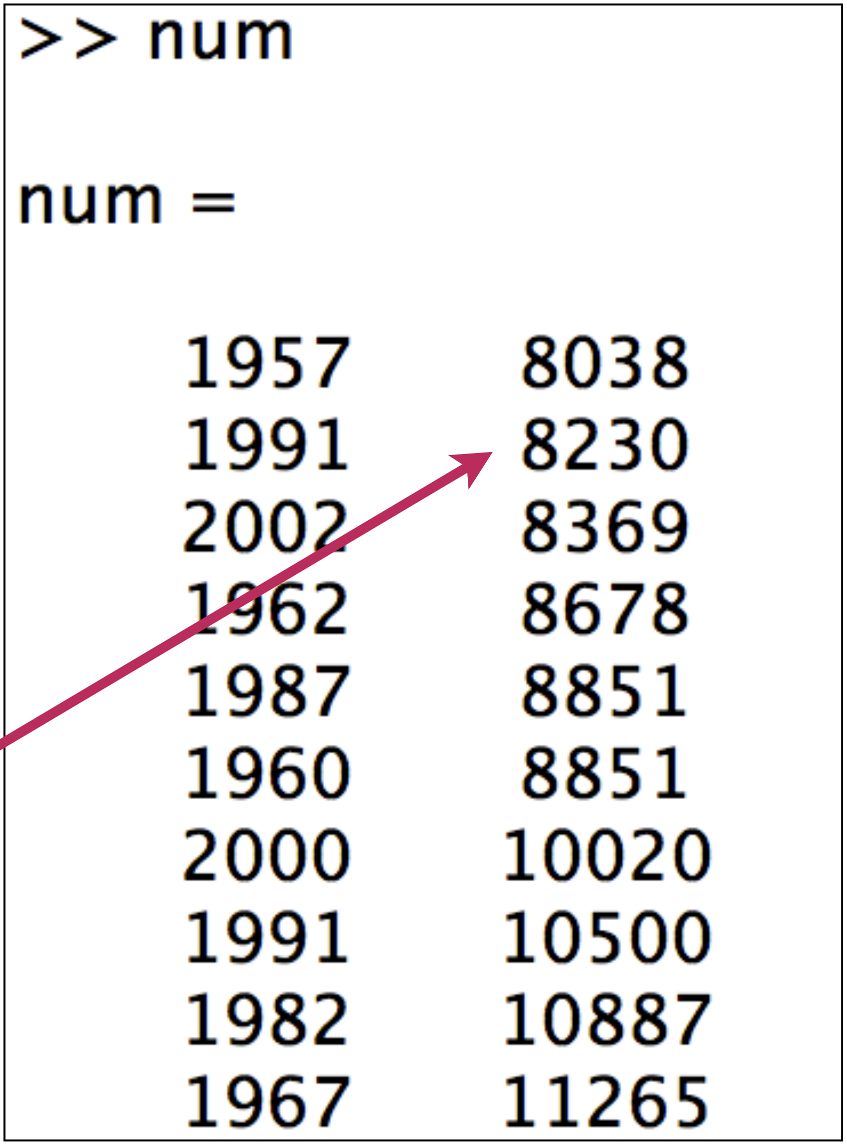
- Three arrays are created using the previous script
- Array ‘**num**” is a standard matrix with size (23 x 2)
- Arrays ‘**raw**’ and ‘**txt**’ are cell arrays (24 x 4) each



```
>> whos
  Name       Size            Bytes  Class     Attributes

  num       23x2               368  double
  raw       24x4             12328  cell
  txt       24x4             11960  cell
```

Observations

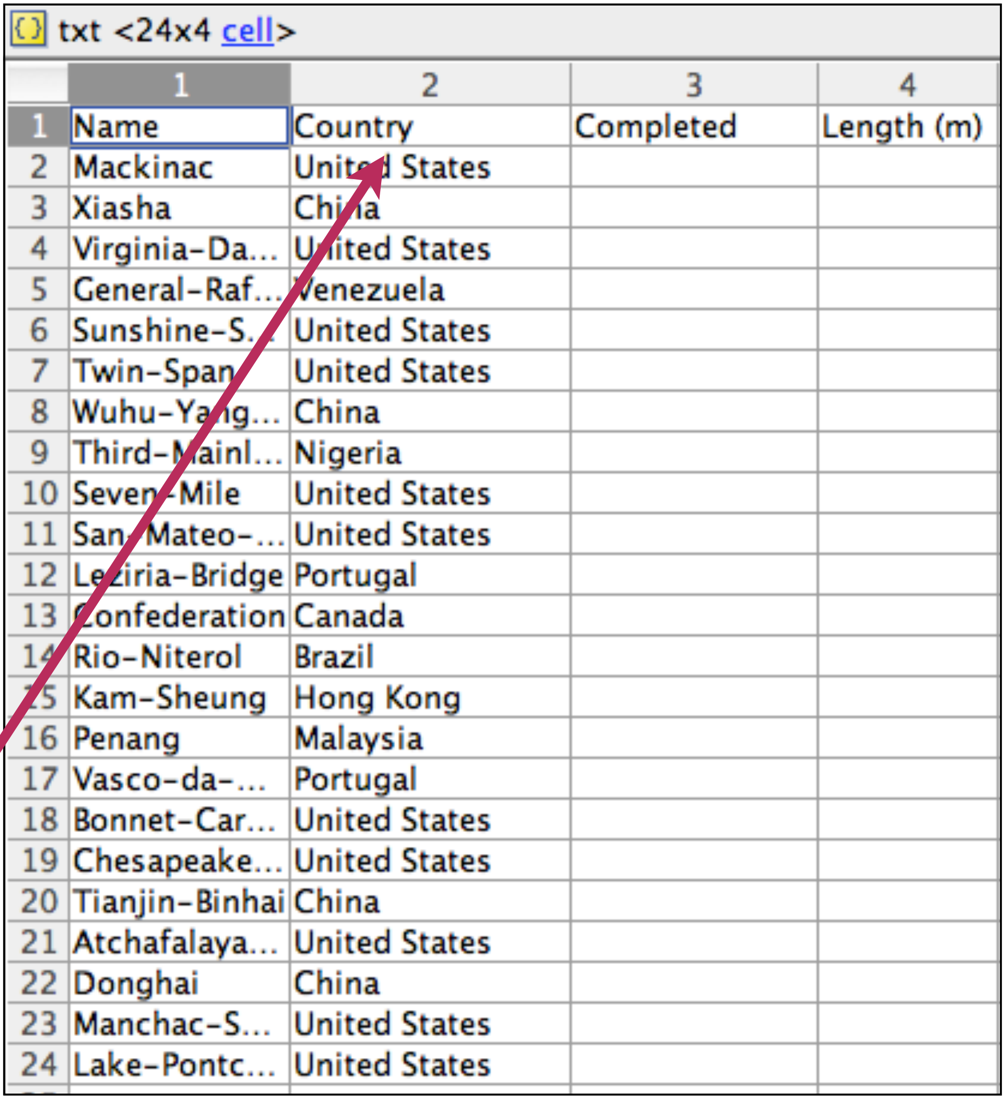
- ‘num’ is a standard numeric array as shown
- Elements of ‘num” can be referenced in the usual (row,column) format
- **num(2,2)=8230**



```
>> num

num =

        1957        8038
        1991        8230
        2002        8369
        1962        8678
        1987        8851
        1960        8851
        2000       10020
        1991       10500
        1982       10887
        1967       11265
```

Observations (2)

- ‘txt’ is a cell array containing **string** data as shown
- Elements of ‘txt” can be referenced using the cell array nomenclature cell{i}(row,column)
- **txt{1,2}=Country**

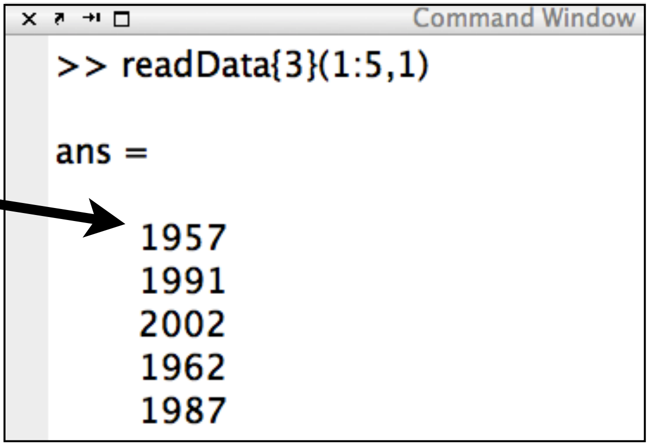


txt <24x4 cell>

	1	2	3	4
1	Name	Country	Completed	Length (m)
2	Mackinac	United States		
3	Xiasha	China		
4	Virginia-Da...	United States		
5	General-Raf...	Venezuela		
6	Sunshine-S...	United States		
7	Twin-Span	United States		
8	Wuhu-Yang...	China		
9	Third-Mainl...	Nigeria		
10	Seven-Mile	United States		
11	San-Mateo-...	United States		
12	Leziria-Bridge	Portugal		
13	Confederation	Canada		
14	Rio-Niterol	Brazil		
15	Kam-Sheung	Hong Kong		
16	Penang	Malaysia		
17	Vasco-da-...	Portugal		
18	Bonnet-Car...	United States		
19	Chesapeake...	United States		
20	Tianjin-Binhai	China		
21	Atchafalaya...	United States		
22	Donghai	China		
23	Manchac-S...	United States		
24	Lake-Pontc...	United States		

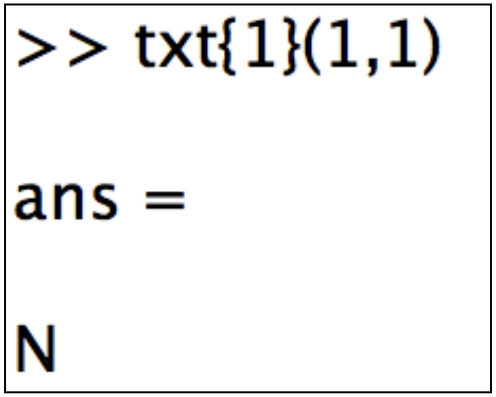
Note Differences in How Cell Arrays Store Information

- In previous case, a cell array storing numerical data can be referenced
- **readData{3}(1:5,1)**



```
>> readData{3}(1:5,1)

ans =

        1957
        1991
        2002
        1962
        1987
```

- In this last case, the cell array contains string information
- **txt{1}(1,2)=N**



```
>> txt{1}(1,1)

ans =

N
```

Matlab **xlsread** can Read a Range in an Excel

- The Matlab statement:
- [num,txt,raw] = xlsread ('bridges_of_the_world_short.xls','Bridge data (A2:D24)');
- Reads the Excel file but only across the range specified (A2:D24)
- This is useful if you know the data structure of the file you are reading

Exporting Capabilities (I)

- The following code exports data from all four one-dimensional arrays to a text file called ‘output.txt’
- The format ‘a’ implies appending information to this file

```
fid = fopen ('output.txt','a');
fprintf(fid, '%4.0f %4.0f %4.0f %4.0f\n',y');
status = fclose(fid);
```

Note that a specific format with four digits has been used in this example.

Displaying Output on the Command Window

- Use function ‘disp’ to display output to the screen.
- Typically used in conjunction with ‘num2str’ to convert numerical to string variables

Example:

```
x = 35
displ([‘This is a test to display  ’, num2str(x), ‘   here’])
```

Results:

```
This is a test to display   35  here
```