

# MATLAB FUNDAMENTALS AND PROGRAMMING TECHNIQUES

(BASIC)

*Tim Yeh*  
*support@terasoft.com.tw*

# Course Outline

- Working with the MATLAB User Interface
- Variables and Commands
- Array Creation and Analysis
- Working with Data Files
- Visualization with Array
- Automating Commands with Scripts
- Appendix: Data Type



# How to Use This Manual

- Code font is used for code, function names and URLs. It also occurs on the slides in one of the lower corners as reference to relevant files or commands for the example.

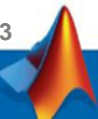
```
>> command_line_code  
>>[a,b,c] = command_in_file(d,f);
```

- At times, code may run off the line. The line continuation character (...) is used to show this. These are valid MATLAB statements if typed as shown, including carriage returns.

```
>> [CFlowAmounts, CFlowDates, TFactors] = ...  
cfamounts(couponRate, settle, maturity);
```

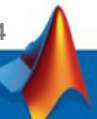
- Menu items, options and key names are highlighted in bold in the notes sections.  
Use **Ctrl+C** to break out of execution.  
Click on **File**, **Set Path ...** to open the Path Browser.

**>> try this at the prompt**

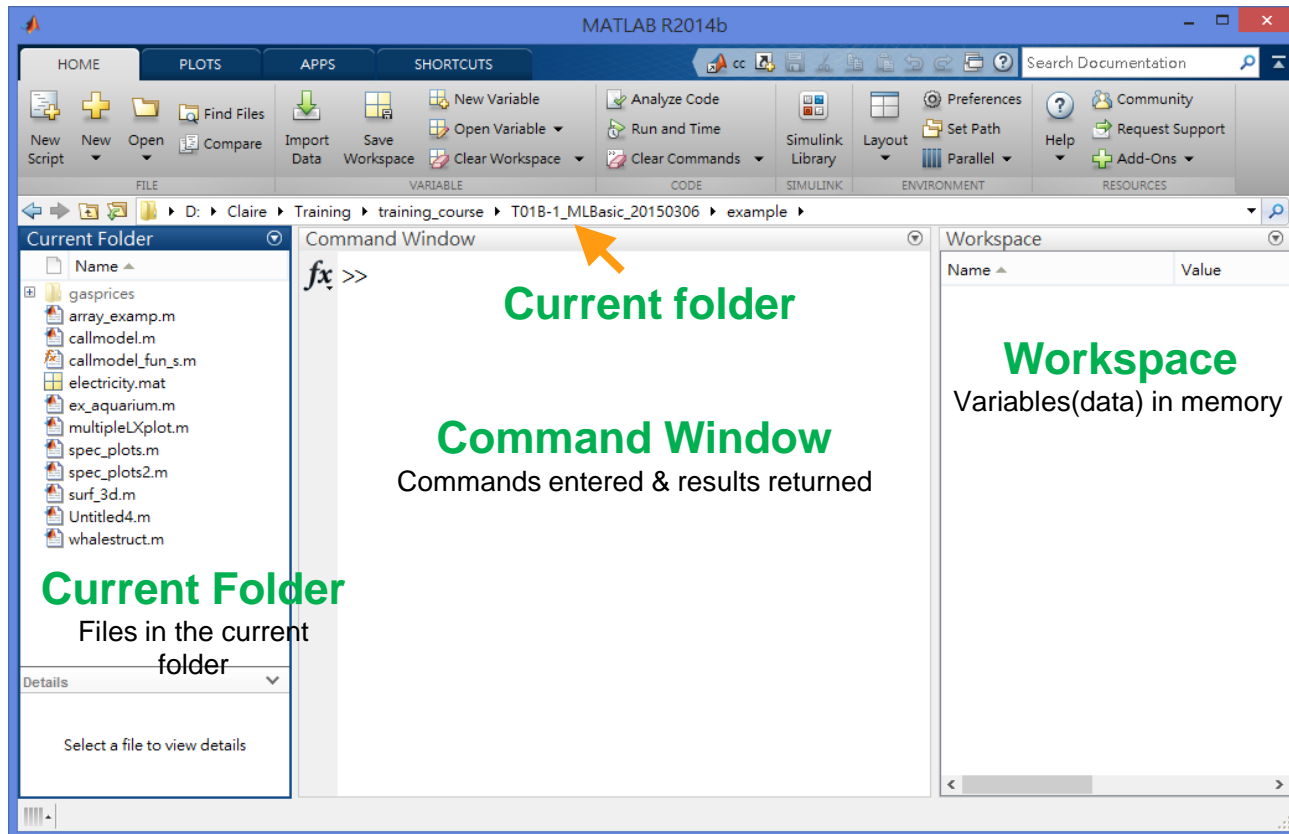


# Course Outline

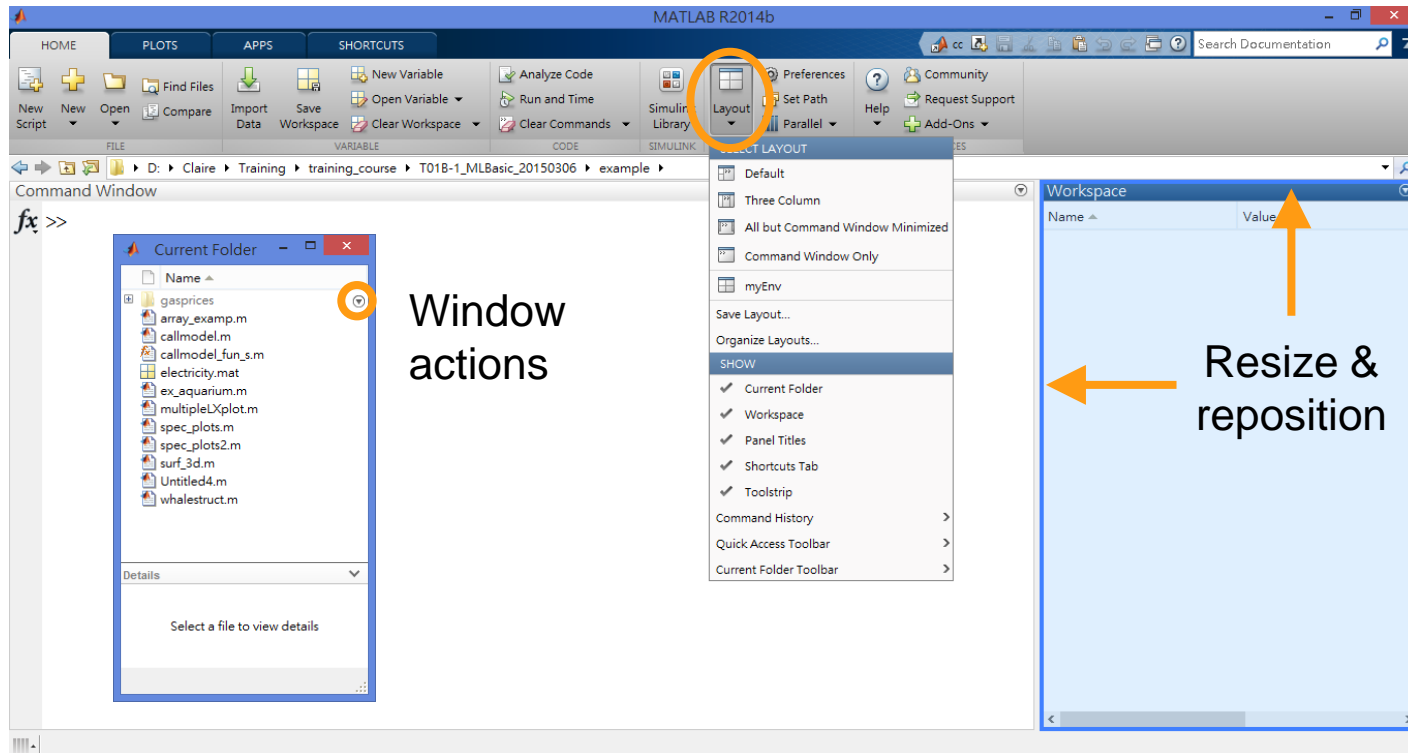
- Working with the MATLAB User Interface
  - Variables and Commands
  - Array Creation and Analysis
  - Working with Data Files
  - Visualization with Array
  - Automating Commands with Scripts
  - Appendix: Data Type



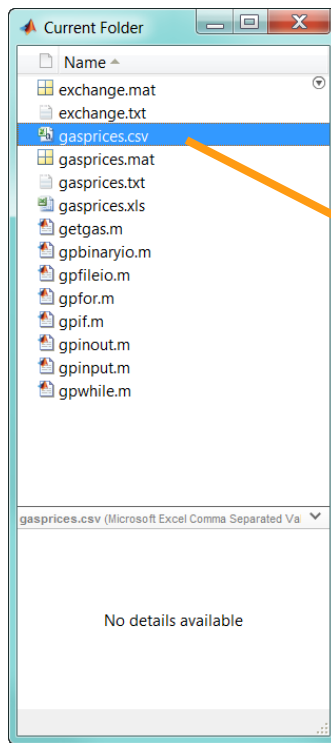
# The MATLAB® Desktop



# Customizing the Desktop



# Course Example: Gas Price Data



gasprices.csv - Microsoft Excel

Average Annual Gasoline (Petrol) Retail Prices in Selected Countries [US\$ per gallon]

Source: US Dept. of Energy  
<http://www.eia.doe.gov/emeu/aer/txt/ptb1108.html>

Year	Australia	Canada	France	Germany	Italy	Japan	Mexico	South Korea	UK	USA
1990	1.87	1.87	3.03	2.03	4.39	3.10	1.00	2.03	2.02	1.10
1991	1.96	1.92	3.45	2.9	4.5	3.46	1.3	2.49	3.01	1.14
1992	1.89	1.73	3.56	3.27	4.53	3.58	1.5	2.65	3.06	1.13
1993	1.73	1.57	3.41	3.07	3.68	4.16	1.56	2.88	2.84	1.11
1994	1.84	1.45	3.59	3.52	3.7	4.36	1.48	2.87	2.99	1.11
1995	1.95	1.53	4.26	3.96	4	4.43	1.11	2.94	3.21	1.15
1996	2.12	1.61	4.41	3.94	4.39	3.64	1.25	3.18	3.34	1.23
1997	2.05	1.62	4	3.53	4.07	3.26	1.47	3.34	3.83	1.23
1998	1.63	1.38	3.87	3.34	3.84	2.82	1.49	3.04	4.06	1.06
1999	1.72	1.52	3.85	3.42	3.87	3.27	1.79	3.8	4.29	1.17
2000	1.94	1.86	3.8	3.45	3.77	3.65	2.01	4.18	4.58	1.51
2001	1.71	1.72	3.51	3.4	3.57	3.27	2.2	3.76	4.13	1.46
2002	1.76	1.69	3.62	3.67	3.74	3.15	2.24	3.84	4.16	1.36
2003	2.19	1.99	4.35	4.59	4.53	3.47	2.04	4.11	4.7	1.59
2004	2.72	2.37	4.99	5.24	5.29	3.93	2.03	4.51	5.56	1.88
2005	3.23	2.89	5.46	5.66	5.74	4.28	2.22	5.28	5.97	2.3
2006	3.54	3.26	5.88	6.03	6.1	4.47	2.31	5.92	6.36	2.59
2007	3.85	3.59	6.6	6.88	6.73	4.49	2.4	6.21	7.13	2.8
2008	4.45	4.08	7.51	7.75	7.63	5.74	2.45	5.83	7.42	3.27

# Interactively Importing

The screenshot illustrates the process of importing data from an Excel file into MATLAB. It features three main windows:

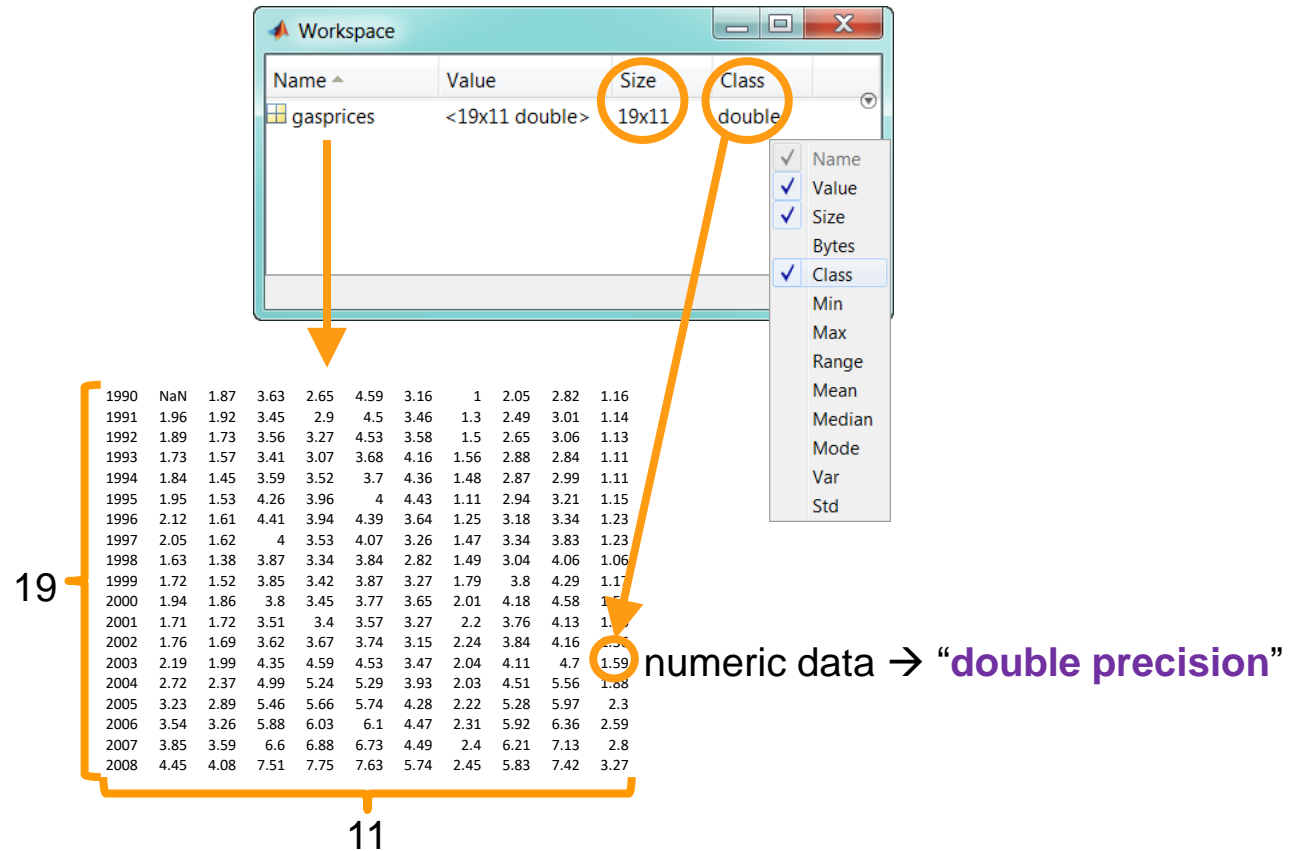
- Current Folder:** Shows the file 'gasprices.xls' selected. The context menu is open, with 'Import Data...' highlighted.
- Import Wizard:** Displays the 'Import' tab for 'gasprices.xls'. The range is 'A6:K24' and 'Variable Names Row' is '5'. The 'Column vector' option is selected and circled in orange. The 'Import Selection' button is also highlighted with an orange arrow.
- Workspace:** Shows the imported data as a table with columns for 'Year' and various countries (Australia, Canada, France, Germany, Italy, Japan, Mexico, South Korea, UK, USA). The data is stored as 'double' arrays.

Below the 'Current Folder' window, there are icons for Excel and a text file, with an orange bracket indicating the source of the data.

Year	Australia	Canada	France	Germany	Italy	Japan	Mexico	SouthKo...	UK	USA
1990	NaN	1.8700	3.6300	2.6500	4.5900	3.1600	1	2.0500	2.8200	
1991	1.9600	1.9200	3.4500	2.9000	4.5000	3.4600	1.3000	2.4900	3.0100	
1992	1.8900	1.7300	3.5600	3.2700	4.5300	3.5800	1.5000	2.6500	3.0600	
1993	1.7300	1.5700	3.4100	3.0700	3.6800	4.1600	1.5600	2.8800	2.8400	
1994	1.8400	1.4500	3.5900	3.5200	3.7000	4.3600	1.4800	2.8700	2.9900	
1995	1.9500	1.5300	4.2600	3.9600	4	4.4300	1.1100	2.9400	3.2100	
1996	2.1200	1.6100	4.4100	3.9400	4.3900	3.6400	1.2500	3.1800	3.3400	
1997	2.0500	1.6200	4	3.5300	4.0700	3.2600	1.4700	3.3400	3.8300	
1998	1.6300	1.3800	3.8700	3.3400	3.8400	2.8200	1.4900	3.0400	4.0600	
1999	1.7200	1.5200	3.8500	3.4200	3.8700	3.2700	1.7900	3.8000	4.2900	
2000	1.9400	1.8600	3.8000	3.4500	3.7700	3.6500	2.0100	4.1800	4.5800	
2001	1.7100	1.7200	3.5100	3.4000	3.5700	3.2700	2.2000	3.7600	4.1300	
2002	1.7600	1.6900	3.6200	3.6700	3.7400	3.1500	2.2400	3.8400	4.1600	
2003	2.1900	1.9900	4.3500	4.5900	4.5300	3.4700	2.0400	4.1100	4.7000	
2004	2.7200	2.3700	4.9900	5.2400	5.2900	3.9300	2.0300	4.5100	5.5600	1.8800
2005	3.2300	2.8900	5.4600	5.6600	5.7400	4.2800	2.2200	5.2800	5.9700	2.3000
2006	3.5400	3.2600	5.8800	6.0300	6.1000	4.4700	2.3100	5.9200	6.3600	2.5900
2007	3.8500	3.5900	6.6000	6.8800	6.7300	4.4900	2.4000	6.2100	7.1300	2.8000
2008	4.4500	4.0800	7.5100	7.7500	7.6300	5.7400	2.4500	5.8300	7.4200	3.2700



# Variables in the Base Workspace



# The Variable Editor

The Variable Editor interface shows the 'gasprices' variable selected in the Workspace. The Variable Editor window displays the data for 'gasprices' as a table with 19 rows and 11 columns. The first column is 'Year' and the subsequent columns are 'Gas Price' for each year from 1990 to 2008. A calculator is overlaid on the data table, and an orange arrow points from the 'gasprices' variable in the Variable Editor to a zoomed-in view of the data table.

Year	Gas Price	Gas Price	Gas Price	Gas Price
1990	1.9600	1.8700	3.6300	2.6500
1991	1.9600	1.9200	3.4500	2.9000
1992	1.8900	1.7300	3.5600	3.2700
1993	1.7300	1.5700	3.4100	2.0700
1994	1.8400	1.4500	3.5900	3.5200
1995	1.9500	1.5300	4.2600	3.9600
1996	2.1200	1.6100	4.4100	3.9400
1997	2.0500	1.6200	4.3500	4.0700
1998	1.6300	1.3800	3.8700	3.3400
1999	1.7200	1.5200	3.8500	3.4200
2000	1.9400	1.8600	3.8000	3.4500
2001	1.7100	1.7200	3.5100	3.4000
2002	1.7600	1.6900	3.6200	3.6700
2003	2.1900	1.9900	4.3500	4.5900
2004	2.7200	2.3700	4.9900	5.2400
2005	3.2300	2.8900	5.4600	5.6600
2006	3.5400	3.2600	5.8800	6.0300
2007	3.8500	3.5900	6.6000	6.8800
2008	4.4500	4.0800	7.5100	7.7500

# New Variables

The screenshot shows the MATLAB environment with the 'gasprices' dataset loaded. The 'Workspace' window displays the following variables:

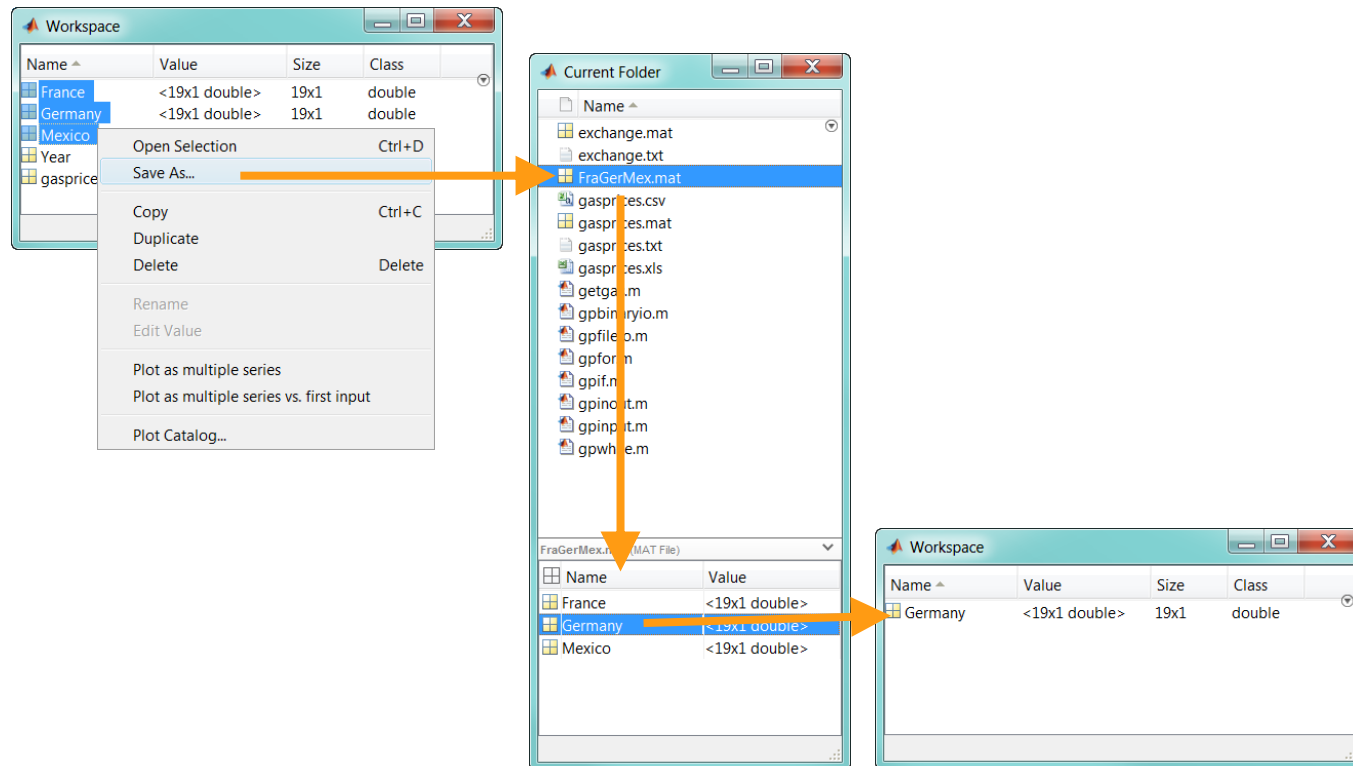
Name	Value	Size	Class
gasprices	<19x11 double>	19x11	double
gasprices1	<19x1 double>	19x1	double

An orange arrow points from the 'gasprices' variable in the Workspace to the 'New Numeric Array' dialog box, which is currently set to 'New Numeric Array' and 'k11 double'.

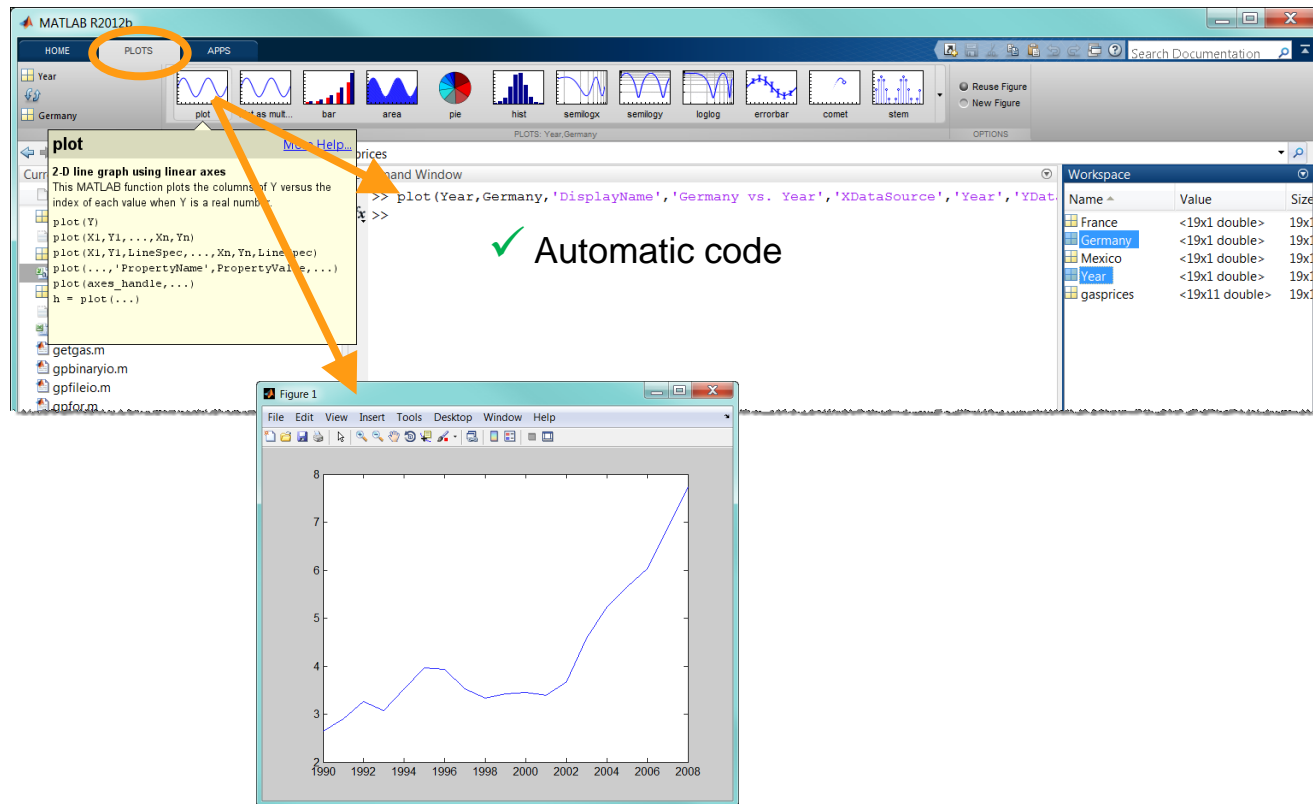
The 'gasprices' dataset is displayed in the main window as follows:

	1	2	3	4	5	6	7	8	9
1	1990	1.9600	1.8700	3.6300	2.6500	4.5900	3.1600	1	2.0500
2	1991	1.9600	1.9200	3.4500	2.9000	4.5000	3.4600	1.3000	2.4900
3	1992	1.8900	1.7300	3.5600	3.2700	4.5300	3.5800	1.5000	2.6500
4	1993	1.7300	1.5700	3.4100	3.0700	3.6800	4.1600	1.5600	2.8800
5	1994	1.8400	1.4500	3.5900	3.5200	3.7000	4.3600	1.4800	2.5900
6	1995	1.9500	1.5300	4.2600	3.9600	4	4.4300	1.1100	2.9400
7	1996	2.1200	1.6100	4.4100	3.9400	4.3900	3.6400	1.2500	3.1800
8	1997	2.0500	1.6200	4	3.5300	4.0700	3.2600	1.4700	3.3400
9	1998	1.6300	1.3800	3.8700	3.3400	3.8400	2.8200	1.4900	3.0400
10	1999	1.7200	1.5200	3.8500	3.4200	3.8700	3.2700	1.7900	3.8000
11	2000	1.9400	1.8600	3.8000	3.4500	3.7700	3.6500	2.0100	4.1800
12	2001	1.7100	1.7200	3.5100	3.4000	3.5700	3.2700	2.2000	3.7600
13	2002	1.7600	1.6900	3.6200	3.6700	3.7400	3.1500	2.2400	3.8400
14	2003	2.1900	1.9900	4.3500	4.5900	4.5300	3.4700	2.0400	4.1100
15	2004	2.7200	2.3700	4.9900	5.2400	5.2900	3.9300	2.0300	4.5100
16	2005	3.2300	2.8900	5.4600	5.6600	5.7400	4.2800	2.2200	5.2800
17	2006	3.5400	3.2600	5.8800	6.0300	6.1000	4.4700	2.3100	5.9200
18	2007	3.8500	3.5900	6.6000	6.8800	6.7300	4.4900	2.4000	6.2100
19	2008	4.4500	4.0800	7.5100	7.7500	7.6300	5.7400	2.4500	5.8300
20									

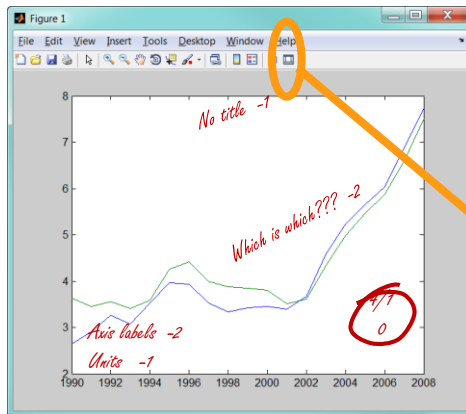
# Saving and Loading Variables



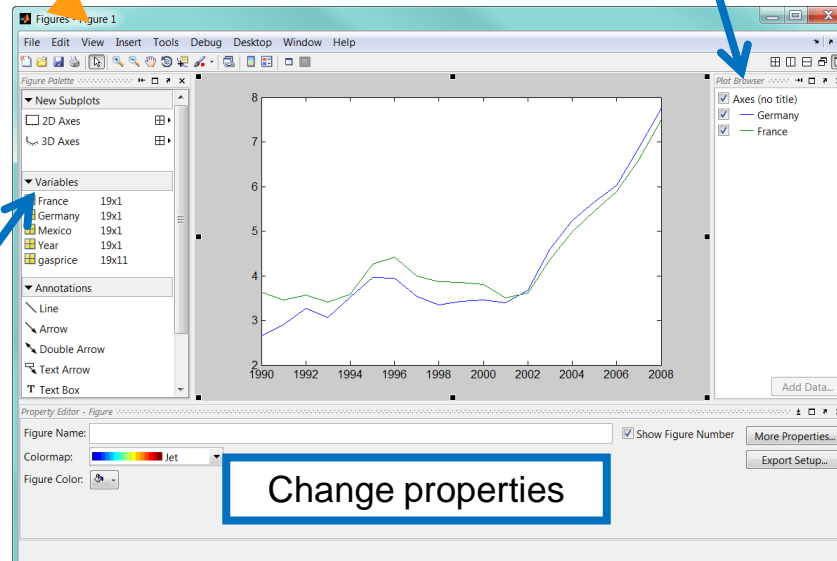
# Plotting the Data



# Plot Tools

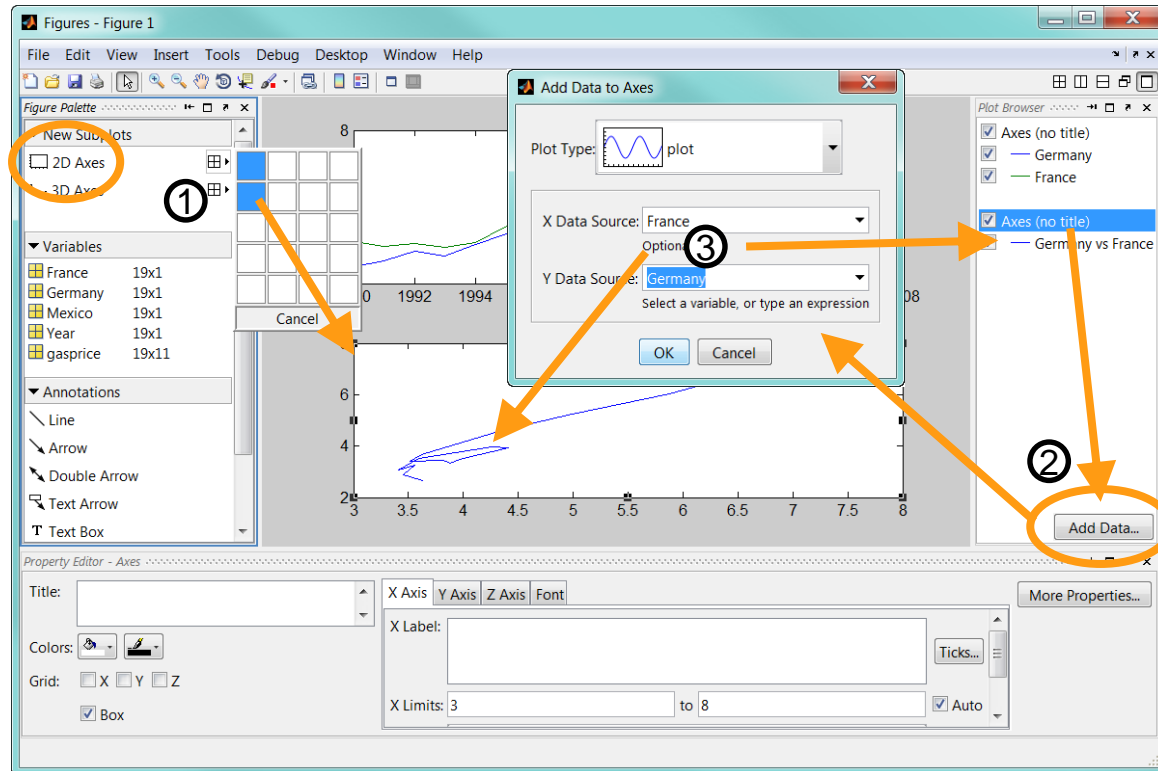


Add plot elements

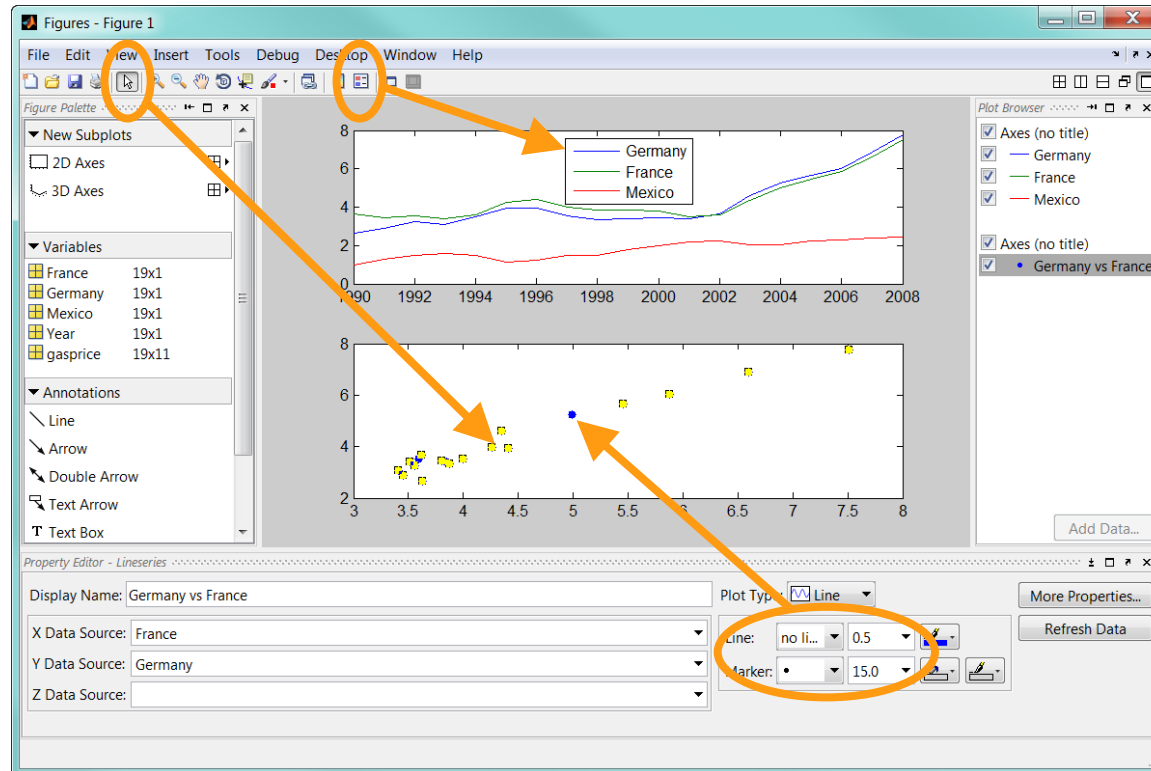


Select plot components

# Multiple Plots

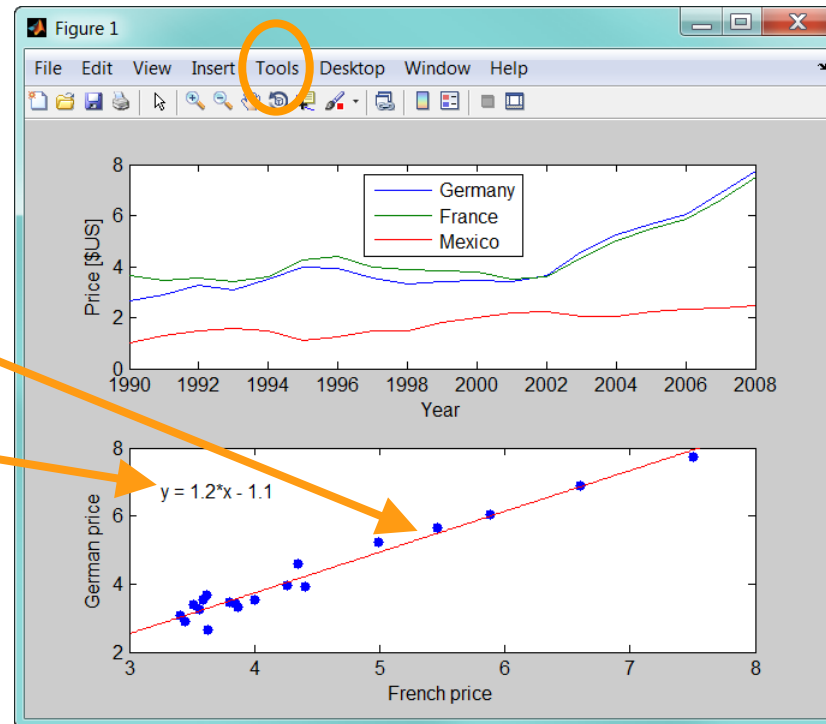
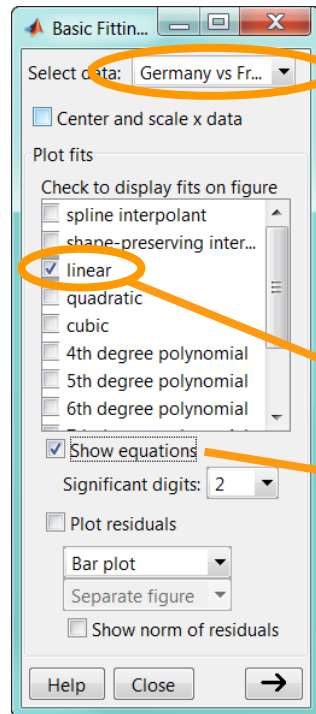


# Formatting the Plot

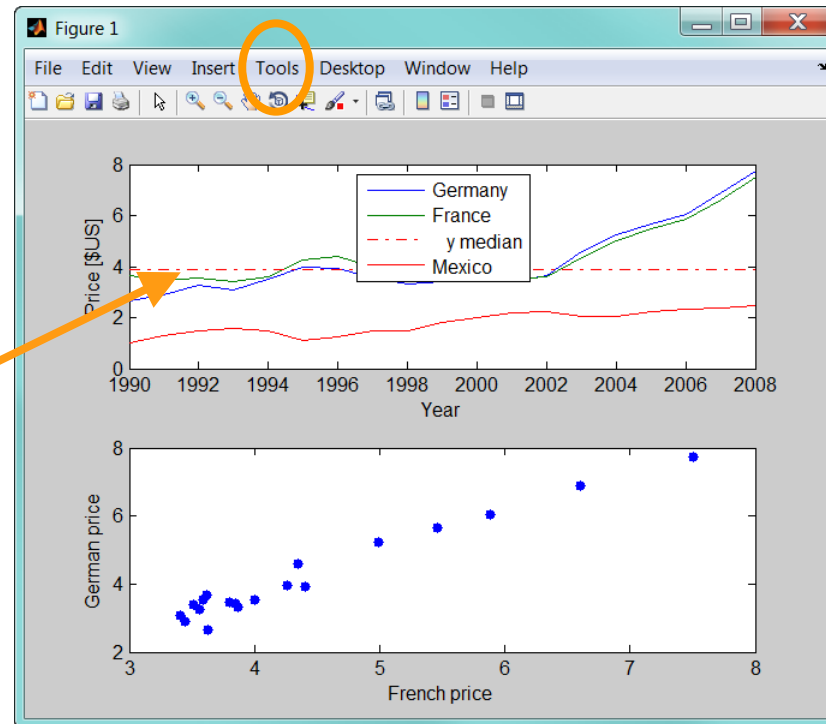
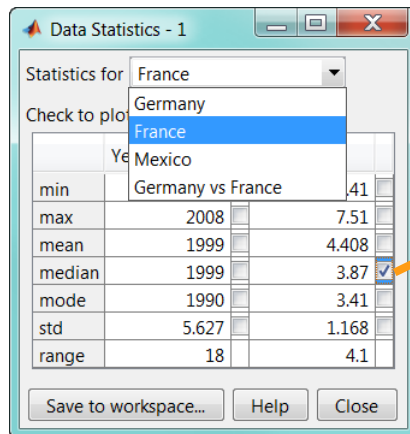




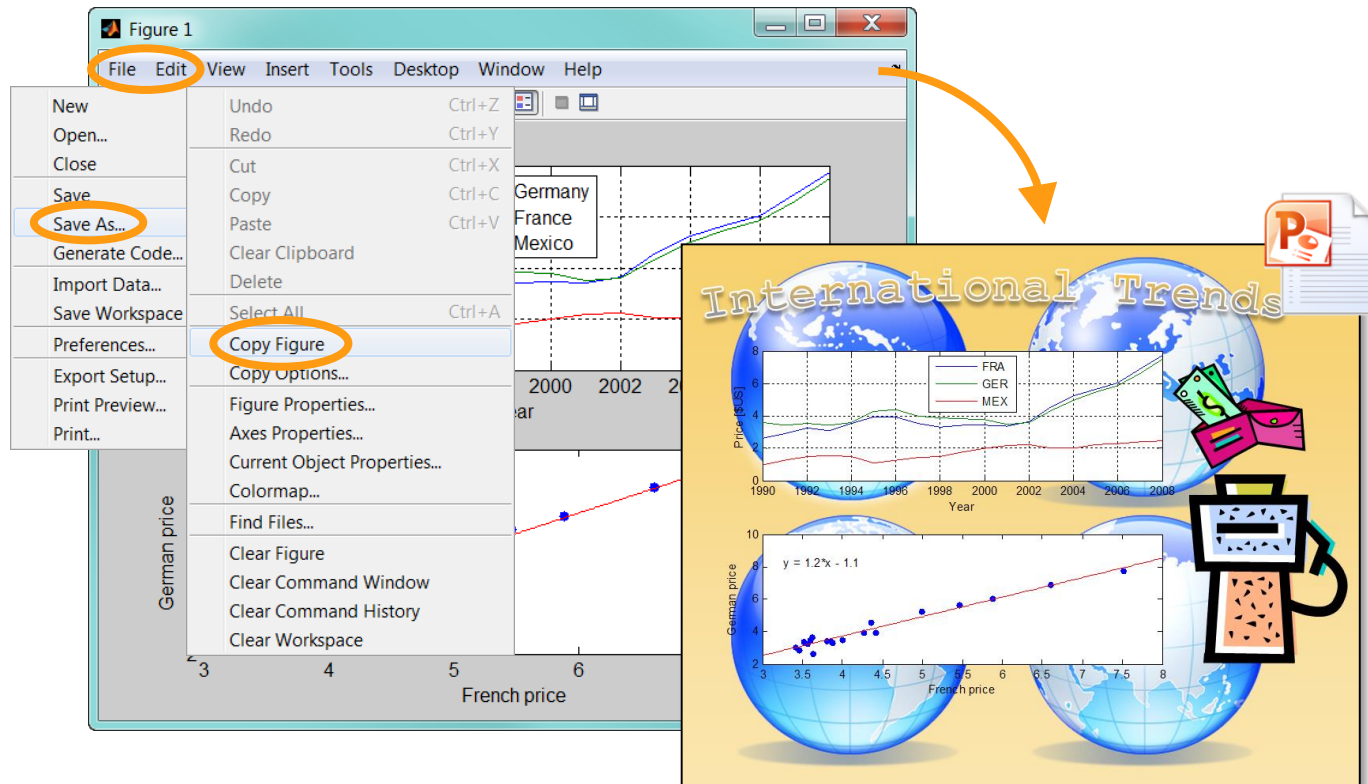
# Basic Fitting Tool



# Data Statistics Tool

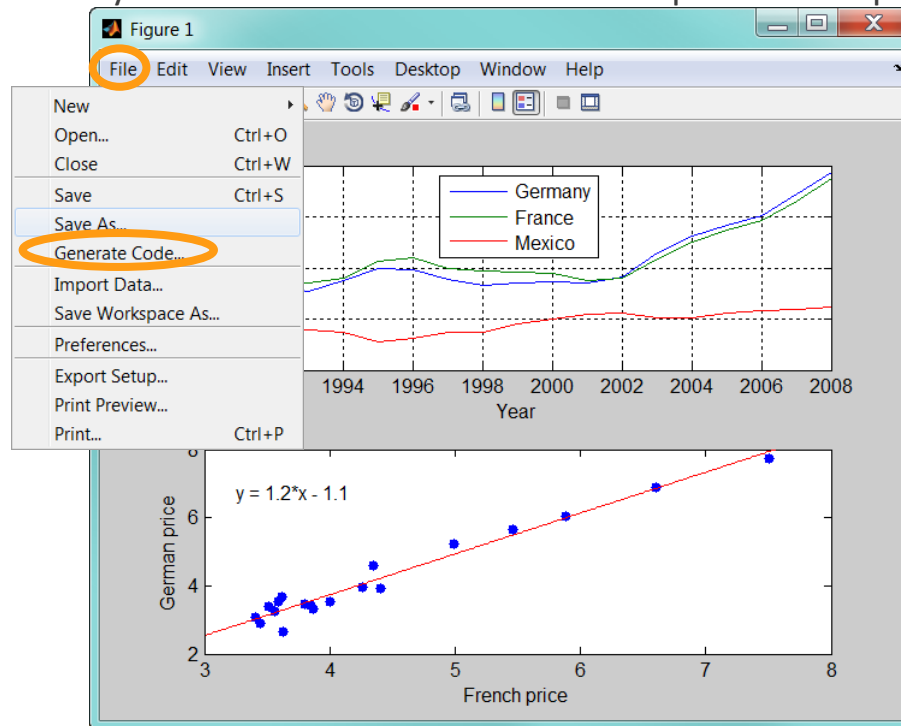


# Exporting to Another Application

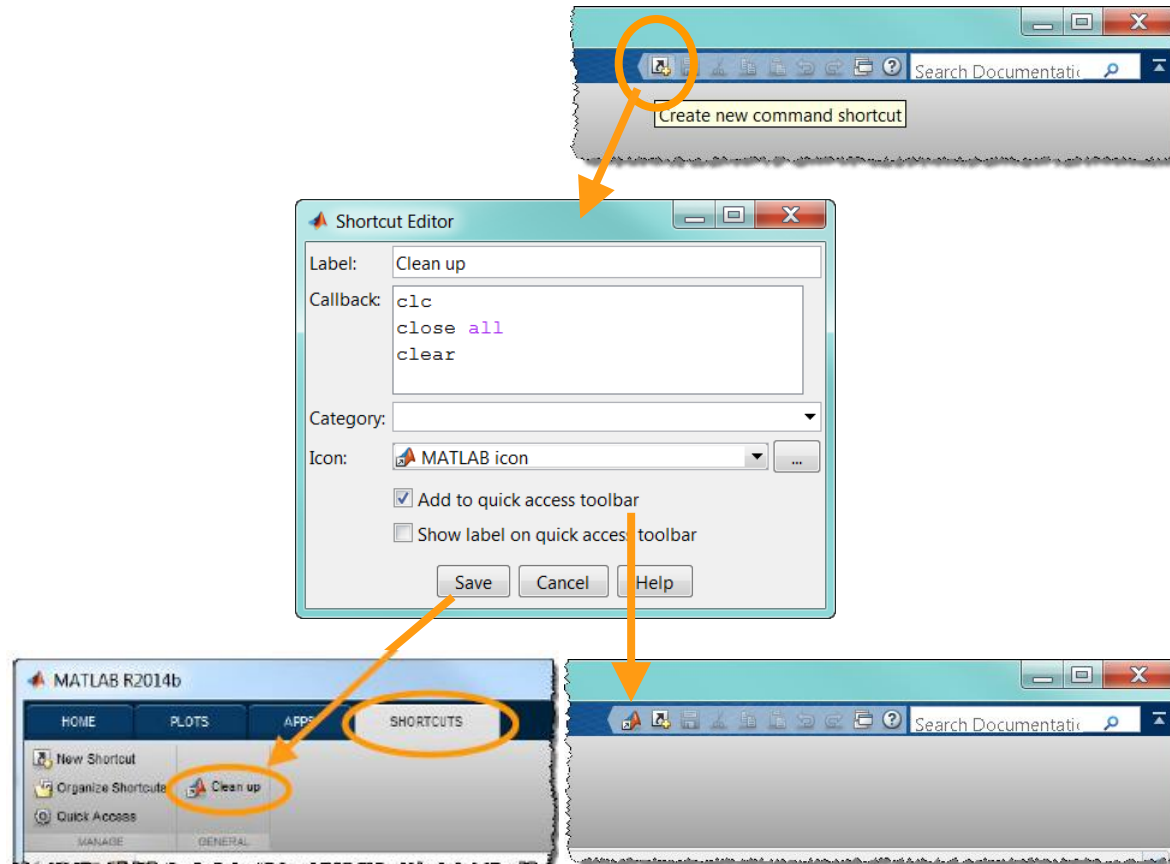


# Generate M-file

- Automatically Create MATLAB Commands to Reproduce Graphics



# Shortcuts

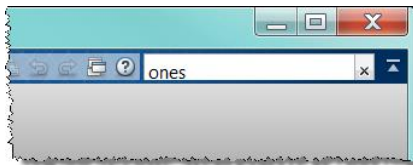


# Help and Documentation

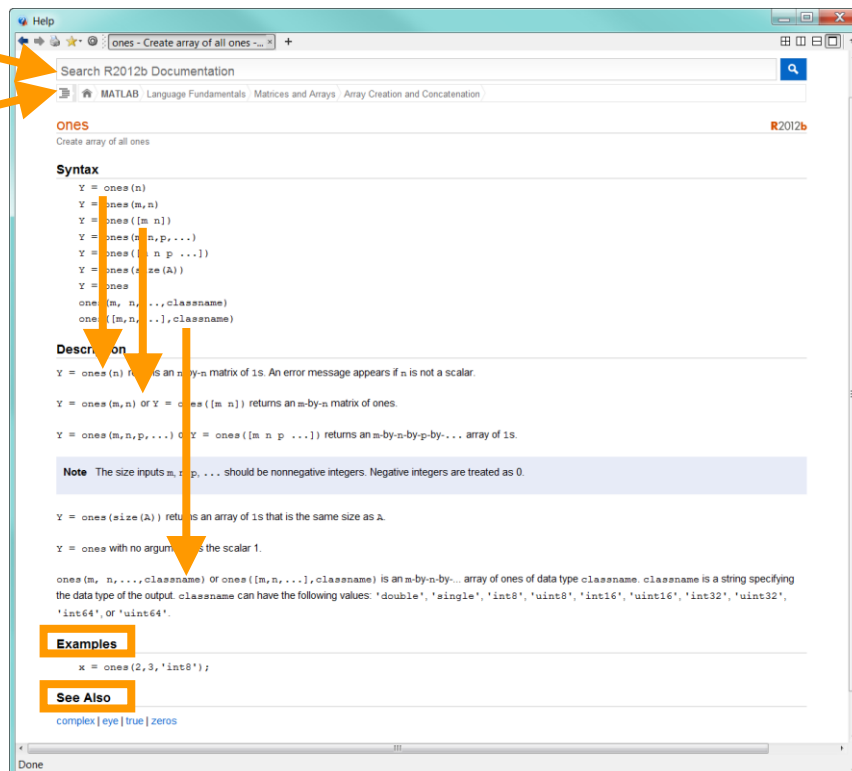
search



browse

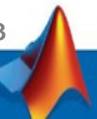


help  
doc  
docsearch



# Chapter 1 Test Your Knowledge

1. Where does MATLAB display a listing of the variables currently in memory and their associated attributes?
  - A. Command Window
  - B. Workspace browser
  - C. Current Directory browser
  - D. Command History
2. The default MATLAB variable type is:
  - A. Single
  - B. Double
  - C. Cell



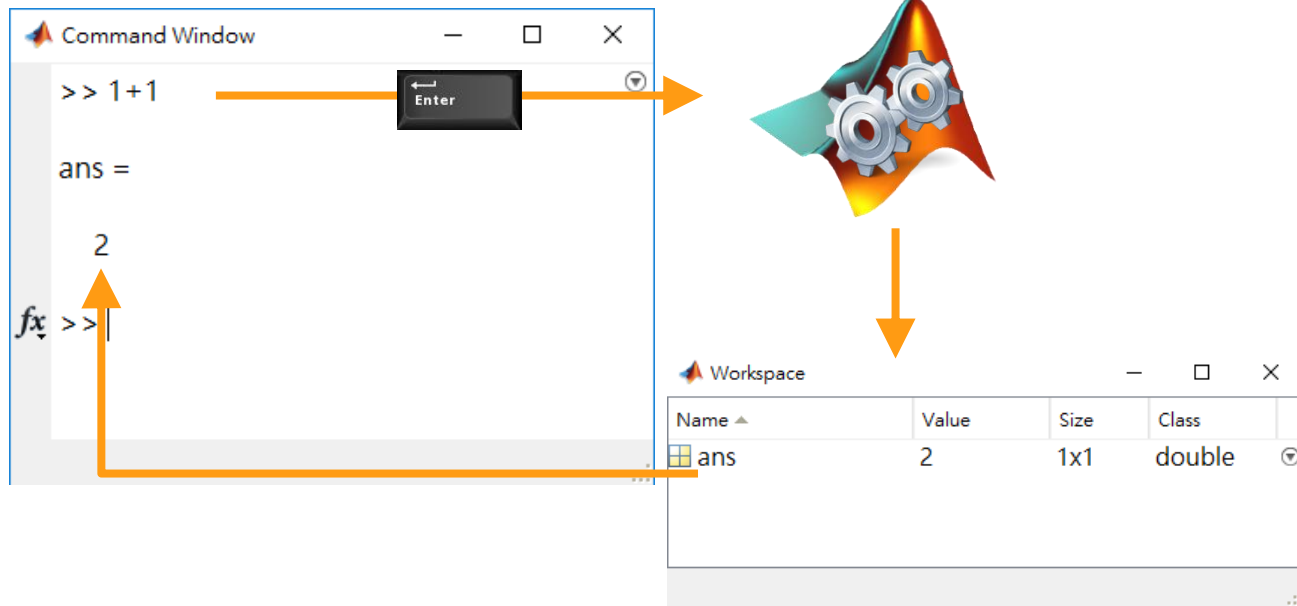
# Course Outline

- Working with the MATLAB User Interface
- Variables and Commands
- Array Creation and Analysis
- Working with Data Files
- Visualization with Array
- Automating Commands with Scripts
- Appendix: Data Type

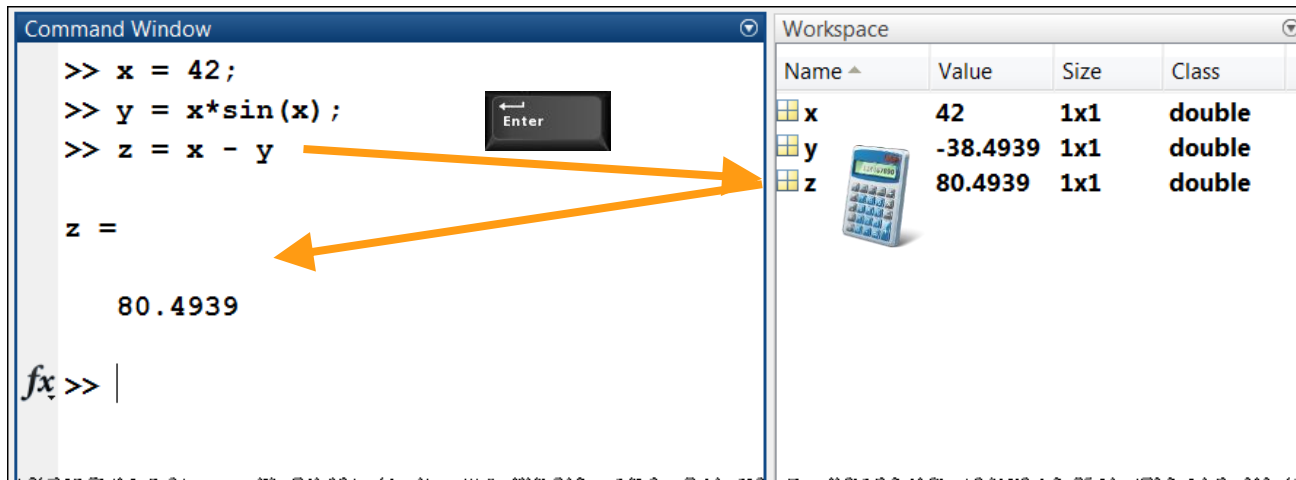
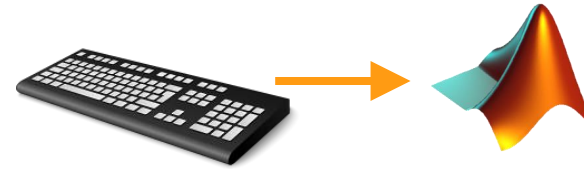


# High-level Calculator

Operators: + - \* / ^



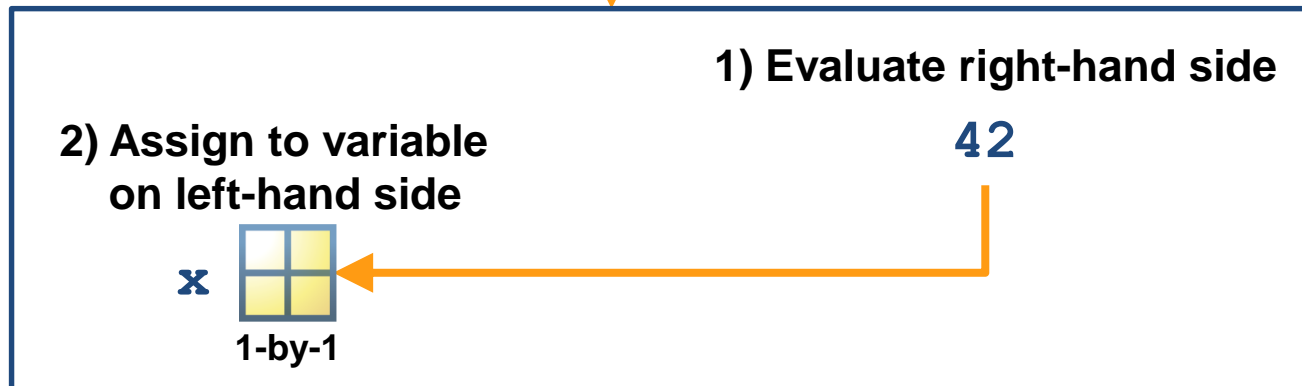
# MATLAB® Commands



# Storing Data in Variables

```
>> x = 6*7;
```

Assignment



# Chapter 2 Test Your Knowledge

1. The default MATLAB variable type is:

- A. Single
- B. Double
- C. Cell

2. Calculate question below :

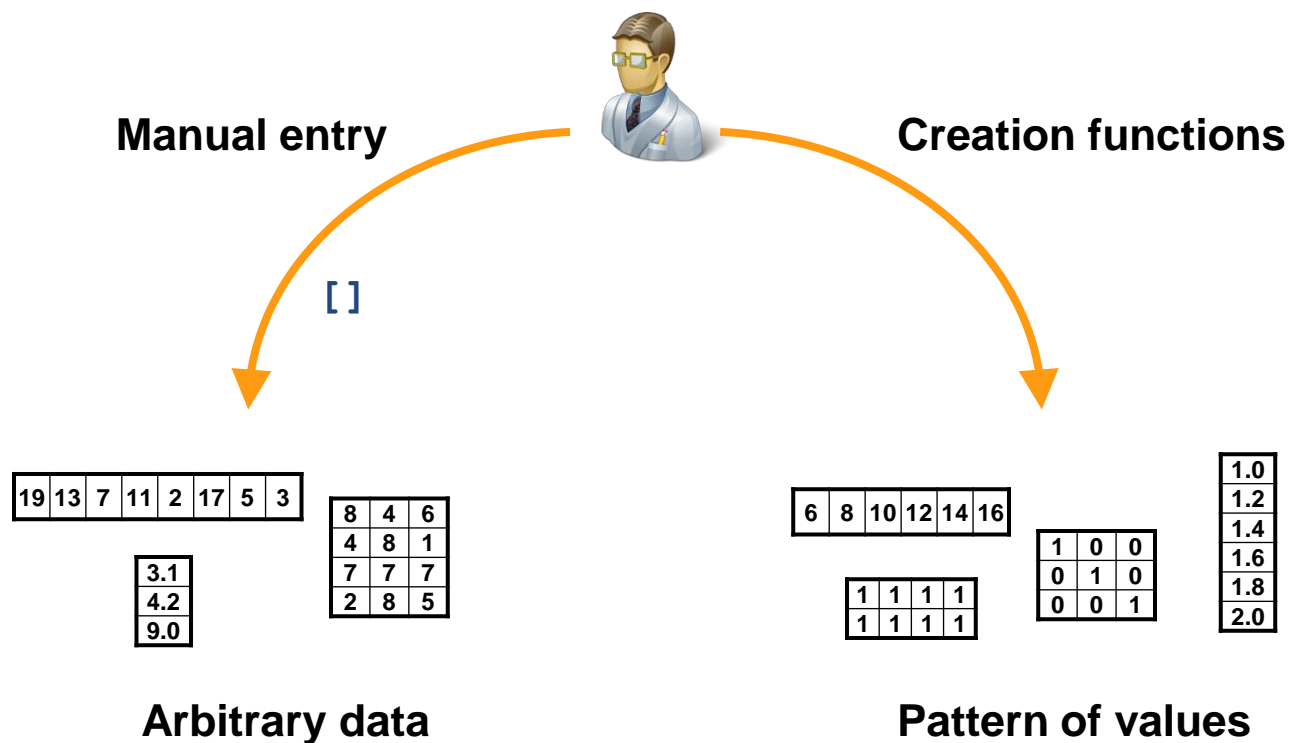
- $0.1 \times \frac{2}{5} + 8^7$
- $\sin(0.8\pi)$
- $10^{\ln(13) \times 6}$
- Find  $e^x$ ,  $x = \frac{\sqrt{(5+8)}}{49}$

# Course Outline

- Working with the MATLAB User Interface
- Variables and Commands
- Array Creation and Analysis
- Working with Data Files
- Visualization with Array
- Automating Commands with Scripts
- Appendix: Data Type




# Creating Array



# Entering Vectors Manually

```
>> x = [19 13 7 11 2 17 5 3]
```



19	13	7	11	2	17	5	3
----	----	---	----	---	----	---	---

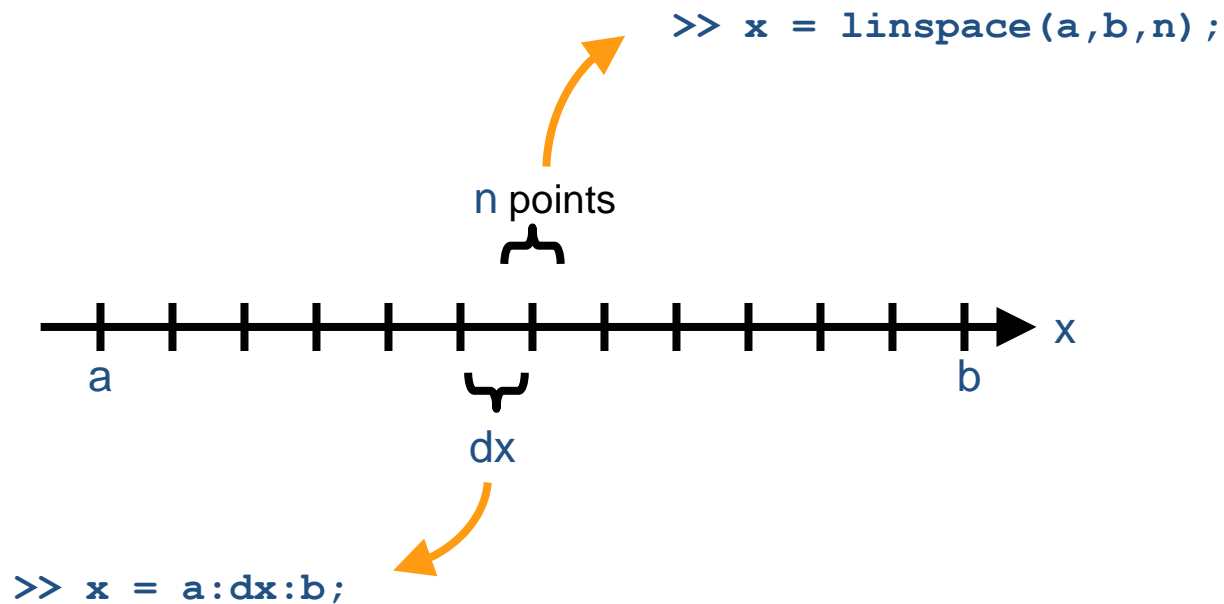
```
>> x = [19;13;7;11;2;17;5;3]
```



19
13
7
11
2
17
5
3



## Creating Vectors of Equally Spaced Values





# Entering Matrices Manually

```
>> A = [1,2,3; 4,5,6; 7,8,9];
```

or

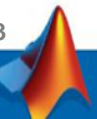
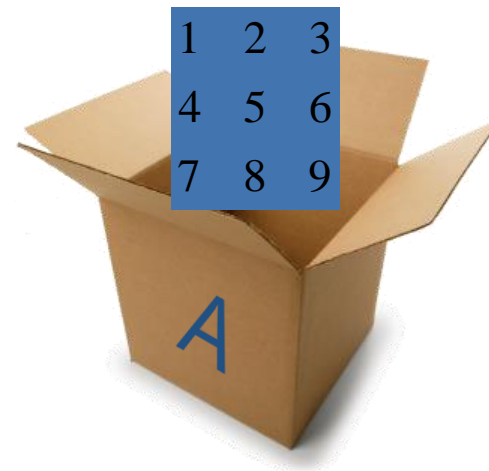
```
>> A = [1 2 3; 4 5 6; 7 8 9];
```

or

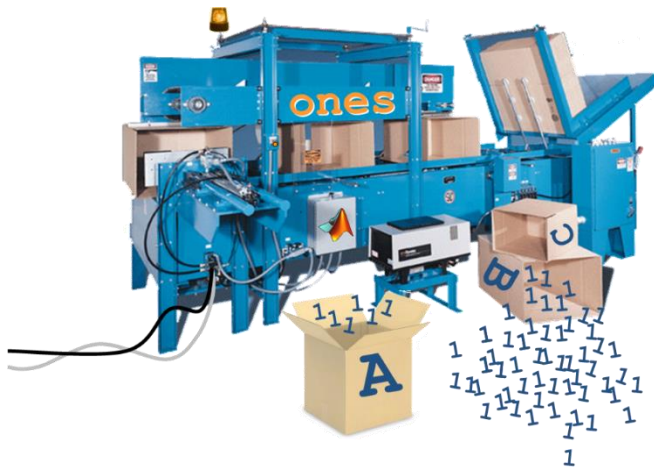
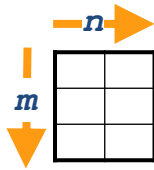
```
>> A = [1 2 3  
4 5 6  
7 8 9]
```

}

data entry  
mode



# Creating Matrices



```
>> A = fun(m,n) ;
```

compan	pascal
eye	rand
gallery	randi
hadamard	randn
hankel	rosser
hilb	toeplitz
invhilb	vander
magic	wilkinson
ones	zeros

## Concatenating Arrays

A

2	3
5	7
11	13

B

-1	1
1	-1

C

0
8
0

>> X = [A;B]

2	3
5	7
11	13
-1	1
1	-1

>> Y = [A,C]

2	3	0
5	7	8
11	13	0

**!!!Please notice the dimension**



## Row, Column Indexing

```
>> gpdata(1,2)
```

```
>> gpdata(3,6)
```

```
>> gpdata(2,end)
```

	1	2	3	4	5	6	7	8	9	10	11	12
1	1990	NaN	1.8700	3.6300	2.6500	4.5900	3.1600		2.0500	2.8200	1.1600	
2	1991	1.9600	1.9200	3.4500	2.9000	4.5000	3.4600	1.3000	2.4900	3.0100	1.1400	
3	1992	1.8900	1.7300	3.5600	3.2700	4.5300	3.5800	1.5000	2.6500	3.0600	1.1300	
4	1993	1.7300	1.5700	3.4100	3.0700	3.6800	4.1600	1.5600	2.8800	2.8400	1.1100	
5	1994	1.8400	1.4500	3.5900	3.5200	3.7000	4.3600	1.4800	2.8700	2.9900	1.1100	
6	1995	1.9500	1.5300	4.2600	3.9600	4	4.4300	1.1100	2.9400	3.2100	1.1500	
7	1996	2.1200	1.6100	4.4100	3.9400	4.3900	3.6400	1.2500	3.1800	3.3400	1.2300	
8	1997	2.0500	1.6200	4	3.5300	4.0700	3.2600	1.4700	3.3400	3.8300	1.2300	
9	1998	1.6300	1.3800	3.8700	3.3400	3.8400	2.8200	1.4900	3.0400	4.0600	1.0600	
10	1999	1.7200	1.5200	3.8500	3.4200	3.8700	3.2700	1.7900	3.8000	4.2900	1.1700	
11	2000	1.9400	1.8600	3.8000	3.4500	3.7700	3.6500	2.0100	4.1800	4.5800	1.5100	
12	2001	1.7100	1.7200	3.5100	3.4000	3.5700	3.2700	2.2000	3.7600	4.1300	1.4600	
13	2002	1.7600	1.6900	3.6200	3.6700	3.7400	3.1500	2.2400	3.8400	4.1600	1.3600	
14	2003	2.1900	1.9900	4.3500	4.5900	4.5300	3.4700	2.0400	4.1100	4.7000	1.5900	
15	2004	2.7200	2.3700	4.9900	5.2400	5.2900	3.9300	2.0300	4.5100	5.5600	1.8800	
16	2005	3.2300	2.8900	5.4600	5.6600	5.7400	4.2800	2.2200	5.2800	5.9700	2.3000	
17	2006	3.5400	3.2600	5.8800	6.0300	6.1000	4.4700	2.3100	5.9200	6.3600	2.5900	
18	2007	3.8500	3.5900	6.6000	6.8800	6.7300	4.4900	2.4000	6.2100	7.1300	2.8000	
19	2008	4.4500	4.0800	7.5100	7.7500	7.6300	5.7400	2.4500	5.8300	7.4200	3.2700	

```
>> gpdata(end,2)
```

```
>> gpdata(end,end)
```

# Multiple Row, Column Indices

```
>> Year = gpdata(:,1)
```

```
>> gpdata([3,4],6:9)
```

	1	2	3	4	5	6	7	8	9	10	11	12
1	1990	NaN	1.8700	3.6300	2.6500	4.5900	3.1600		1	2.0500	2.8200	1.1600
2	1991	1.9600	1.9200	3.4500	2.9000	4.5000	3.4600	1.3000	2.4900	3.0100	1.1400	
3	1992	1.8900	1.7300	3.5600	3.2700	4.5300	3.5800	1.5000	2.6500	3.0600	1.1300	
4	1993	1.7300	1.5700	3.4100	3.0700	3.6800	4.1600	1.5600	2.8800	2.8400	1.1100	
5	1994	1.8400	1.4500	3.5900	3.5200	3.7000	4.3600	1.4800	2.8700	2.9900	1.1100	
6	1995	1.9500	1.5300	4.2600	3.9600	4	4.4300	1.1100	2.9400	3.2100	1.1500	
7	1996	2.1200	1.6100	4.4100	3.9400	4.3900	3.6400	1.2500	3.1800	3.3400	1.2300	
8	1997	2.0500	1.6200	4	3.5300	4.0700	3.2600	1.4700	3.3400	3.8300	1.2300	
9	1998	1.6300	1.3800	3.8700	3.3400	3.8400	2.8200	1.4900	3.0400	4.0600	1.0600	
10	1999	1.7200	1.5200	3.8500	3.4200	3.8700	3.2700	1.7900	3.8000	4.2900	1.1700	
11	2000	1.9400	1.8600	3.8000	3.4500	3.7700	3.6500	2.0100	4.1800	4.5800	1.5100	
12	2001	1.7100	1.7200	3.5100	3.4000	3.5700	3.2700	2.2000	3.7600	4.1300	1.4600	
13	2002	1.7600	1.6900	3.6200	3.6700	3.7400	3.1500	2.2400	3.8400	4.1600	1.3600	
14	2003	2.1900	1.9900	4.3500	4.5900	4.5300	3.4700	2.0400	4.1100	4.7000	1.5900	
15	2004	2.7200	2.3700	4.9900	5.2400	5.2900	3.9300	2.0300	4.5100	5.5600	1.8800	
16	2005	3.2300	2.8900	5.4600	5.6600	5.7400	4.2800	2.2200	5.2800	5.9700	2.3000	
17	2006	3.5400	3.2600	5.8800	6.0300	6.1000	4.4700	2.3100	5.9200	6.3600	2.5900	
18	2007	3.8500	3.5900	6.6000	6.8800	6.7300	4.4900	2.4000	6.2100	7.1300	2.8000	
19	2008	4.4500	4.0800	7.5100	7.7500	7.6300	5.7400	2.4500	5.8300	7.4200	3.2700	

```
>> gp08 = gpdata(end,2:end)
```

# Linear Indexing

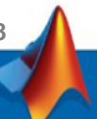
```
>> A = magic(5)
```

```
A =
```

1	17	6	24	11	1	16	8	21	15
2	23	7	5	12	7	17	14	22	16
3	4	8	6	13	13	18	20	23	22
4	10	9	12	14	19	19	21	24	3
5	11	10	18	15	25	20	2	25	9

Indices Data

↑  
end



# Exercise

◆ `A = magic(5)`

◆ `a=A(?)`

◆ `b=A(?)`

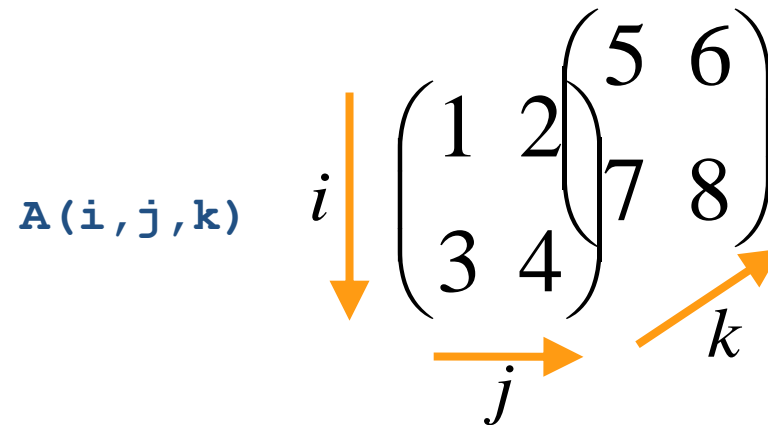
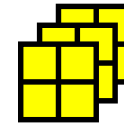
◆ `c=A(?)`

**A =**

	17	24	1	8	15
	23	5	7	14	16
c	4	6	13	20	22
	10	12	19	21	3
b	11	18	25	2	9

Diagram illustrating a 5x5 magic square matrix A. The matrix is shown with its elements. Three specific elements are highlighted with arrows and labels: 'a' points to the element 24 in the first row, second column; 'b' points to the element 11 in the fifth row, first column; and 'c' points to the element 4 in the third row, first column. The elements 4, 6, 13, 20, 22, 12, 19, 21, and 3 are also highlighted with a blue border.

# Multidimensional Arrays



**Construct**

```
>> A(:, :, 1) = [1, 2; 3, 4];  
>> A(:, :, 2) = [5, 6; 7, 8];  
>> B = rand(2, 2, 3);
```

**Access**

```
>> A(:, 2, 1)  
ans =  
2  
4
```





## Chapter 3-1 Test Your Knowledge

1. (Select all that apply) Which of the following will create a matrix with three rows?
  - A. `A = [zeros(2,4);ones(1,4)] ;`
  - B. `A = [1;2;3,4;5;6] ;`
  - C. `A = [1,2;3,4;5,6] ' ;`
  - D. `A = rand(3) ;`
2. Given a 5-by-5 matrix A, `A(4:end,3:4)` will produce a matrix of what size?
  - A. 1-by-2
  - B. 2-by-2
  - C. 2-by-3
  - D. 3-by-2
3. (Select all that apply) Which commands are equivalent to the command  
`>> x = 1.4:2:6.8 ; ?`
  - A. `x = [1.4 2 6.8] ;`
  - B. `x = [1.4 6.8] ;`
  - C. `x = [1.4 3.4 5.4] ;`
  - D. `x = [1.4 3.4 5.4 6.8] ;`
  - E. `x = [3.4 5.4] ;`



# Array Operations

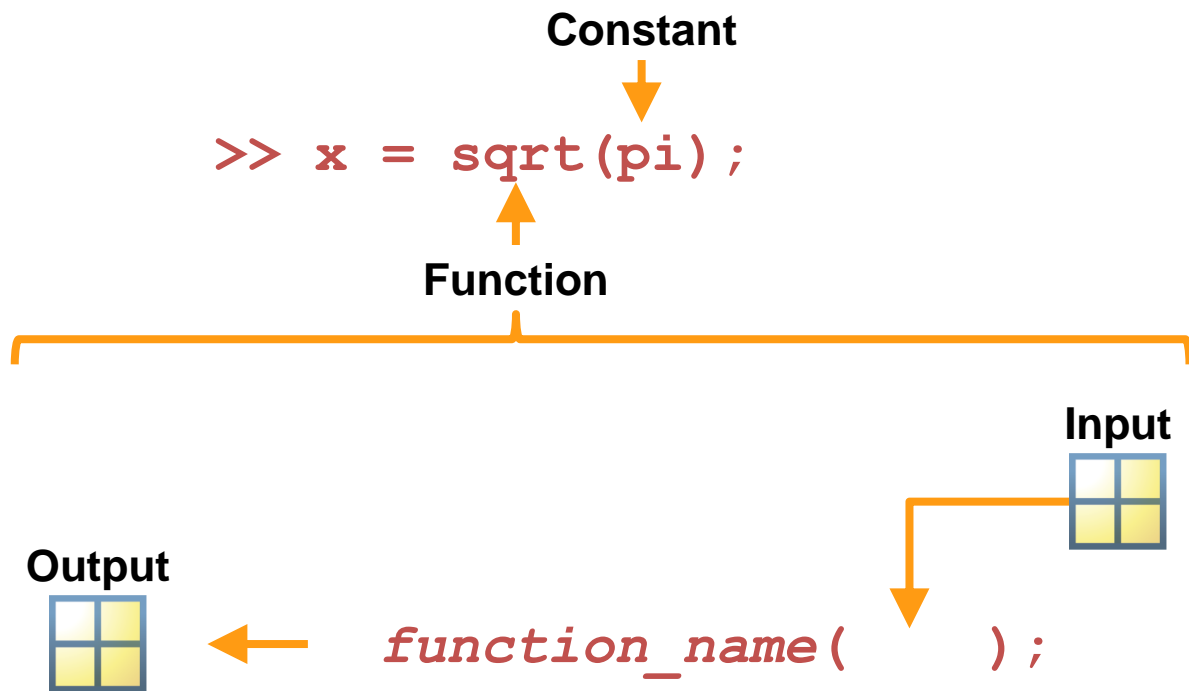
```
>> GMSum = Germany + Mexico
```

3.65		2.65		1.00
4.20		2.90		1.30
4.77		3.27		1.50
4.63		3.07		1.56
5.00		3.52		1.48
5.07		3.96		1.11
5.19		3.94		1.25
5.00		3.53		1.47
4.83	=	3.34		1.49
5.21		3.42		1.79
5.46		3.45		2.01
5.60		3.40		2.20
5.91		3.67		2.24
6.63		4.59		2.04
7.27		5.24		2.03
7.88		5.66		2.22
8.34		6.03		2.31
9.28		6.88		2.40
10.20		7.75		2.45

```
>> load gasprices
```



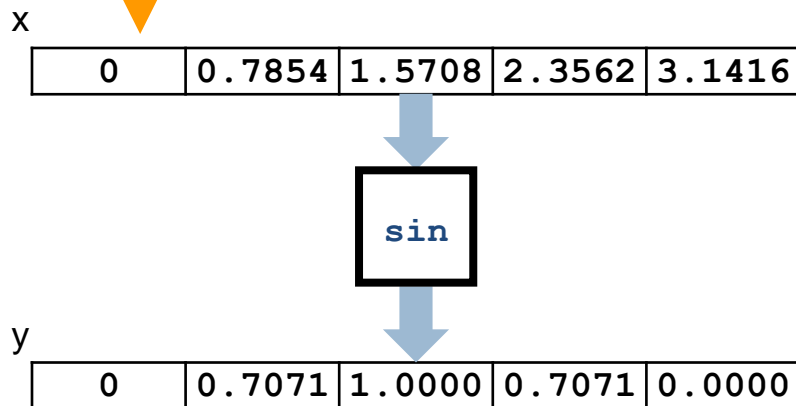
# Using Built-In Functions and Constants



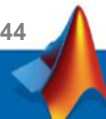
# Mathematical Functions

```
>> y = sin(x);
```

**“Vectorized”**

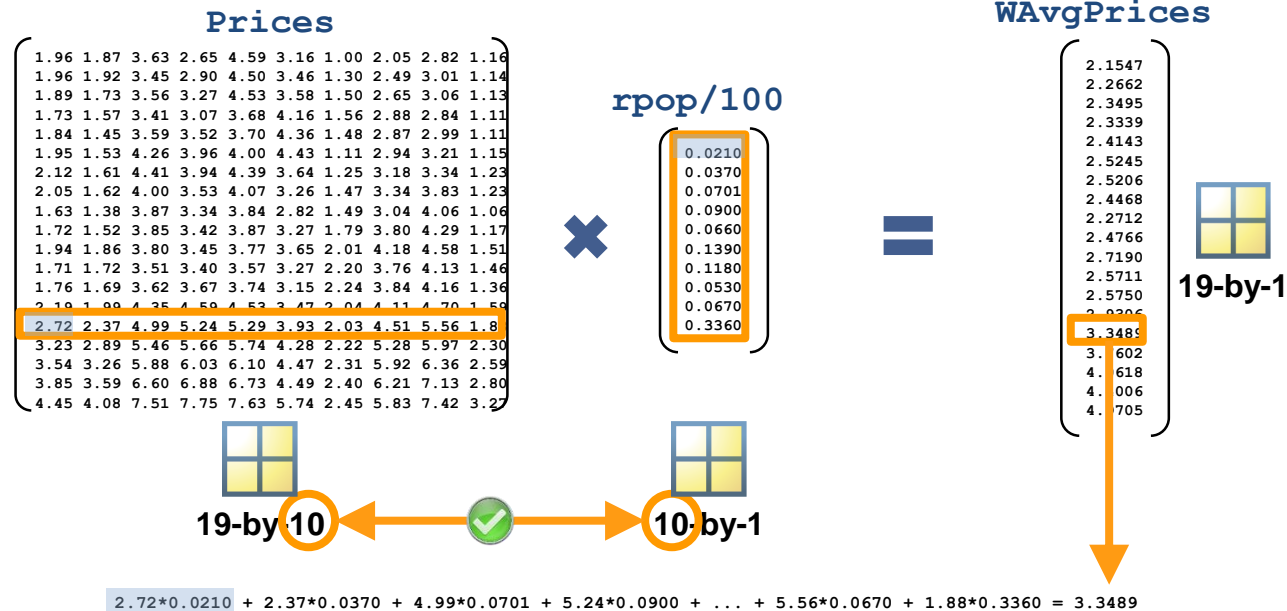


`sin`  
`sind`  
`sinh`  
`asin`  
`exp`  
`log`  
`log2`  
`log10`  
`sqrt`  
`nthroot`  
`abs`  
`angle`  
`floor`  
`ceil`  
`round`  
`mod`

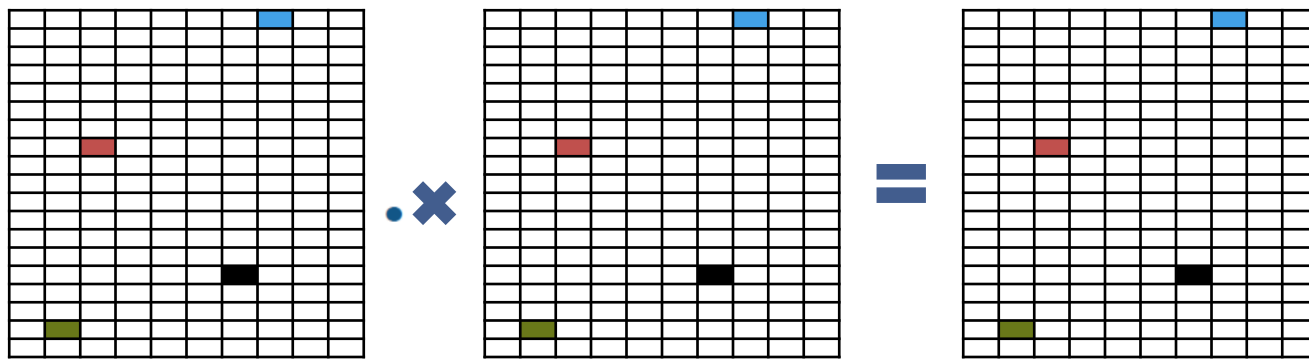


# Matrix Operations

Inner dimensions must be equal!!



# Array Operations



Prices



cx

  
19-by-10

  
19-by-10



  
19-by-10



## Array Operations (Cont.)

Matrices  
must have  
the same  
dimensions!!

```
>> GMRatio = Germany ./ Mexico
```

2.65		2.65		1.00
2.23		2.90		1.30
2.18		3.27		1.50
1.97		3.07		1.56
2.38		3.52		1.48
3.57		3.96		1.11
3.15		3.94		1.25
2.40		3.53		1.47
2.24	=	3.34	÷	1.49
1.91		3.42		1.79
1.72		3.45		2.01
1.55		3.40		2.20
1.64		3.67		2.24
2.25		4.59		2.04
2.58		5.24		2.03
2.55		5.66		2.22
2.61		6.03		2.31
2.87		6.88		2.40
3.16		7.75		2.45

# Data in the MATLAB® Environment

← Variables →

Observations or cases ↑ ↓

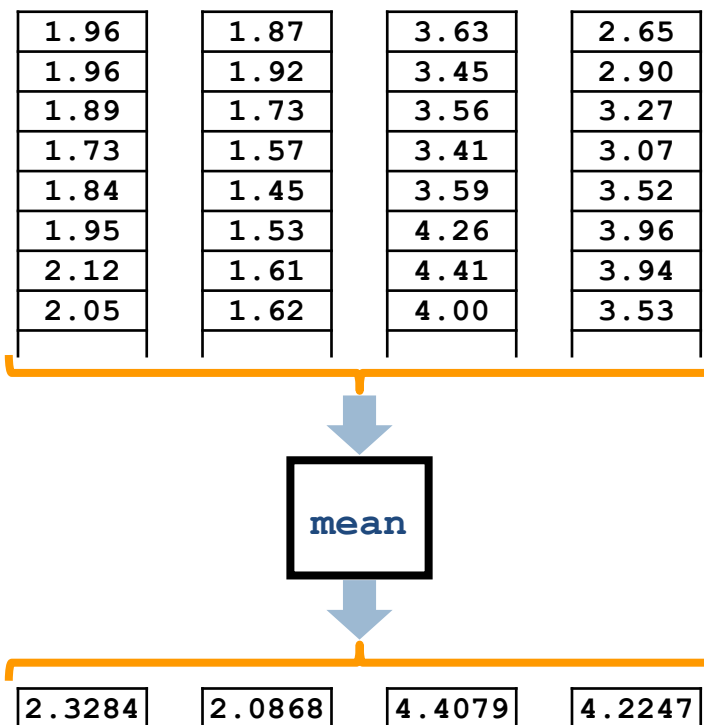
	Australia	Canada	France	Germany	...
1990	1.96	1.87	3.63	2.65	
1991	1.96	1.92	3.45	2.90	
1992	1.89	1.73	3.56	3.27	
1993	1.73	1.57	3.41	3.07	
1994	1.84	1.45	3.59	3.52	
1995	1.95	1.53	4.26	3.96	
1996	2.12	1.61	4.41	3.94	
1997	2.05	1.62	4.00	3.53	

↑ ↓

Independent columns → mean



# Statistical Operations



max  
min  
mean  
median  
std  
sum  
prod  
diff  
gradient  
cumsum  
cumprod  
corrcoef  
cov

# Example: Array Operations

- In most languages - use loops:

```
>> tic; for I = 1:10000
Density(I) = Mass(I) / (Length(I)*Width(I)*Height(I));
end; toc
elapsed_time =
    4.7260
```

Use **TIC** and **TOC** to  
measure elapsed time

- In MATLAB - use array operations:

```
>> tic; Density = Mass./(Length.*Width.*Height); toc
elapsed_time =
    0
```

Vectorized code is  
much faster than loops

```
>> array_examp
```



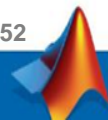
# Chapter 3-2 Test Your Knowledge

1. (Select all that apply) Given two matrices, **A** and **B**, where **A** is a 2-by-3 matrix and **B** is a 3-by-2 matrix, which of the following operations are valid?
  - A. **A+B**
  - B. **A.+B**
  - C. **A\*B**
  - D. **A.\*B**
  
2. Given a vector **x**, what is the command to add 3 to each element, double that value, then sum all the resulting values?
  - A. **sum(2\*x+3)**
  - B. **sum(2\*[x(k)+3])**
  - C. **sum[2\*x+3]**
  - D. **sum(2\*(x+3))**

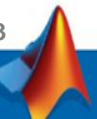
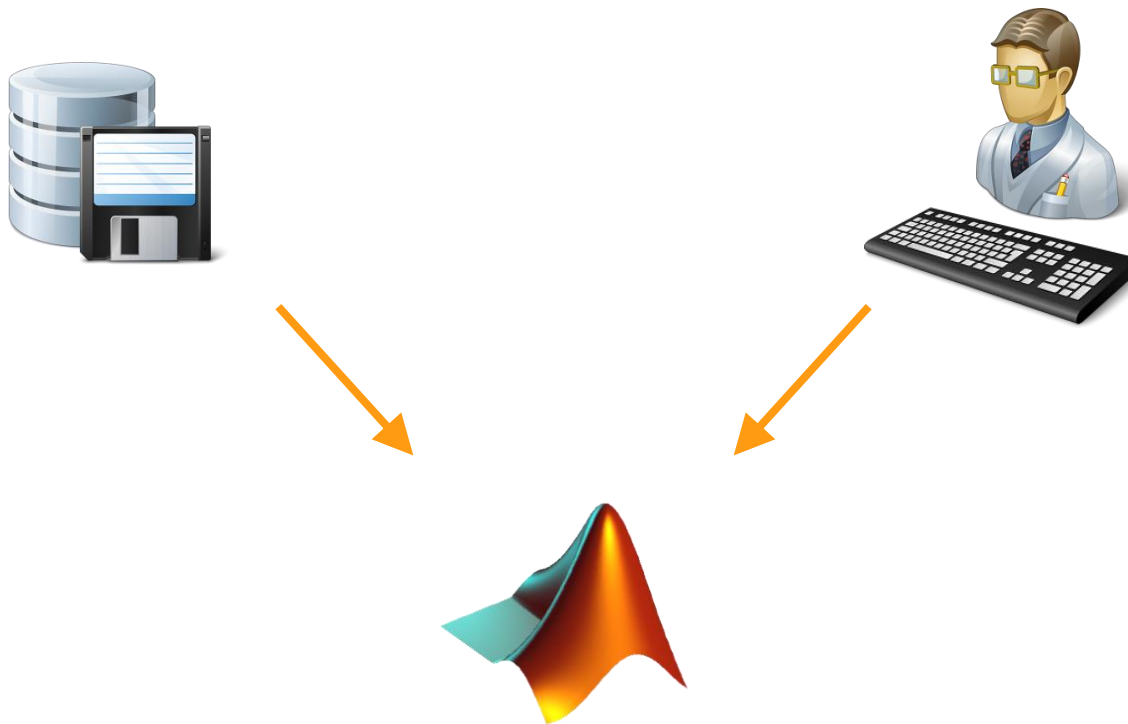


# Course Outline

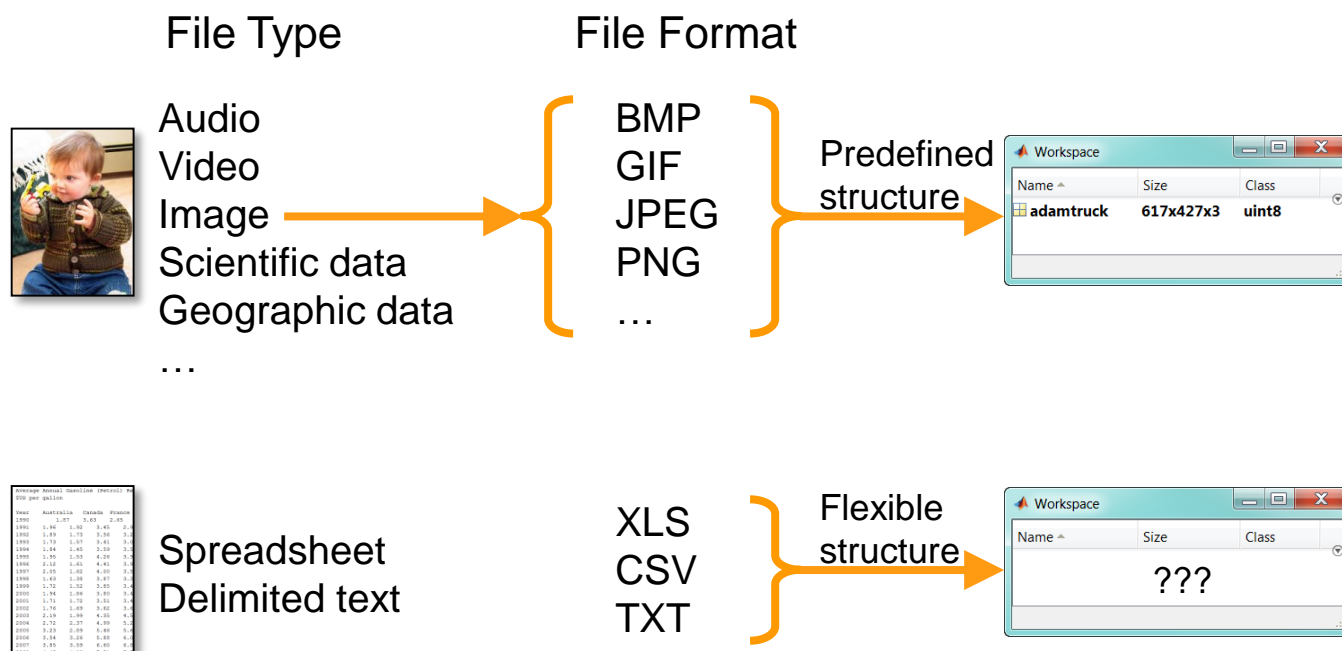
- Working with the MATLAB User Interface
- Variables and Commands
- Array Creation and Analysis
- Working with Data Files
- Visualization with Array
- Automating Commands with Scripts
- Appendix: Data Type



# Getting Data into MATLAB®



# File Types and File Formats



# Reading and Writing Fixed-Structure Files



→ `audioread`

`cdfread`

1.20E+01, 9.38E-07, 4.  
1.30E+01, 6.11E-08, 9.  
1.40E+01, 4.58E-07, 6.  
1.50E+01, 6.89E-07, 3.  
1.60E+01, 1.66E-07, 2.  
1.70E+01, 2.13E-07, 2.  
1.80E+01, 2.38E-07, 3.  
1.90E+01, 4.13E-07, 9.  
2.00E+01, 8.23E-07, 7.  
2.10E+01, 6.60E-07, 4.  
2.20E+01, 9.11E-07, 8.  
2.30E+01, 7.77E-07, 1.

→ `csvread`

`dlmread`

`h5read`



→ `imread`

`ncread`



→ `urlread`

`xmlread`

...

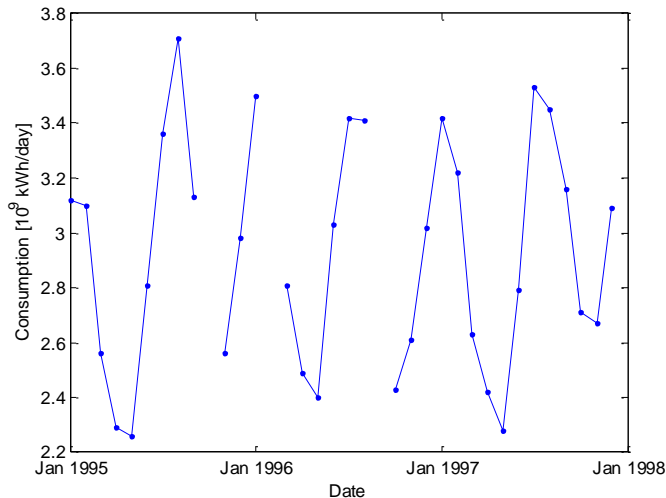


```
data = xyzread('file.xyz')
```

```
xyzwrite(data, 'file.xyz')
```



# Dealing with Missing Data



```
>> load electricityNaNs
>> x = Residential(215:220);
>> mean(x)
>> nnz(isnan(x))
```

mean

```
3.190
0
4.030
0
4.380
0
```



ans =  
NaN

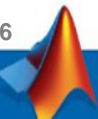
```
3.190
0
4.030
NaN
4.380
0
```

```
>> nanmean(x)
```

```
3.190
0
4.030
0
4.380
0
```

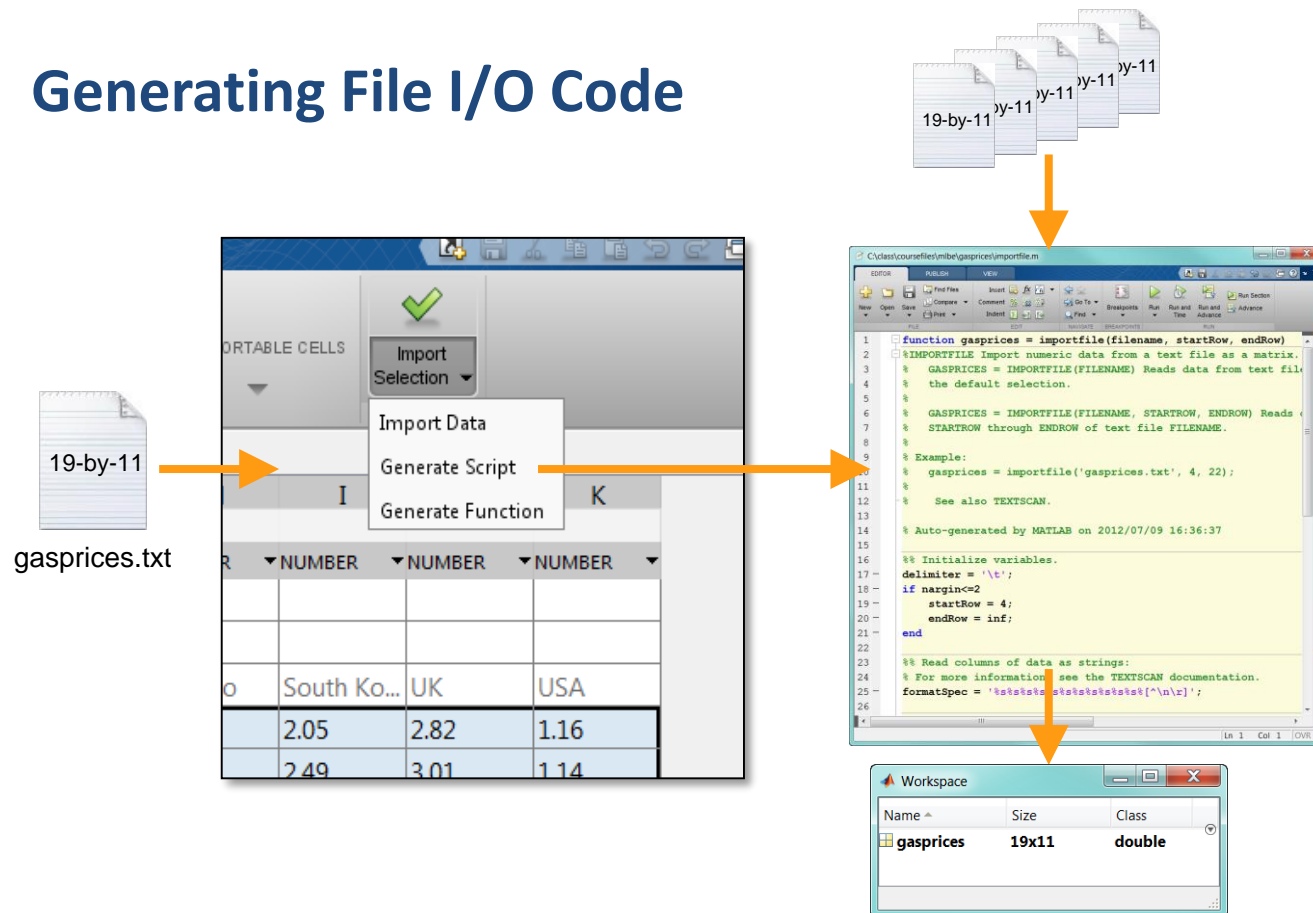
```
>> x(isnan(x)) =
[];
>> mean(x)
```

```
NaN
3.430
0
3.190
3.050
0
4.030
0
4.380
0
3.880
8
3.430
3.8808;
>> mean(x)
```

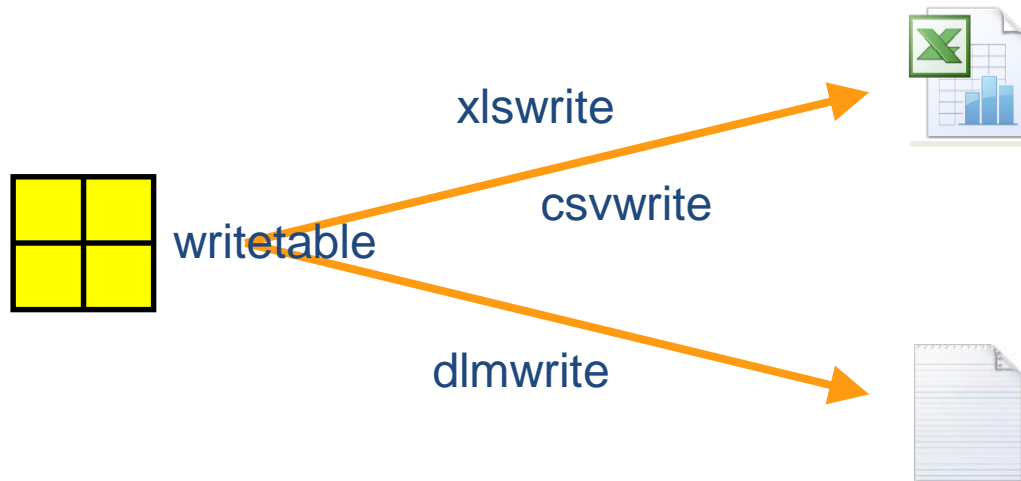




# Generating File I/O Code



# Exporting Spreadsheets



# Chapter 4 Test Your Knowledge

1. (Select all that apply) Which of the following can you change when importing a delimited text file with the Import Tool?
  - A. Which columns to import.
  - B. The format (text, number, date, etc.) of each column.
  - C. The delimiter character.
  - D. How to import missing values.
  
2. Suppose “ABC” is a supported format in MATLAB. Which command would import data from an ABC file mydata.abc into a matrix A?
  - A. `mydata = abcread(A) ;`
  - B. `abcread(A,mydata) ;`
  - C. `abcread('mydata.abc',A) ;`
  - D. `A = abcread('mydata.abc') ;`
  - E. `A = abcread(mydata) ;`



# Course Outline

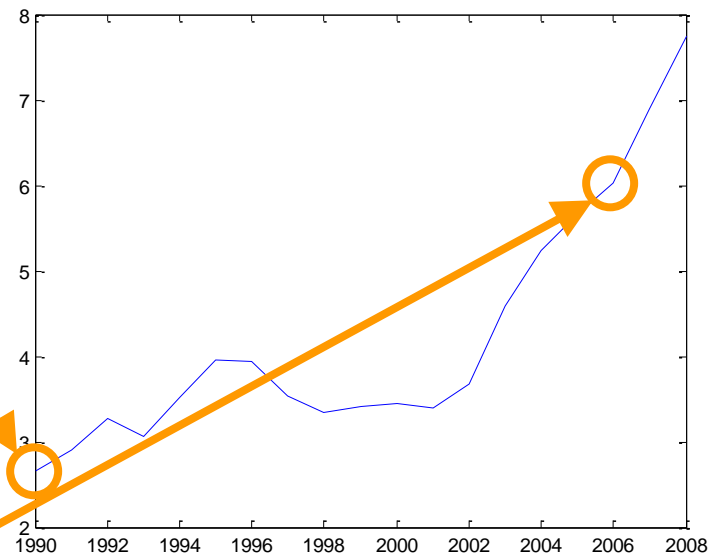
- Working with the MATLAB User Interface
- Variables and Commands
- Array Creation and Analysis
- Working with Data Files
- Visualization with Array
- Automating Commands with Scripts
- Appendix: Data Type



# Plotting Vectors

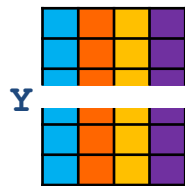
```
>> plot(Year, Germany)
```

1990	2.65
1991	2.90
1992	3.27
1993	3.07
1994	3.52
1995	3.96
1996	3.94
1997	3.53
1998	3.34
1999	3.42
2000	3.45
2001	3.40
2002	3.67
2003	4.59
2004	5.24
2005	5.66
2006	6.03
2007	6.88
2008	7.75

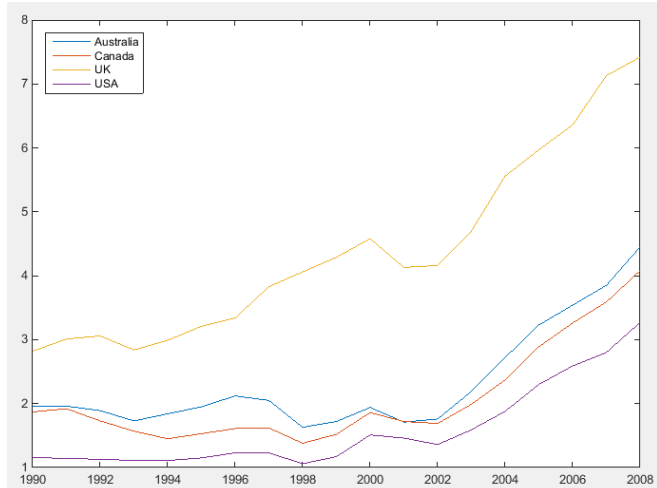


## Plotting Multiple Columns

```
>> EndIdx = [1 2 9 10];  
>> Y = Prices(:, EndIdx);  
>> plot(Year, Y)  
>> legend(country(EndIdx), 'Location', 'NW')
```

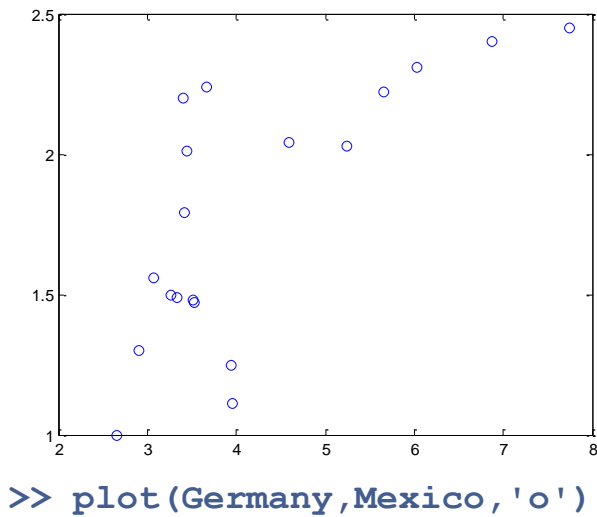


4 plots

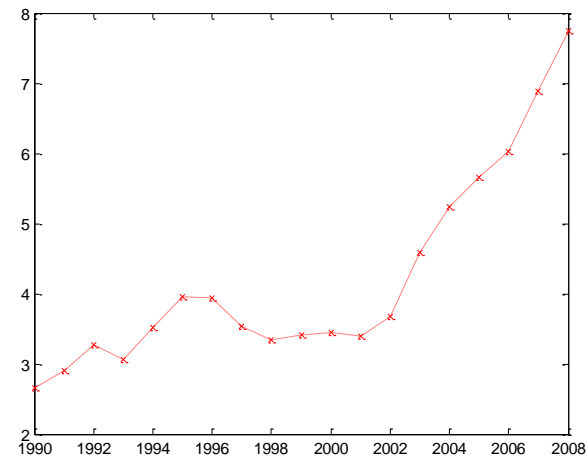


# Plot Options

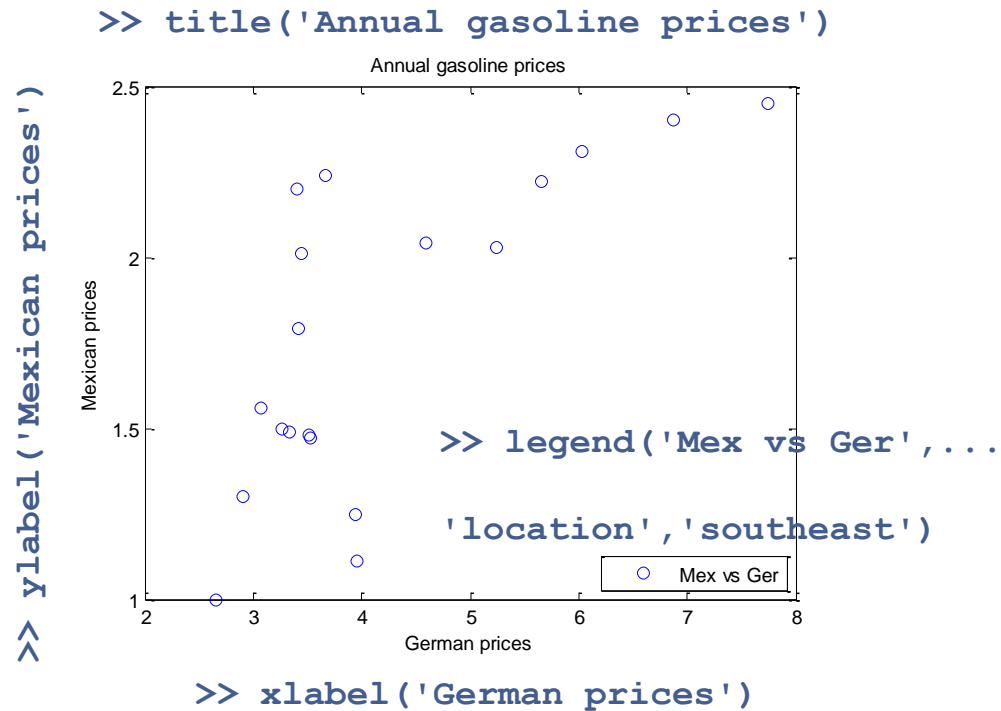
Data Type



```
>> plot(Year,Germany,'rx:')
```

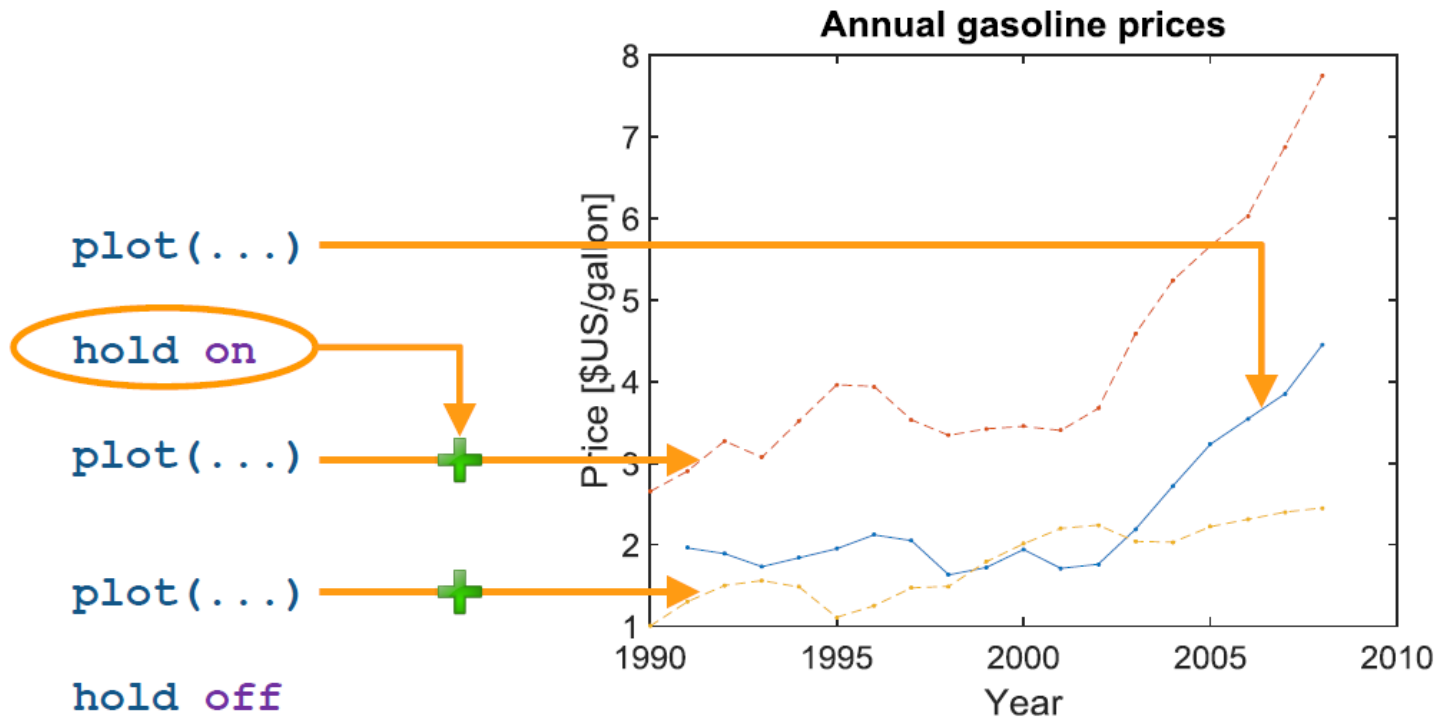


# Annotating Plots



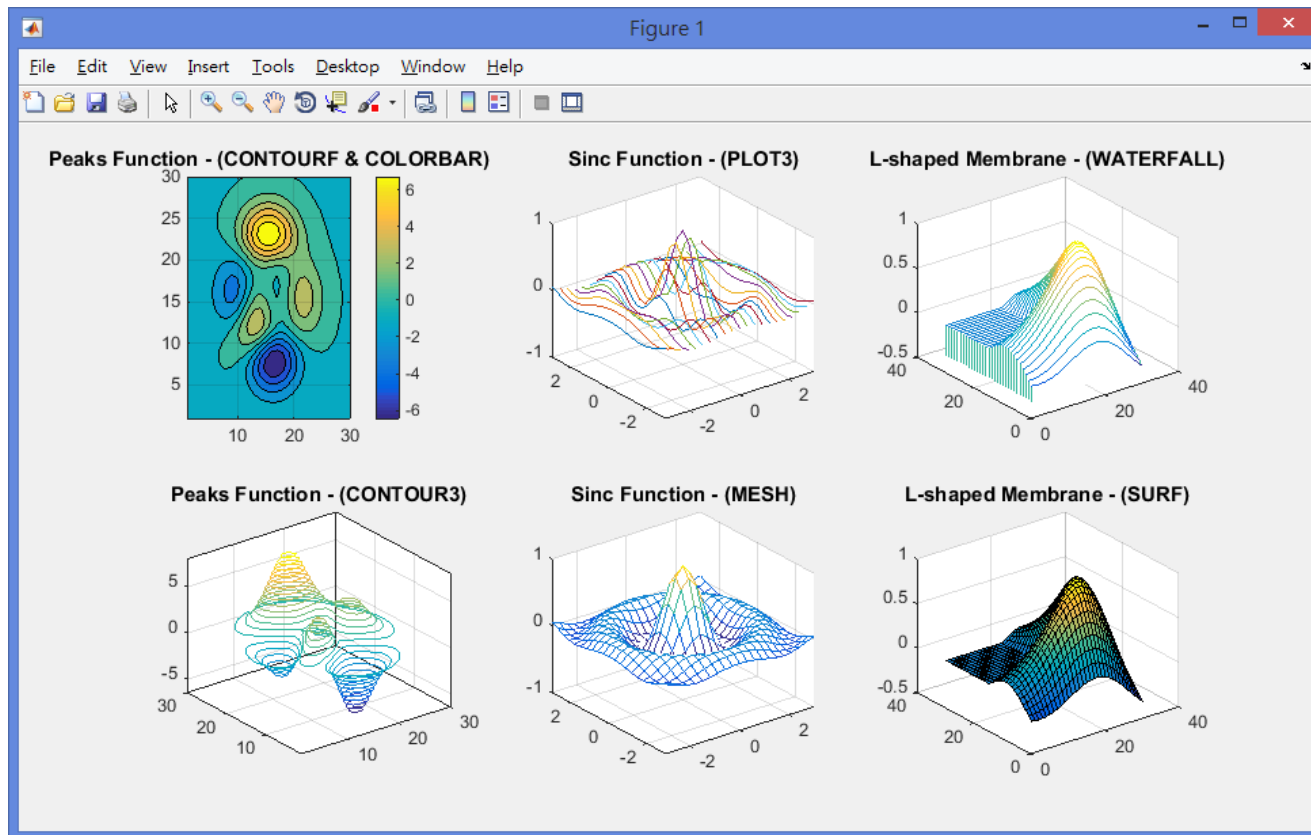


# Adding Plots



```
>> edit multipleLXplot
```

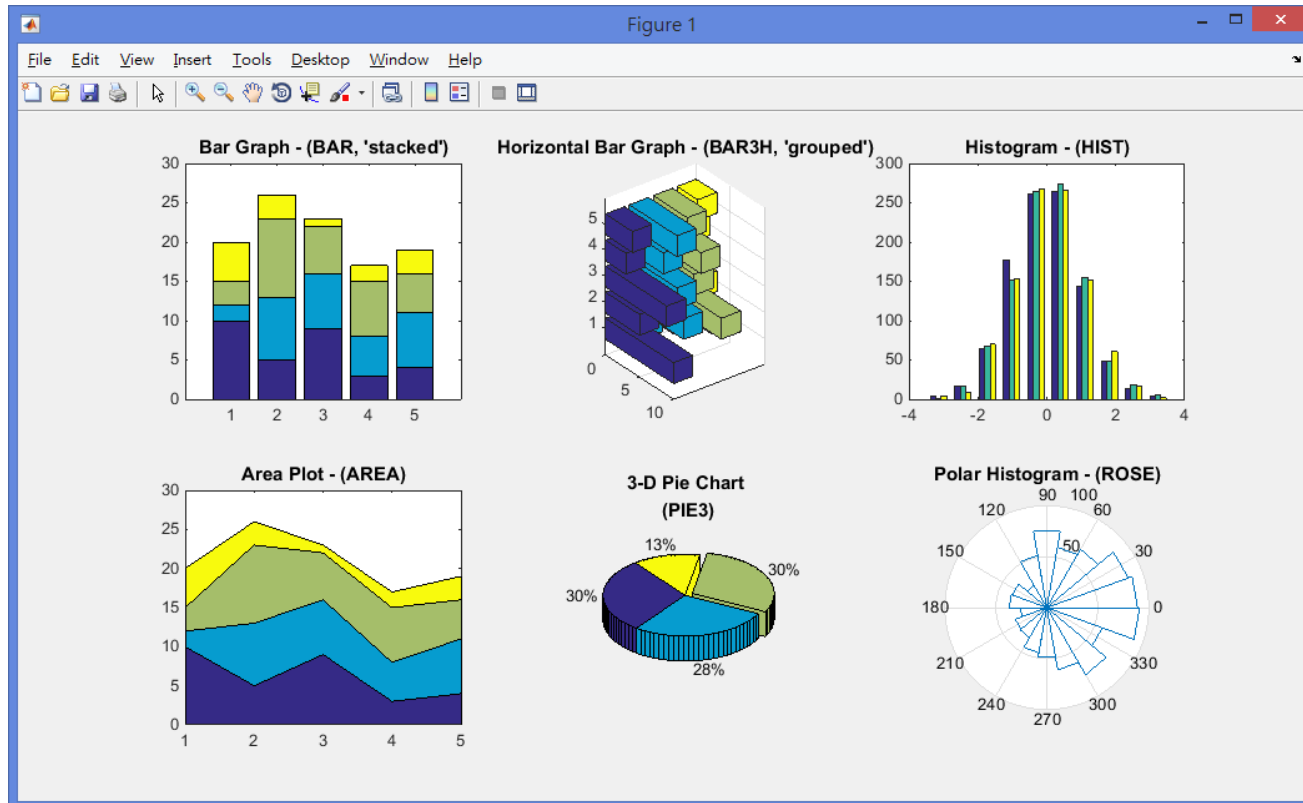
## 3-D Surface Plotting



```
>> surf_3d
```

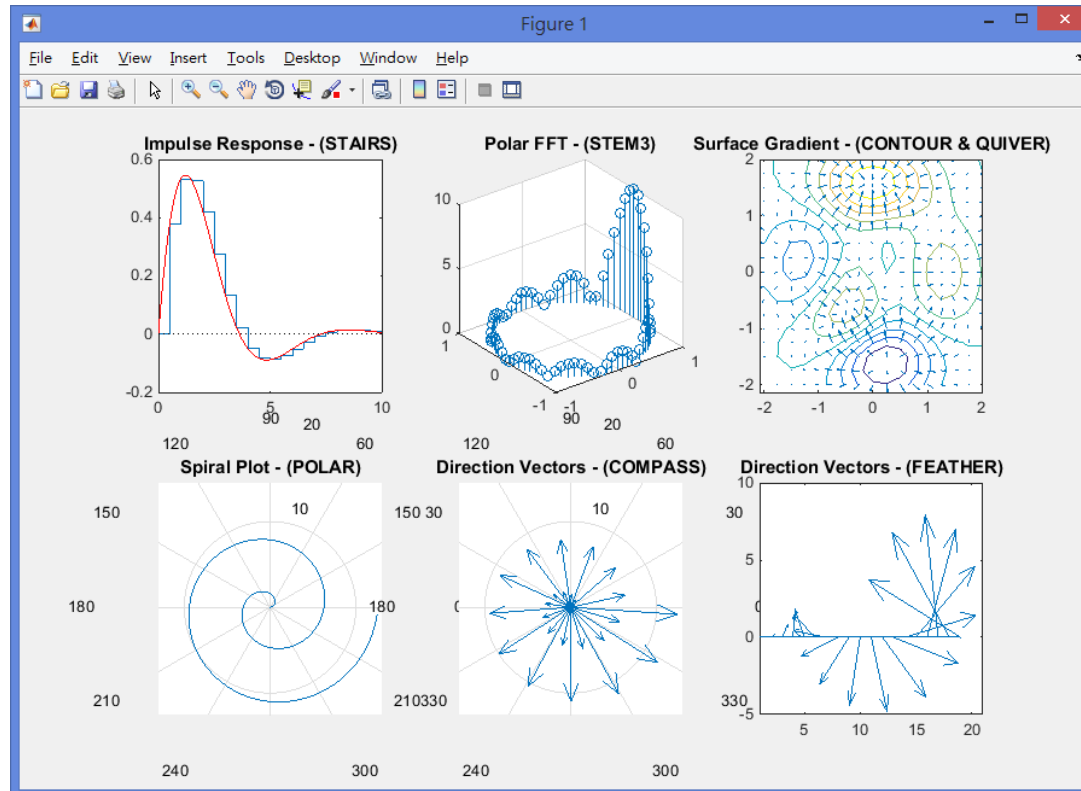


# Specialized Plotting Routines



>> spec\_plots

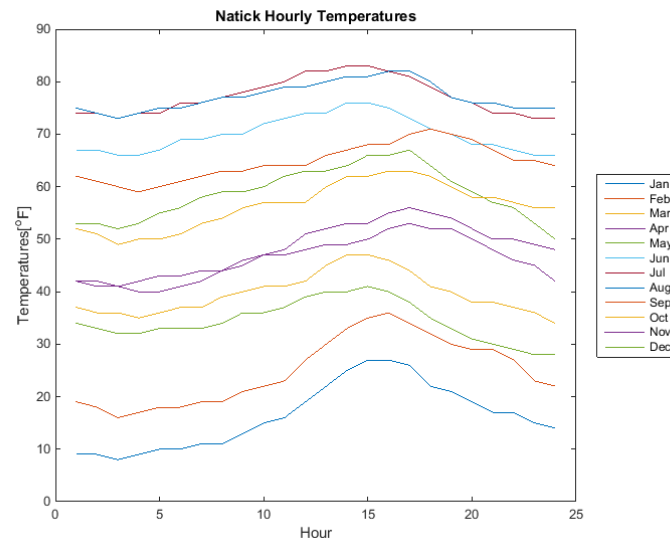
# Specialized Plotting Routines (Continued)



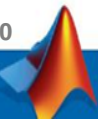
```
>> spec_plots2
```

## Exercise: Natick Hourly Temperatures

1. Load `natickData.mat`.
2. Calculate the average temperature for each month.
3. Determine which month had the largest standard deviation.
4. Determine which month had the largest range (max – min).
5. Plot temperature vs. time for each month.



```
>> edit  
natickTemps
```



## Chapter 5 Test Your Knowledge

1. (Select all that apply) Which of the following will create a scatter plot of `frogs` on the horizontal axis and `GDP` on the vertical axis, with red markers at the data points?
  - A. `plot(GDP,frogs,'ro')`
  - B. `plot(GDP,frogs,'o','r')`
  - C. `plot(frogs,GDP,'ro')`
  - D. `plot(frogs,GDP,'red')`
2. If `A` is a 15-by-7 matrix, which of the following commands will result in five line plots on the same axes?
  - A. `plot(A(:))`
  - B. `plot(A(:,3:end))`
  - C. `plot(A(11:end,:))`
  - D. `plot(A(2:6))`



# Course Outline

- Working with the MATLAB User Interface
- Variables and Commands
- Array Creation and Analysis
- Working with Data Files
- Visualization with Array
- Automating Commands with Scripts
- Appendix: Data Type



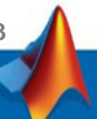
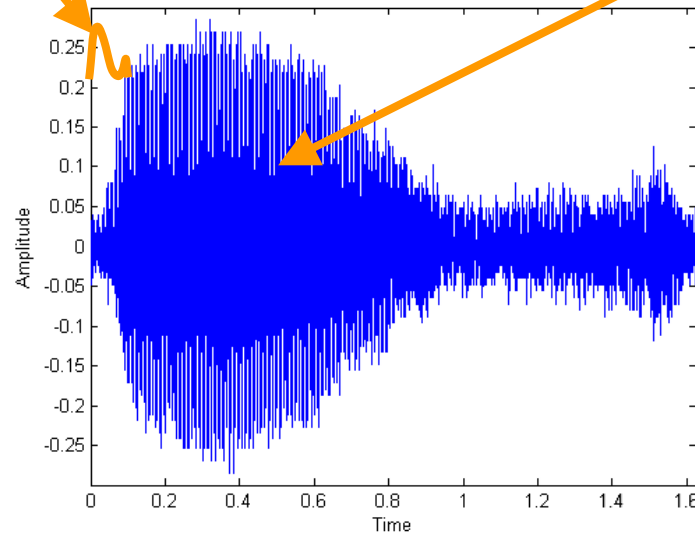
# Example: Modeling a Whale Call



Amplitude modulated:  $y(t) = A(t)y_0(t)$

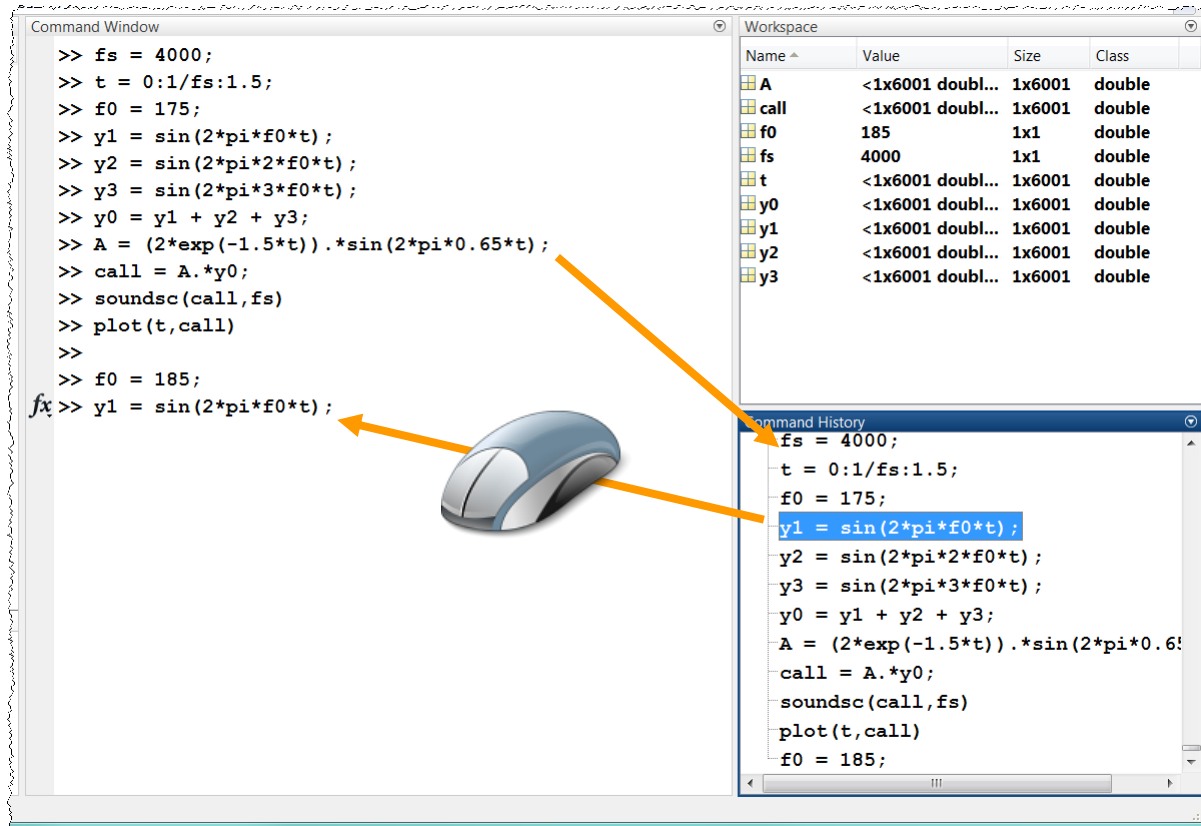
Decaying oscillation  
 $A(t) = A_0 e^{-Bt} \sin(2\pi f_m t)$

Sum of harmonics of a  
fundamental frequency  
 $y_0(t) = \sum_n \sin(2\pi n f_0 t)$





# The Command History



The image shows the MATLAB Command Window, Workspace, and Command History windows. The Command Window contains the following code:

```
>> fs = 4000;  
>> t = 0:1/fs:1.5;  
>> f0 = 175;  
>> y1 = sin(2*pi*f0*t);  
>> y2 = sin(2*pi*2*f0*t);  
>> y3 = sin(2*pi*3*f0*t);  
>> y0 = y1 + y2 + y3;  
>> A = (2*exp(-1.5*t)) .* sin(2*pi*0.65*t);  
>> call = A.*y0;  
>> soundsc(call,fs)  
>> plot(t,call)  
>>  
>> f0 = 185;  
fx>> y1 = sin(2*pi*f0*t);
```

The Workspace window shows the following variables:

Name	Value	Size	Class
A	<1x6001 doubl...	1x6001	double
call	<1x6001 doubl...	1x6001	double
f0	185	1x1	double
fs	4000	1x1	double
t	<1x6001 doubl...	1x6001	double
y0	<1x6001 doubl...	1x6001	double
y1	<1x6001 doubl...	1x6001	double
y2	<1x6001 doubl...	1x6001	double
y3	<1x6001 doubl...	1x6001	double

The Command History window shows the following commands:

```
fs = 4000;  
t = 0:1/fs:1.5;  
f0 = 175;  
y1 = sin(2*pi*f0*t);  
y2 = sin(2*pi*2*f0*t);  
y3 = sin(2*pi*3*f0*t);  
y0 = y1 + y2 + y3;  
A = (2*exp(-1.5*t)) .* sin(2*pi*0.65*t);  
call = A.*y0;  
soundsc(call,fs)  
plot(t,call)  
f0 = 185;
```

A mouse cursor is pointing to the Command History window, specifically to the line `y1 = sin(2*pi*f0*t);`.

# The MATLAB® Editor

edit

The screenshot displays the MATLAB Editor environment. The main window is titled 'Editor - Untitled\*' and contains a script with the following code:

```
1 fs = 4000;  
2 t = 0:1/fs:1.5;  
3 f0 = 175;  
4 y1 = sin(2*pi*f0*t);  
5 y2 = sin(2*pi*2*f0*t);  
6 y3 = sin(2*pi*3*f0*t);  
7 y0 = y1 + y2 + y3;  
8 A = (2*exp(-1.5*t)).*sin(2*pi*0.65*t);  
9 call = A.*y0;  
10 soundsc(call,fs)  
11 plot(t,call)  
12
```

Two orange arrows point to the code: one to the variable `fs` on line 1, and another to the `soundsc` function call on line 10.

The 'Workspace' window on the right shows the following variables:

Name	Value	Size	Class
A	<1x6001 doubl...	1x6001	double
call	<1x6001 doubl...	1x6001	double
f0	175	1x1	double
fs	4000	1x1	double
t	<1x6001 doubl...	1x6001	double
y0	<1x6001 doubl...	1x6001	double
y1	<1x6001 doubl...	1x6001	double
y2	<1x6001 doubl...	1x6001	double
y3	<1x6001 doubl...	1x6001	double

The 'Command Window' at the bottom left shows the execution of the script:

```
>> fs = 4000;  
>> t = 0:1/fs:1.5;  
>> f0 = 175;  
>> y1 = sin(2*pi*f0*t);  
>> y2 = sin(2*pi*2*f0*t);  
>> y3 = sin(2*pi*3*f0*t);  
>> y0 = y1 + y2 + y3;  
>> A = (2*exp(-1.5*t)).*sin(2*pi*0.65*t);  
>> call = A.*y0;  
>> soundsc(call,fs)  
>> plot(t,call)  
fx >>
```

The 'Command History' window on the bottom right shows the same sequence of commands. A right-click context menu is open over the first line of the history, listing options such as 'Evaluate Selection', 'Create Script', 'Create Shortcut', 'Profile Code', 'Cut', 'Copy', 'Delete Selection', 'Delete to Selection', 'Select All', 'Find...', 'Print...', 'Print Selection...', and 'Page Setup...'.

# Script Files

```
H1 line % CALLMODEL Models a blue whale B call.
        %
        % Uses a model of the form  $y = A \cdot y_0$ 
        % where  $A = A_0 \exp(-B \cdot t) \cdot \sin(2\pi \cdot f_m \cdot t)$ 
        % and  $y_0$  is a sum of harmonics
        %  $y_n = \sin(2\pi \cdot n \cdot f_0 \cdot t)$ 

        % Create the time base for the signal.
        fs = 4000;
        t = 0:(1/fs):1.5;

        % Set the fundamental frequency of the call.
        f0 = 175;

        % Create the harmonics.
        y0 = sin(2*pi*f0*t) + sin(2*pi*2*f0*t) +
            sin(2*pi*3*f0*t);
```

Help

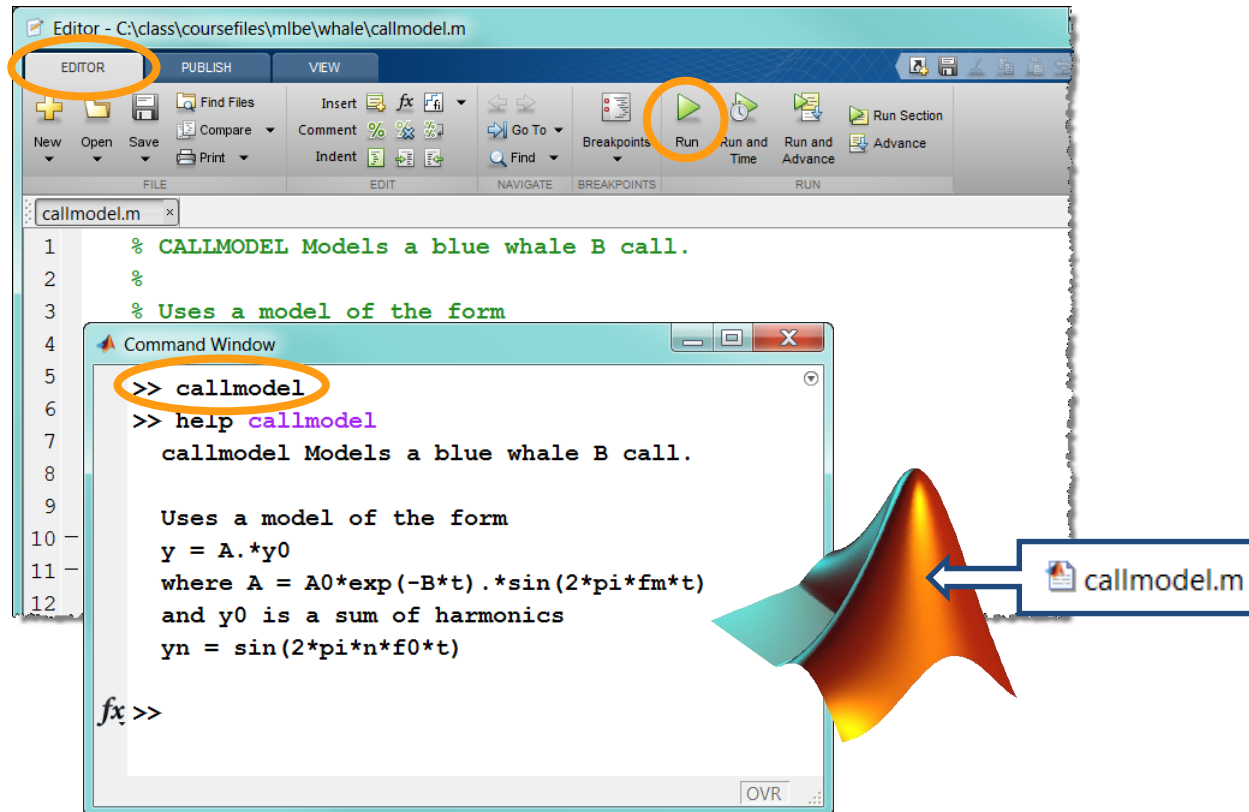
Code

Comments

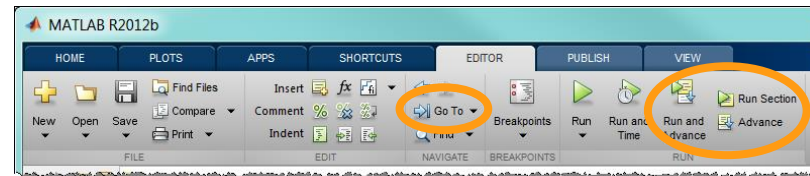
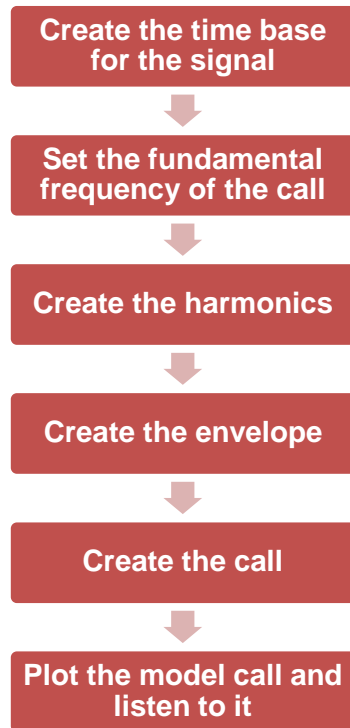
```
>> edit callmodel.m
```



# Running a Script



# Code Sections



%%

```
%% Set the fundamental frequency of the call.
f0 = 175;

%% Create the harmonics.
y0 = sin(2*pi*f0*t) + sin(2*pi*2*f0*t) + sin(2*pi*3*f0*t);

%% Create the envelope
% Set the additional parameters in the model.
A0 = 2; % Initial amplitude.
B = 1.5; % Amplitude decay rate.
fm = 0.65; % Frequency of the modulating envelope.
% Create the envelope
A = A0*exp(-B*t).*sin(2*pi*fm*t);

%% Create the call.
call = A.*y0;
```

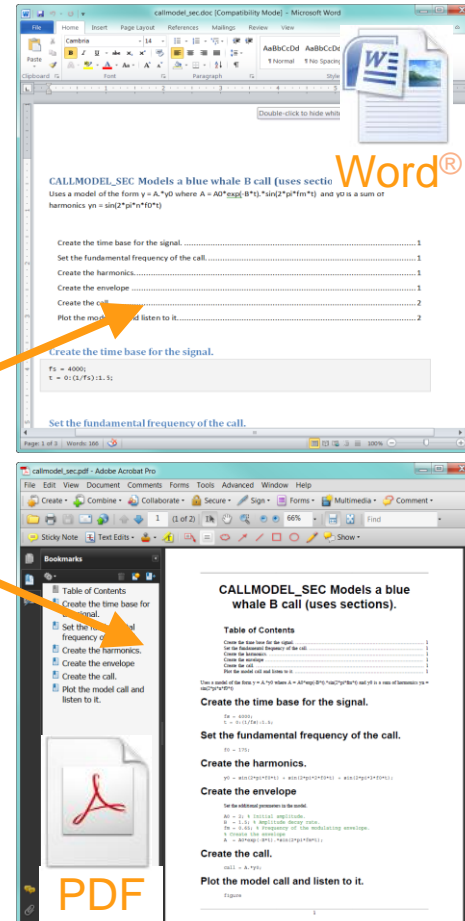
# Publishing Code

```
%% Set the fundamental frequency of the call.
f0 = 175;

%% Create the harmonics.
y0 = sin(2*pi*f0*t) + sin(2*pi*2*f0*t) + sin(2*pi*3*f0*t);

%% Create the envelope
% Set the additional parameters in the model.
A0 = 2; % Initial amplitude.
B = 1.5; % Amplitude decay rate.
fm = 0.65; % Frequency of the modulating envelope.
% Create the envelope
A = A0*exp(-B*t).*sin(2*pi*fm*t);

%% Create the call.
call = A.*y0;
```



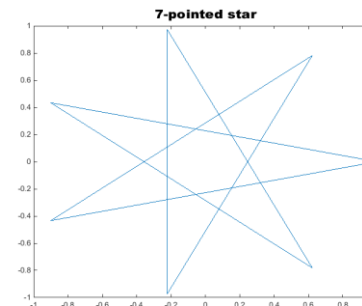
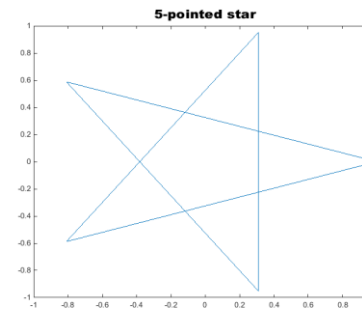
# Exercise: N-Pointed Star

- Create a script to draw a 5-pointed and a 7-pointed star using the equations

$$x = \cos\left(\frac{(n-1)t\pi}{n}\right)$$

$$y = \sin\left(\frac{(n-1)t\pi}{n}\right)$$

with  $t = 0, 1, 2, \dots, 100$



```
>> edit nPointStar
```

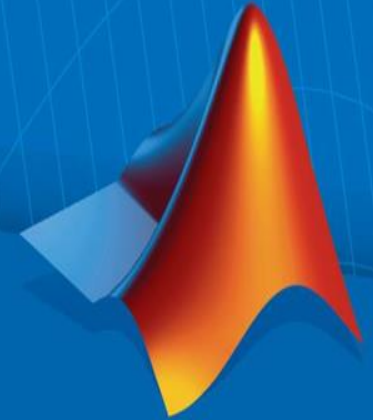


## Chapter 6 Test Your Knowledge

- T/F: Anything following a % sign is ignored by MATLAB as a comment.
- T/F: In section mode, you can modify your code and rerun it without having to save the file.
- T/F: Script files can access and modify any variables already in the base MATLAB workspace.







Q & A

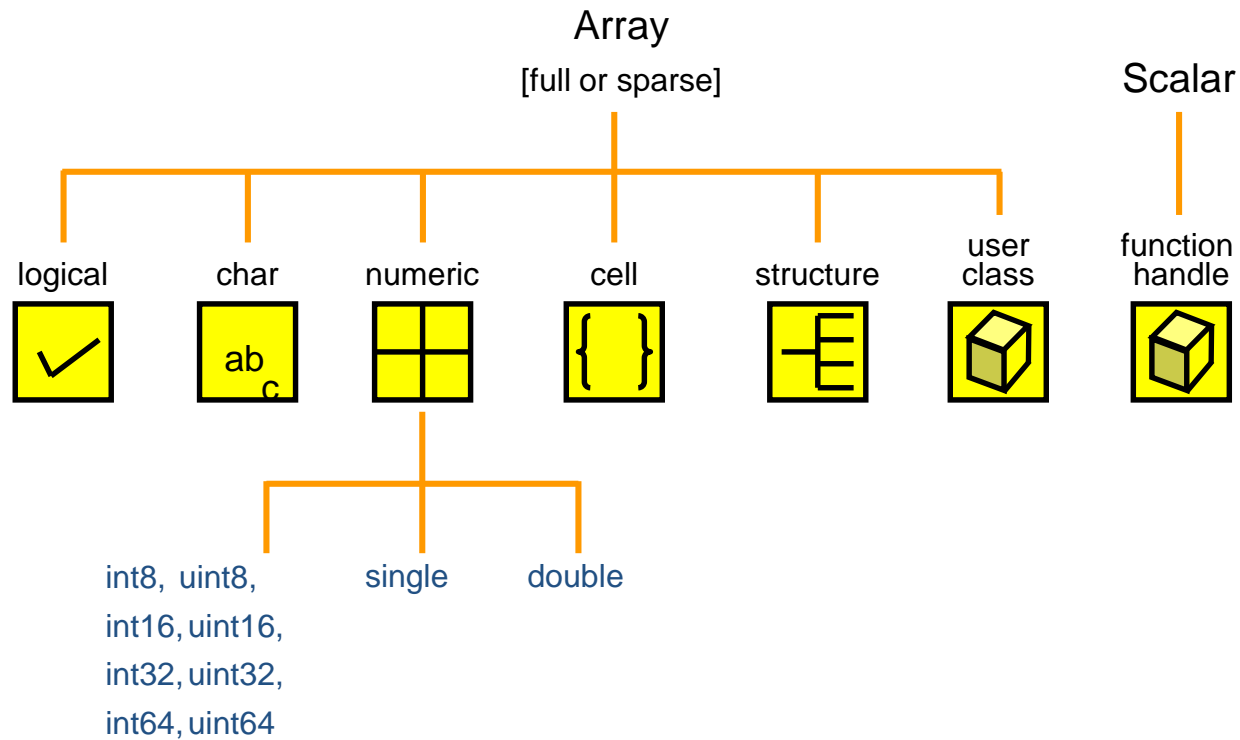
Thank you very much

Have a good time

# Course Outline

- Working with the MATLAB User Interface
- Variables and Commands
- Array Creation and Analysis
- Working with Data Files
- Visualization with Array
- Automating Commands with Scripts
- Appendix: Data Type

# MATLAB® Data Types



# Integer Arrays



$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

**Construct**    `>> A = uint8([1, 2; 3, 4]);`

**Access**    `>> A(:,2)`  
              `ans =`  
              2  
              4

# Nondouble Arithmetic

[back](#)

$$\begin{array}{|c|c|} \hline \text{single} & \text{single} \\ \hline \end{array} + \begin{array}{|c|c|} \hline \text{single} & \text{single} \\ \hline \end{array} = \begin{array}{|c|c|} \hline \text{single} & \text{single} \\ \hline \end{array}$$

$$\begin{array}{|c|c|} \hline \text{double} & \text{single} \\ \hline \end{array} + \begin{array}{|c|c|} \hline \text{double} & \text{single} \\ \hline \end{array} = \begin{array}{|c|c|} \hline \text{double} & \text{single} \\ \hline \end{array}$$

$$\begin{array}{|c|c|} \hline \text{int8} & \text{int8} \\ \hline \end{array} + \begin{array}{|c|} \hline \text{double} \\ \hline \end{array} = \begin{array}{|c|c|} \hline \text{int8} & \text{int8} \\ \hline \end{array}$$

$$\begin{array}{|c|c|} \hline \text{int8} & \text{int8} \\ \hline \end{array} + \begin{array}{|c|c|} \hline \text{double} & \text{double} \\ \hline \end{array} = \begin{array}{|c|c|} \hline \text{double} & \text{double} \\ \hline \end{array}$$

$$\begin{array}{|c|c|} \hline \text{int8} & \text{int8} \\ \hline \end{array} + \begin{array}{|c|c|} \hline \text{int8} & \text{int8} \\ \hline \end{array} = \begin{array}{|c|c|} \hline \text{int8} & \text{int8} \\ \hline \end{array}$$

$$\begin{array}{|c|c|} \hline \text{int8} & \text{int8} \\ \hline \end{array} + \begin{array}{|c|c|} \hline \text{int16} & \text{int16} \\ \hline \end{array} = \begin{array}{|c|c|} \hline \text{int16} & \text{int16} \\ \hline \end{array}$$

# Characters and Strings

back

```
>> y = x  
>> y = 'x'
```

variable  
character

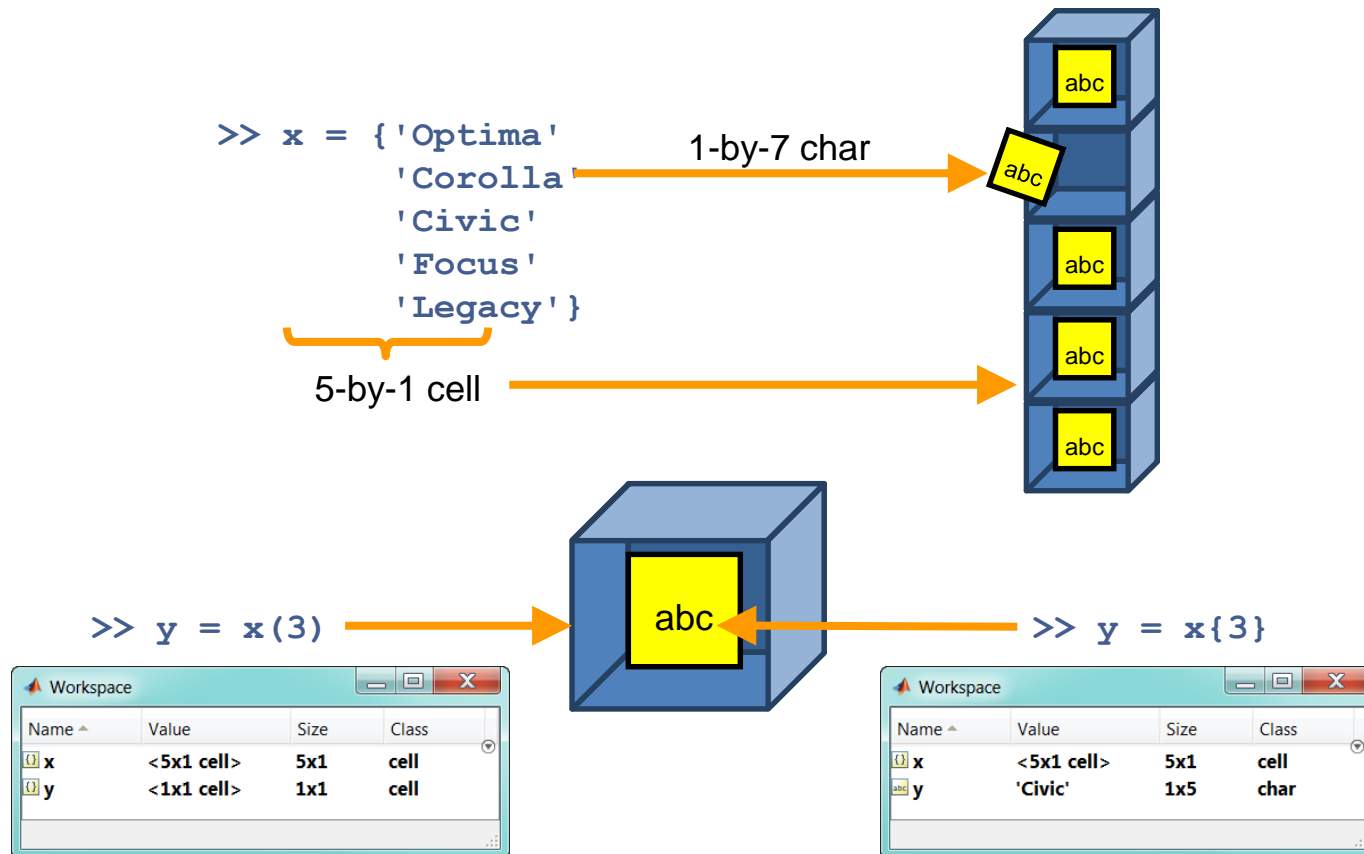
```
>> MarkA = 'Friends, Romans, countrymen, lend me your ears';
```

1-by-46 char array

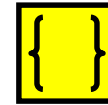
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
'F'	'r'	'i'	'e'	'n'	'd'	's'	','	' '	'R'	'o'	'm'	'a'	'n'	's'	','	' '	'c'

```
>> FriendNationality = MarkA(10:15)
```

# Cell Arrays (1)



## Cell Arrays (2)



3.14		foo
bar	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	2
	B	

**Construct**    `>> A = {pi,[],'foo';...  
                  'bar',eye(2),2;...  
                  [],B,[]};`

**Access**    `>> A{2,2}(:,2)`  
              `ans =`  
                  0  
                  1





# Structures

MyWhale

name	Mushu
fundamental frequency	175
amplitude	2
decay rate	1.5
modulation frequency	0.65
number of harmonics	3

Fields

`[x,t] = callmodel_fun_s(MyWhale);`

```
>> edit  
whalestruct
```



# Structure Arrays

```
>> aquarium
```

```
aquarium =  
1x3 struct array with fields:  
    name  
    fundamentalfreq  
    amplitude  
    decayrate  
    modulationfreq  
    harmonics
```

```
aquarium(1)  
  name ————— Mushu  
fundamental frequency — 175  
  amplitude ————— 2  
  decay rate ————— 1.5  
modulation frequency — 0.65  
number of harmonics — 3
```

```
aquarium(2)  
  name ————— Willy  
fundamental frequency — 180  
  amplitude ————— 2.5  
  decay rate ————— 1.9  
modulation frequency — 0.5  
number of harmonics — 5
```

```
aquarium(3)  
  name ————— Shamu  
fundamental frequency — 165  
  amplitude ————— 1  
  decay rate ————— 0.8  
modulation frequency — 1.0  
number of harmonics — 4
```



```
>> ex_aquarium
```



# Indexing into Structure Arrays

