LECTURE 2

Using Script Files

and

Managing Data

Recall that you can perform operations in MATLAB in two ways:

1. In the interactive mode: all commands are entered directly in the Command window,

In the *script* file mode: By running a MATLAB program stored in *script* file. This type of file contains MATLAB commands, so running it is equivalent to typing all the commands—one at a time—at the Command window prompt. You can run the file by typing its name at the Command window prompt.

In the interactive mode

disp can be used to display the value of a variable.

```
>> abc = [5  9  1;  7  2  4];
>> disp(abc)
     5     9     1
     7     2     4


>> disp('The problem has no solution.')

The problem has no solution.
>>
```

A 2 × 3 array is assigned to variable `abc`.

The `disp` command is used to display the `abc` array.

The array is displayed without its name.

The `disp` command is used to display a message.

```
>> x=3.0;
>> y=3^x;
>> disp(y)
    27
```

Example

```
>> disp(' And now for something completely different' )

 And now for something completely different
```

Example

```
>> disp('----------------------------------');
```

input command

The form of the input command is:

```
variable_name = input('string with a message that
                       is displayed in the Command Window')
```

Example

>> x=input('Enter r=')

Enter r=5

x =

5

Example

```
yr=[1984 1986 1988 1990 1992 1994 1996];
pop=[127 130 136 145 158 178 211];
tableYP(:,1)=yr';
tableYP(:,2)=pop';
disp('         YEAR      POPULATION')
disp('                   (MILLIONS)')
disp(' ')
disp(tableYP)
```

The population data is entered in two row vectors.

`yr` is entered as the first column in the array `tableYP`.

`pop` is entered as the second column in the array `tableYP`.

Display heading (first line).

Display heading (second line).

Display an empty line.

Display the array `tableYP`.

When this script file (saved as PopTable) is executed, the display in the Command Window is:

```
>> PopTable
          YEAR      POPULATION
                    (MILLIONS)

          1984         127
          1986         130
          1988         136
          1990         145
          1992         158
          1994         178
          1996         211
```

Headings are displayed.

An empty line is displayed.

The `tableYP` array is displayed.

Input command can also be used to assign a string to a variable

```
variable_name = input('prompt message','s')
```

where the 's' inside the command defines the characters that will be entered as a string.

```
>> name=input('What is your name  ?  ' , 's')
What is your name  ?  Hakan
name =
Hakan
>> name
Hakan
```

fprintf command

- The fprintf command can be used to display output (text and data) on the screen or to save it to file.The output can be formatted.

Using the `fprintf` command to display text:

To display text, the `fprintf` command has the form:

```
fprintf('text typed in as a string')
```

>> fprintf('The problem has no solution \n')

This is displayed on the screen. The \n starts a new line.

The problem has no solution

With the fprintf command it is possible to start a new line in the middle of a string.

```
fprintf('The problem, as entered, has no solution.\nPlease check the input data.')
```

When this line executes, the display in the Command Window is:

```
The problem, as entered, has no solution.
Please check the input data.
```

Using the fprintf command to display a mix of text and numerical data

To display a mix of text and a number (value of a variable), the `fprintf` command has the form:

```
fprintf('text as string %-5.2f additional text',
                                         variable_name)
```

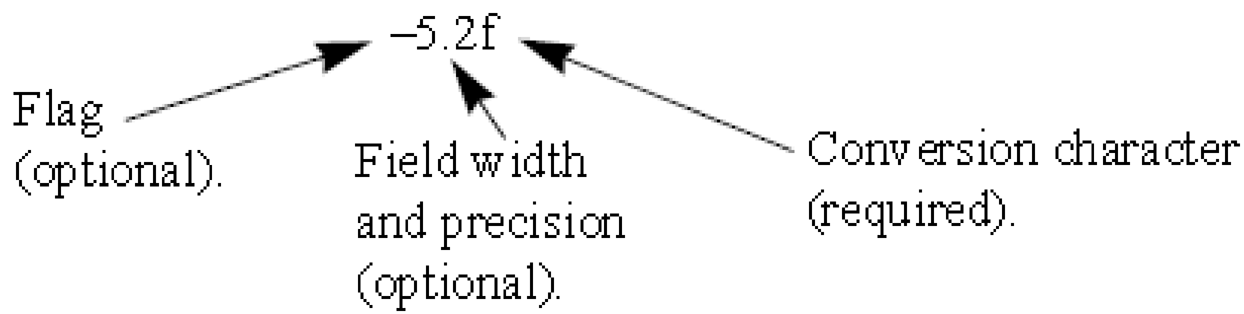
The % sign marks the spot where the number is inserted within the text.

Formatting elements (define the format of the number).

The name of the variable whose value is displayed.

continued

The formatting elements are:



The flag, which is optional, can be one of the following three characters:

Character used for flag	**Description**
– (minus sign)	Left-justifies the number within the field.
+ (plus sign)	Prints a sign character (+ or –) in front of the number.
0 (zero)	Adds zeros if the number is shorter than the field.

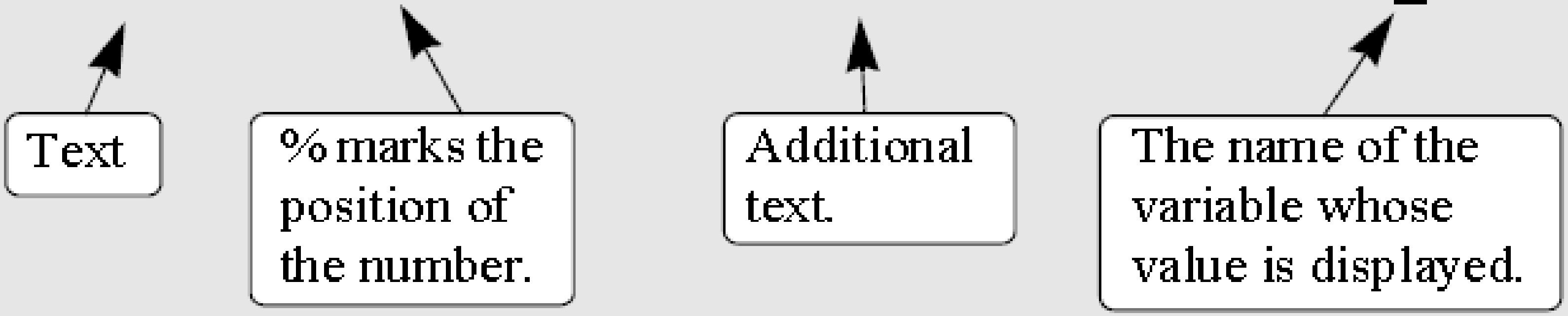
The last element in the formatting elements, which is required, is the conversion character, which specifies the notation in which the number is displayed. Some of the common notations are:

e	Exponential notation using lower-case e (e.g., 1.709098e+001).
E	Exponential notation using upper-case E (e.g., 1.709098E+001).
f	Fixed-point notation (e.g., 17.090980).
g	The shorter of e or f notations.
G	The shorter of E or f notations.
i	Integer.

Example

```
% This script file calculates the average points scored in three games.
% The values are assigned to the variables by using the input command.
% The fprintf command is used to display the output.
game(1) = input('Enter the points scored in the first game    ');
game(2) = input('Enter the points scored in the second game    ');
game(3) = input('Enter the points scored in the third game    ');
ave_points = mean(game);
```

```
fprintf('An average of %f points was scored in the three games.',ave_points)
```



Solution save the file as game.m

```
Enter the points scored in the first game    75

Enter the points scored in the second game   60

Enter the points scored in the third game    81

An average of 72.000000 points was scored in the three games.
>>
```

The display generated by the `fprintf` command combines text and a number (value of a variable).

The `fprintf` command is vectorized. This means that when a variable that is a vector or a matrix is included in the command, the command repeats itself until all the elements are displayed. If the variable is a matrix, the data is used column by column.

Example:

Create a 2x5 matrix T in which the first row contains numbers 1 through 5 and the second row shows the corresponding square roots

```
x=1:5;
y=sqrt(x);
T=[x; y]
fprintf('If the number is: %i, its square root is: %f\n',T)
```

Create a vector x.

Create a vector y.

Create 2×5 matrix T, first row is x, second row is y.

The fprintf command displays two numbers from T in every line.

Save it as table.m

```
T =
    1.0000    2.0000    3.0000    4.0000    5.0000
    1.0000    1.4142    1.7321    2.0000    2.2361
If the number is: 1, its square root is: 1.000000
If the number is: 2, its square root is: 1.414214
If the number is: 3, its square root is: 1.732051
If the number is: 4, its square root is: 2.000000
If the number is: 5, its square root is: 2.236068
```

The 2×5 matrix `T`.

The `fprintf` command repeats five times, using the numbers from the matrix `T` column after column.

Using the fprintf command to save output to a file

Writing output to a file requires three steps:

a) Opening a file using the `fopen` command.

b) Writing the output to the open file using the `fprintf` command.

c) Closing the file using the `fclose` command.

Step a:

Before data can be written to a file, the file must be opened. This is done with the `fopen` command, which creates a new file or opens an existing file. The `fopen` command has the form:

```
fid= fopen('file_name','permission')
```

`fid` is a variable called the file identifier. A scalar value is assigned to `fid` when `fopen` is executed. The file name is written (including its extension) within single quotes as a string. The permission is a code (also written as a string) that tells how the file is opened. Some of the more common permission codes are:

`'r'`	Open file for reading (default).
`'w'`	Open file for writing. If the file already exists, its content is deleted. If the file does not exist, a new file is created.
`'a'`	Same as `'w'`, except that if the file exists the written data is appended to the end of the file.
`'r+'`	Open file for reading and writing.
`'w+'`	Open file for writing and writing. If the file already exists, its content is deleted. If the file does not exists, a new file is created.
`'a+'`	Same as `'w+'`, except that if the file exists the written data is appended to the end of the file.

If a permission code is not included in the command, the file opens with the default code `'r'`. Additional permission codes are described in the help menu.

- Once the file is open ,the fprintf command can be used to write output to the file.The variable fid is inserted inside the command.fprintf command has the form.

```
fprintf(fid,'text %-5.2f additional text',vari
                                          able_name)
```

`fid` is added to the `fprintf` command.

Step *c*:

When the writing of data to the file is complete, the file is closed using the `fclose` command. The `fclose` command has the form:

```
fclose(fid)
```

Example

-
- We need a conversion table from pound force (lb_f) to Newton (N).
- 1 Newton =4.448 pound force
- Newton is a force unit in SI system
- Pound force is force unit in English system
- Write script file and save it as conversion.m

```
% Conversion table from pound force  to Newton
% FN-force in Newton( N)
% Flbf-force in pound force(lbf)
Flbf=200:200:2000;
FN=4.448*Flbf;
TABLE=[Flbf;FN];
fid1=fopen('FlbftoFN.txt','w')
fprintf(fid1,'Force Conversion Table \n  \n');
fprintf(fid1, '   Poundforce        Newtons  \n');
fprintf(fid1 , '  %10.2f      %10.2f   \n' ,TABLE)
fclose(fid1)
```

Example

- Compute the capacitance of a parallel plate capacitor with air dielectric is given by

C= 0.0000088855 $\frac{A}{s}$

where C=capacitance in (microfarads)

A =area in (m^2)

s=separation distance in (m)

Let A=0.43 (m^2) and s=0.0005 (m)

Write a script file , name the script file CAPACITOR.m and save the result in a file called capacitance.txt.

```
% Calculation of capacitance of capacitor
A=0.43;
s=0.005;
fid=fopen('capacitance.txt','w');
% calculate the capacitance
C=0.000008855*A/s;
% print the result to a file
fprintf(fid, 'Capacitance C=%10.5f   \n',  C );
fclose(fid);
```

save and load commands

- The save command

The save command is used for saving the variables (all or some of them) that are stored in the workspace. The two simplest forms of the save command are:

```
save file_name
```

and

```
save('file_name')
```

save command can also be used for saving only some of the variables that are in workspace.

```
save file_name var1 var2
```

or

```
save('file_name','var1','var2')
```

The save command can also be used for saving in ASCII format, which can be read by applications outside MATLAB. Saving in ASCII format is done by adding the argument -ascii in the command (for example, save file_name

```
>> clear
>> A=magic(3)
A =
    8    1    6
    3    5    7
    4    9    2

>> save data.dat A -ascii
>>
```

load command

The load command can be used for retrieving variables that were saved with the save command back to the workspace, and for importing data that was created with other applications and saved in ASCII format or in text (.txt) files. Variables that were saved with the save command in .mat files can be retrieved with the command:

```
load file_name
```

or

```
load('file_name')
```

The load command can be used for retrieving some of the variables that are in the saved .mat file. For example, to retrieve two variables named var1 and var2, the command is:

```
load file_name var1 var2
```

or

```
load('file_name','var1','var2')
```

When data is loaded from an ASCII or text file into the workspace it has to be assigned to a variable name. Data in ASCII format can be loaded with either of the following two forms of the load command:

```
load file_name
```

or

```
VarName=load('file_name')
```

If the data is in a text file, the extension .txt has to be added to the file name. The form of the load command is then:

```
load file_name.txt
```

or

```
VarName=load('file_name.txt')
```

Example

Data shown below is written in a note pad and saved as PumpData.dat

Use load command to retrieve data

Continued on next slide

600	2
800	6
1000	10
1500	12
2000	14
4000	20
6000	30
8000	40
10000	50
20000	55
30000	57

```
>> data=load('PumpData.dat')
data =
      600        2
      800        6
     1000       10
     1500       12
     2000       14
     4000       20
     6000       30
     8000       40
    10000       50
    20000       55
    30000       57
```

Example

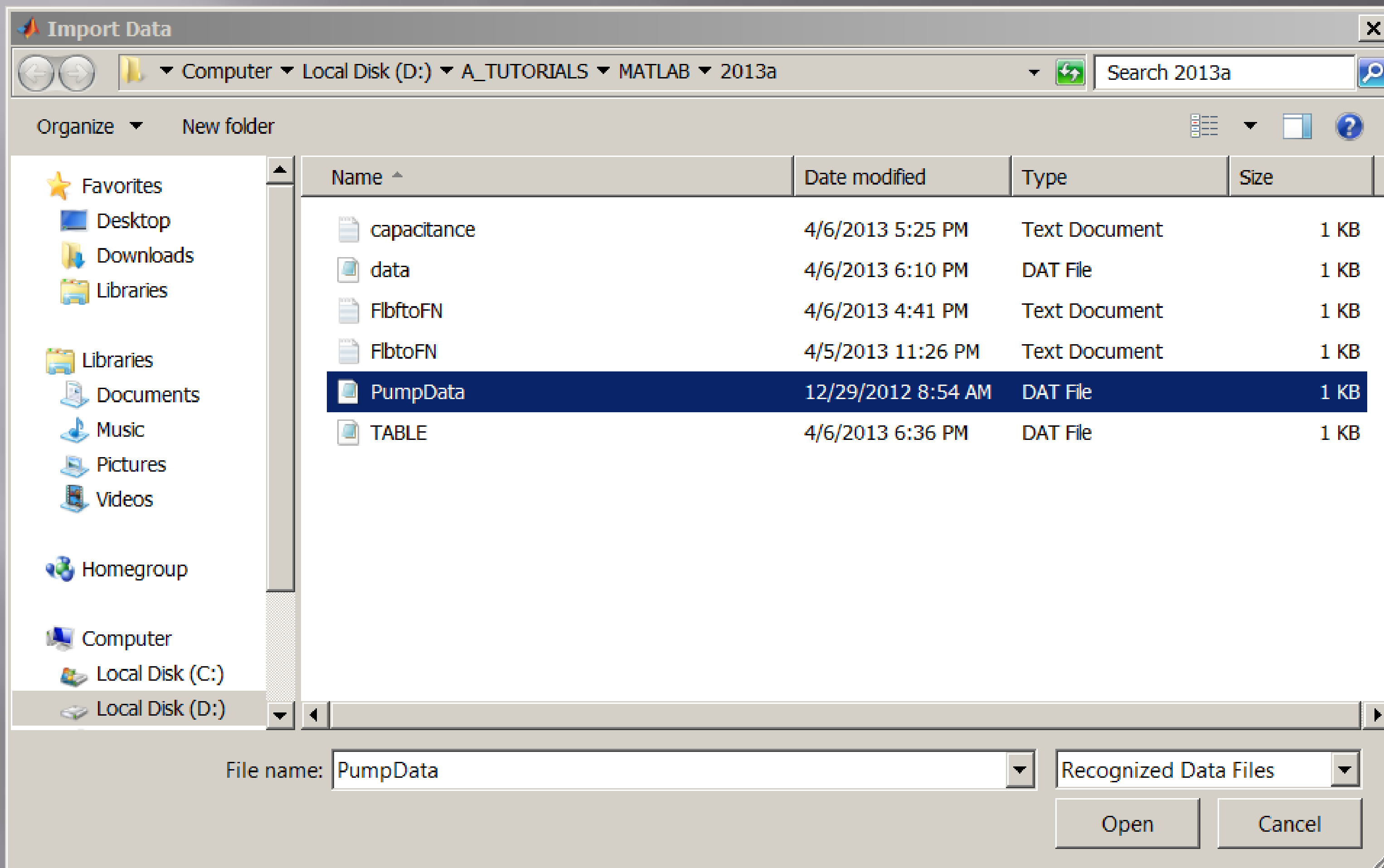
```
>> x=0:0.1:0.4
x =
        0    0.1000    0.2000    0.3000    0.4000
>> y=exp(x)
y =
    1.0000    1.1052    1.2214    1.3499    1.4918
>> T=[x;y]
T =
        0    0.1000    0.2000    0.3000    0.4000
    1.0000    1.1052    1.2214    1.3499    1.4918
>>  save TABLE.dat T -ascii
```

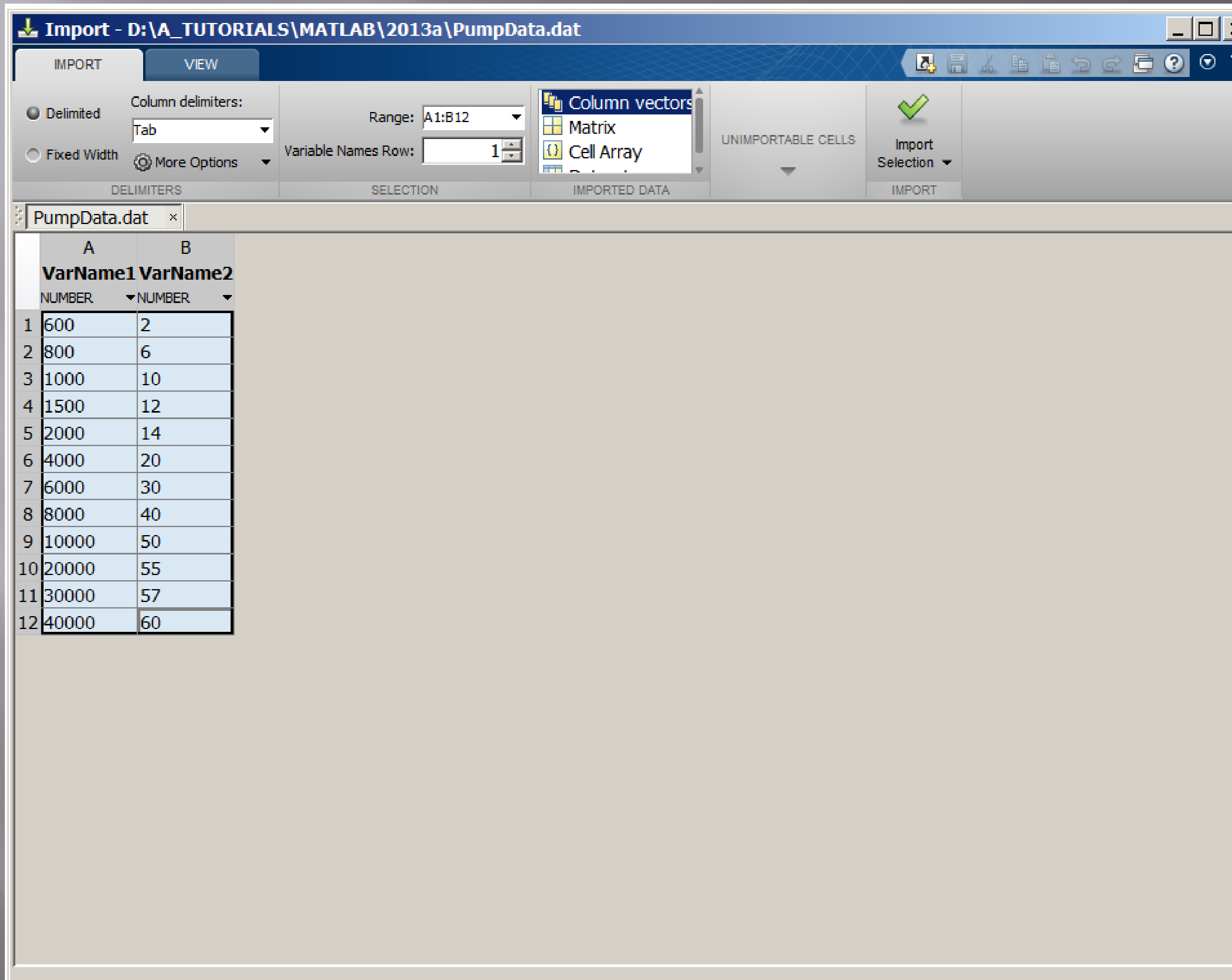
Now let us retrieve this data saved in file named table

```
>> data=load('TABLE.dat')
data =
        0    0.1000    0.2000    0.3000    0.4000
   1.0000    1.1052    1.2214    1.3499    1.4918
>>
```

We can also use Import Wizard to retrieve data

Suppose we want to import the data in PumpData.dat file


Import Data
Computer ▸ Local Disk (D:) ▸ A_TUTORIALS ▸ MATLAB ▸ 2013a
Search 2013a
Organize
New folder
Favorites
Desktop
Downloads
Libraries
Libraries
Documents
Music
Pictures
Videos
Homegroup
Computer
Local Disk (C:)
Local Disk (D:)
Name
Date modified
Type
Size
capacitance 4/6/2013 5:25 PM Text Document 1 KB
data 4/6/2013 6:10 PM DAT File 1 KB
FlbftoFN 4/6/2013 4:41 PM Text Document 1 KB
FlbtoFN 4/5/2013 11:26 PM Text Document 1 KB
PumpData 12/29/2012 8:54 AM DAT File 1 KB
TABLE 4/6/2013 6:36 PM DAT File 1 KB
File name: PumpData
Recognized Data Files
Open
Cancel


Import - D:\A_TUTORIALS\MATLAB\2013a\PumpData.dat
IMPORT
VIEW
Delimited
Fixed Width
Column delimiters:
Tab
More Options
DELIMITERS
Range: A1:B12
Variable Names Row: 1
SELECTION
Column vectors
Matrix
Cell Array
IMPORTED DATA
UNIMPORTABLE CELLS
Import Selection
IMPORT
PumpData.dat
A
B
VarName1
VarName2
NUMBER
NUMBER
1 600 2
2 800 6
3 1000 10
4 1500 12
5 2000 14
6 4000 20
7 6000 30
8 8000 40
9 10000 50
10 20000 55
11 30000 57
12 40000 60

STEPS IN ENGINEERING PROBLEM SOLVING

1. Understand the purpose of the problem.
2. Collect the known information. Realize that some of it might later be found unnecessary.
3. Determine what information you must find.
4. Simplify the problem only enough to obtain the required information. State any assumptions you make.
5. Draw a sketch and label any necessary variables.
6. Determine which fundamental principles are applicable.
7. Think generally about your proposed solution approach and consider other approaches before proceeding with the details.
8. Label each step in the solution process.
9. If you solve the problem with a program, hand check the results using a simple version of the problem. Checking the dimensions and units and printing the results of intermediate steps in the calculation sequence can uncover mistakes.
10. Perform a "reality check" on your answer. Does it make sense? Estimate the range of the expected result and compare it with your answer. Do not state the answer with greater precision than is justified by any of the following:
    (a) The precision of the given information.
    (b) The simplifying assumptions.
    (c) The requirements of the problem.

    Interpret the mathematics. If the mathematics produces multiple answers, do not discard some of them without considering what they mean. The mathematics might be trying to tell you something, and you might miss an opportunity to discover more about the problem.

GETTING HELP

`help`	On-line help. `help` lists all the primary help topics. `help <command>` displays information about the command.
`doc`	On-line help hypertext reference manual. `doc` accesses the manual. `doc <command>` displays information about the command.
`helpbrowser`	Accesses the main page of the on-line reference manual.
`type <command>`	Displays the actual MATLAB code for this command.
`lookfor <keyword>`	Searches all MATLAB commands for this keyword.
`who`	Lists all the current variables.
`whos`	Lists all the current variables in more detail than `who`.
`demo`	Runs demonstrations of many of the capabilities of MATLAB.
`save`	Saves all of your variables.
`load`	Loads back all of the variables which have been saved previously.
`^C`	Abort the command which is currently executing (i.e., hold down the control key and type "c").

diary

Diary command saves your input to MATLAB and most of the output to disk. This command toggles diary on and off. (If no file is given, it is saved to the file diary in the current directory.)

diary on turns the diary on.

diary off turns the diary off.

Homework 2

1) . The wind chill temperature, T_{wc}, is the air temperature felt on exposed skin due to wind. In U.S. customary units it is calculated by:

$$T_{wc} = 35.74 + 0.6215\,T - 35.75\,v^{0.16} + 0.4275\,T\,v^{0.16}$$

where T is the temperature in degrees F, and v is the wind speed in mi/h. Write a MATLAB program in a script file that displays the following chart of wind chill temperature for given air temperature and wind speed in the Command Window:

	Temperature (F)								
	40	30	20	10	0	-10	-20	-30	-40
Speed (mi/h)									
10	34	21	9	-4	-16	-28	-41	-53	-66
20	30	17	4	-9	-22	-35	-48	-61	-74
30	28	15	1	-12	-26	-39	-53	-67	-80
40	27	13	-1	-15	-29	-43	-57	-71	-84
50	26	12	-3	-17	-31	-45	-60	-74	-88
60	25	10	-4	-19	-33	-48	-62	-76	-91

2)

The variation of vapor pressure p (in units of mm Hg) of benzene with temperature in the range of $0 \le T \le 42°C$ can be modeled with the equation (Handbook of Chemistry and Physics, CRC Press)

$$\log_{10} p = b - \frac{0.05223a}{T}$$

where $a = 34172$ and $b = 7.9622$ are material constants and T is absolute temperature (K). Write a program in a script file that calculates the pressure for various temperatures. The program should create a vector of temperatures from $T = 0°C$ to $T = 42°°C$ with increments of 2 degrees, and display a two-column table p and T, where the first column temperatures in $°C$, and the second column the corresponding pressures in mm Hg.

3)

For many gases the temperature dependence of the heat capacity C_p of can be described in terms of a cubic equation:

$$C_p = a + bT + cT^2 + dT^3$$

The following table gives the coefficients of the cubic equation for four gases. C_p is in joules/(g mol)(°C) and T is in °C.

Gas	a	b	c	d
SO_2	38.91	3.904×10^{-2}	-3.105×10^{-5}	8.606×10^{-9}
SO_3	48.50	9.188×10^{-2}	-8.540×10^{-5}	32.40×10^{-9}
O_2	29.10	1.158×10^{-2}	-0.6076×10^{-5}	1.311×10^{-9}
N_2	29.00	0.2199×10^{-2}	-0.5723×10^{-5}	-2.871×10^{-9}

Calculate the heat capacity for each gas at temperatures ranging between 200 and 400 °C at 20 °C increments. To present the results, create an 11×5 matrix where the first column is the temperature, and the second through fifth columns are the heat capacities of SO_2, SO_3, O_2, and N_2, respectively.