Topic 6

Introduction to External File

Introduction

Output from script can be stored in an external file to be manipulated/printed

Input data for script file can come from file that has been created by other source.

- Many methods of reading and writing to files.
- This section only covers **`save`** and **`load`** **commands**.
- Command syntax is convenient for command line use. With command syntax, strings do not need to be enclosed in single quotation marks.
- **Cannot** use **`save/load`** **command syntax** if inputs such as filename are **variables**, instead use **functions**.

Other methods of reading and writing to files will be covered later.

Writing Data File – `save` command

- **`Save`** command is used to **write** or **append** data from a matrix to a **data** file.

`save` *filename variablename* **`-ascii`**

```
>> uji=randi(9,2,3)
uji =
     8     6     8
     9     1     9
```

Creates matrix **uji**

```
>> save ujifail.dat uji-ascii
```

Saves **uji** matrix to a data file **ujifail**

No single quotation **(‘ ‘)**

This writes a new file. If a file with the same name already exists, it will be over-written.

```
>> type ujifail.dat

   8.0000000e+00   6.0000000e+00   8.0000000e+00
   9.0000000e+00   1.0000000e+00   9.0000000e+00
```

Displays the file content of **ujifail**

Appending Data File - `save` command

`save` *filename variablename* **`– ascii -append`**

```
>> uji2=randi(9,3)
uji2 =
     7     4     7
     7     6     1
     7     2     3
```

```
>> save ujifail.dat uji2 -ascii -append
```

Creates another matrix **uji2**

Appends **uji2** to the existing data file

```
>> type ujifail.dat

   8.0000000e+00   6.0000000e+00   8.0000000e+00
   9.0000000e+00   1.0000000e+00   9.0000000e+00
   7.0000000e+00   4.0000000e+00   7.0000000e+00
   7.0000000e+00   6.0000000e+00   1.0000000e+00
   7.0000000e+00   2.0000000e+00   3.0000000e+00
```

uji

uji2

Technically any size matrix can be appended

Reading from Data File – `load` command

`load` command is used to **read matrix content** of a file **into a matrix variable**. The matrix variable name has the **same as the name of the file**.

`Load` *filename.ext*

```
>> load ujifail.dat
>> ujifail
ujifail =
     8     6     8
     9     1     9
     7     4     7
     7     6     1
     7     2     3
```

No single quotation (‘ ‘) for string

The matrix from the file is assigned to the variable (same name as the file)

`load` command only works if the number of values in each line is the same so that they can be stored in a matrix.

`save` command can only writes from a matrix to a file.

Otherwise low-level I/O functions must be used.

Save & load command - Example

Create matrix below and save in a data file.

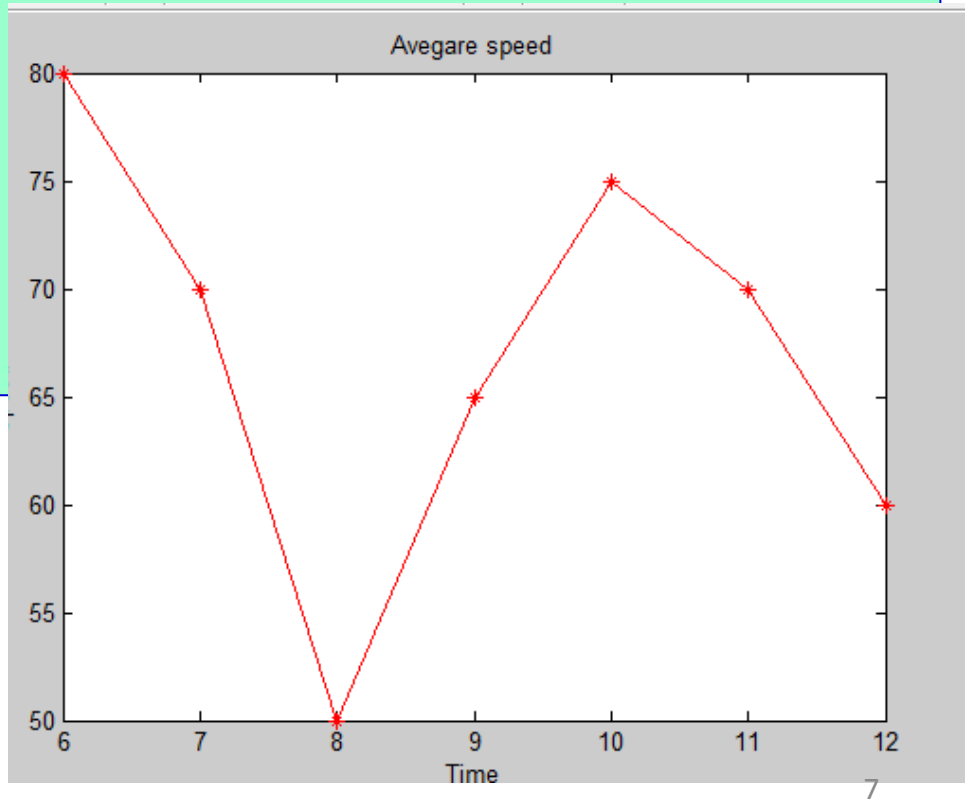
Time	6	7	8	9	10	11	12
Average speed	85	70	50	65	85	75	60

```
>> timespeed=[6:12;80 70 50 65 75 70 60]

timespeed =
     6     7     8     9    10    11    12
    80    70    50    65    75    70    60

>> save datatimespeed.dat timespeed -ascii
```

You can see **datatimespeed.dat** file in the current folder window.

Save & load command - Example (cont'd)

```
% use load command and plot

load datatimespeed.dat
time=datatimespeed(1,:);  % assign first row to time
speed=datatimespeed(2,:); % assign second row to speed
plot(time,speed,'-r*')
xlabel('Time')
ylabel('Speed')
title('Avegare speed')
```



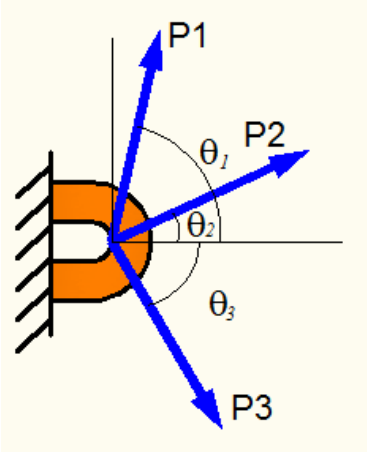
`load` function

- In the previous slide, load command was used to retrieve a matrix from a data file.
- To allow the user to select any of the existing file, we assign the filename to a variable first, then call the file.
- In order to do this we have to use **`load` function syntax**.

Problem Example

Problem Example 1

Determine the resultant of the forces acting on the bracket show. Input must be read form a data file in the format as shown in the sample below.



Data file

```
P1  P2  P3
A1  A2  A3
```

Problem Example 1 - cont'd

```
% Resultant of 3 concurrent Forces

data1=input('Enter file name: ','s');
fname=load (data1, '-ascii');
f=fname(1,:);
a=fname(2,:);
fx=f.*cosd(a);
fy=f.*sind(a);
fxtot=sum(fx);
fytot=sum(fy);
fr=sqrt(fxtot.^2+fytot.^2);
fra=atand(fytot/fxtot);
mat=[1:3;f;a];
```

Problem Example 1 - cont'd

```
clc
disp ('Resultant of 3 Concurrent Forces')
disp('')
disp('       Force      Angle')
fprintf('%i      %.1f      %.1f\n',mat)

fprintf('\nResultant force = %.1f and\n',fr)
fprintf('angle is %.1f degrees\n',fra)
```

Problem Example 1 - cont'd

```
>> Resultant3forcesLoadFile
Enter file name: data3fa.dat
Resultant of 3 Concurrent Forces
        Force       Angle
1       450.0       50.0
2       200.0       0.0
3       600.0       -75.0

Resultant force = 686.0 and
angle is -20.0 degrees
```

More practice see Problem A6

Thank You