Lesson 4: Input/Output

Topics:

- Obtaining input from user
- Displaying output
- Read and Write to file

MATLAB Commands:

1. input()
2. disp()
3. fprintf()
4. save()
5. load()

Book Sections: 3.1 - 3.4, 3.6

1. Input Command

MATLAB's input command provides a means to obtain input from the user.

```
% Numerical input
x=input('Prompt the user with some text: ');

% Vector input
x=input('Prompt: '); % e.g.,  [1,2,3]

% Character input
x=input('Prompt the user with some text: ','s');
```

2. Disp(lay) Command

MATLAB's disp() command provides a means to display output from the user or program. Note: disp() does not show the variable name.

```
% You should read the help file
help disp
```

```
  disp Display array.
     disp(X) displays array X without printing the array name or
     additional description information such as the size and class name.
     In all other ways it's the same as leaving the semicolon off an
     expression except that nothing is shown for empty arrays.

     If X is a string or character array, the text is displayed.

     See also fprintf, sprintf, int2str, num2str, rats, format, details.

     Reference page for disp
     Other functions named disp
```

```
disp(5);
```

```
     5
```

```
x=5;
disp(x);
```

```
     5
```

```
x='String';
disp(x);
```

```
 String
```

3. Format print

MATLAB's fprintf() command provides a means to display or write output from the user or program to the screen or a **file**.

```
% You should read the help file
help fprintf
```

```
 fprintf Write formatted data to text file.
    fprintf(FID, FORMAT, A, ...) applies the FORMAT to all elements of array A and
    any additional array arguments in column order, and writes the data to a text
    file.  FID is an integer file identifier.  Obtain FID from FOPEN, or set it to 1
    (for standard output, the screen) or 2 (standard error). fprintf uses the
    encoding scheme specified in the call to FOPEN.

    fprintf(FORMAT, A, ...) formats data and displays the results on the screen.

    COUNT = fprintf(...) returns the number of bytes that fprintf writes.

    FORMAT is a character vector that describes the format of the output fields, and
    can include combinations of the following:

       * Conversion specifications, which include a % character, a
```

```
% Useful syntax used in fprintf
% \t = Tab
% \n = New line
% %i = integer
% %#.#f = floating point

vol=10;
fprintf('The volume is %i\n',vol);
```

```
The volume is 10
```

```
% Show value of pi
fprintf('The value of pi is %4.5f\n',pi)
```

```
The value of pi is 3.14159
```

```
% For floating point numbers, the first
% number after the % symbol is the field
% size, the minimum number of characters
% to print.

% To see the difference, try the following:
```

```
fprintf('The value of pi is %1.4f\n',pi)
```

```
The value of pi is 3.1416
```

```
fprintf('The value of pi is %8.4f\n',pi)
```

```
The value of pi is   3.1416
```

```
fprintf('The value of pi is %12.4f\n',pi)
```

```
The value of pi is       3.1416
```

```
% The value after the decimal is the precision,
% that is the number of digits to display after
% the decimal (obvious).


% If the vector has multiple data in a row
% put the place-holders for each number.

x=[pi 2*pi];
fprintf(['The values of x are %1.3f and %1.3f \n '],x);
```

```
The values of x are 3.142 and 6.283
```

NOTE: the fprintf() function can be used to write to a file too, but we will see that later.

4. Saving and loading stored variables

Saving variables is often desired. Loading a saved variable is therefore also a needed operation. The two functions to handle these operations are **save** and **load**.

4.1 Saving workspace variables

```
% You should read the help file
help save
```

```
  save Save workspace variables to file.
     save(FILENAME) stores all variables from the current workspace in a
```

MATLAB formatted binary file (MAT-file) called FILENAME. Specify
FILENAME as a character vector or a string scalar. For example, specify
FILENAME as 'myFile.mat' or "myFile.mat".

save(FILENAME,VARIABLES) stores only the specified variables. Specify
FILENAME and VARIABLES as character vectors or string scalars.

save(FILENAME,'-struct',STRUCTNAME,FIELDNAMES) stores the fields of the
specified scalar structure as individual variables in the file. If you
include the optional FIELDNAMES, the **save** function stores only the
specified fields of the structure. You cannot specify VARIABLES and
the '-struct' keyword in the same call to **save**.

```
% Initialize some variables
a=1;
b=2;
c=3;

% Save a single variable by its name
save a    % command form
save('a') % function form
```

```
Error using save
Must be a string scalar or character vector.
```

```
% Save variable to a file; save(FILENAME,VARIABLE)
save('myVariable','a')  % function form
save myVariable a       % command form

% Save all workspace variables
% save(FILENAME)

% Save many workspace variables
% save(FILENAME,VARIABLES)

% NOTE: filename and variables need to be
%       in single quotes in function form
% ---------------------------------------- %
```

```
% Variables can be saved (and appended) to
% binary files (default), or ascii file types
% such as .txt or .dat.


% Example using function form and command form
% txt and dat file extensions.
save('abc.txt','a','b','c','-ascii')
save -ascii abc.dat a b c
```

4.1 Loading variables

Previously saved variables can be loaded in the event that workspace variables are cleared (e.g., shutdown/restart).

```
% Create variable
a=1;

% Display variable
disp(a);

% Save variable
save a

% Clear variable from workspace
clear a

% Check to be sure a no longer exists
whos

% Load variable back into the workspace
load a.mat

%
```

4.1.1 Appending data to file

```
% Append data to dat file

mat1=rand(2,3);
mat2=rand(3,3);
```

```
save randmatrices.dat mat1 -ascii

save randmatrices.dat mat2 -append -ascii
```

Return to MAT225