UNIVERSITY OF UTAH
ELECTRICAL AND COMPUTER ENGINEERING DEPARTMENT

ECE1020 **COMPUTING ASSIGNMENT 9**
N. E. COTTER **MATLAB® FILE I/O: MODULAR COMMUNICATION SYSTEM**

READING
Matlab® Student Version: learning Matlab 6, Appendix A-17
Mastering Matlab® 6, Ch 13

TOPICS
File and directory management. Reading and writing I/O files.

OVERVIEW
The next modification to our simulated communication system will allow us to process long sequences of commands for our rover and write the results to files that we can look at with a text editor (outside the Matlab command window).

When we deal with large sets of data, it is convenient to store that data in files that we process with programs. It is also convenient to write the results out to a file. This allows us to modify the input data and read the output data without having to change our program or print out variables from the Matlab window. It also means that our data is retained after we exit the Matlab program. Furthermore, we can share input data and results with others.

PROCEDURE
In this assignment you will use file Input/Output (I/O) to break your communication system into pieces representing the transmitter, noise source, and receiver. Each piece will read data from a file and write results to another file. The output of one piece will serve as input for the next piece.

+5 pts Command File for calculations
Using a text editor program on your PC, create a command file (a type of function file) called **comm_io.m** containing matlab commands to perform the calculations in this assignment.

+5 pts Start by copying your code from Computing Assignment 8 into comm_io.m. Then modify the code as described below. Remember that definitions for functions called from the main function must be at the bottom of the file.

+5 pts Do **not** use semicolons at the ends of commands in your files, and place new function definitions at the bottom of comm_io.m.

+15 pts Create rd_str_file function
Place the following line of code at the beginning of comm_io.m. (This code replaces the code that created command_str in Computing Assignment 8.) This code will call a function to read command strings for the rover from a file called "in_comm_str.txt".

```
command_str = rd_str_file('in_comm_str.txt')
```

Write code for the rd_str_file function as described by the following definition and comments:

```
function command_str_matrix = rd_str_file(filename)
% Read strings from file called filename.
%  File format: one string per row of file.
%  Strings in file are ascii with no quote marks.
%  Stores results in character matrix command_str_matrix, with each
%   (non-blank) row of the file saved as character string that is one
%   row of command_str_matrix.
```

Note that the variable names in the function definition may be slightly different than the names used when calling the function.
In the rd_str_file function, use the following Matlab I/O functions: fopen, fgetl, isstr, sscanf, and fclose.

Using Notepad or other text editor, create the file in_comm_str.txt with the following content (rover commands):

```
right
0
right
1
left
1
right
0
```

Note that there is a carriage return after the last zero in the file, the file is stored as "text only", and there is only one word on each line.

+10 pts Use save function
To provide practice using I/O functions, your code will create output files and read them back in as you proceed through the program.
Immediately after the code in your program that sets the value of command_code, use the "save" command to store command_code in a file called xmit_code_bits.mat.

+5 pts Clear command_code
To prove that the next step is reading in the command_code variable, clear the command_code variable by setting it to [] immediately after issuing the save command.

+10 pts Use load function
Immediately after clearing command_code, use the "load" command to read the contents of xmit_code_bits.mat back into command_code. (Later on, there will be code that adds noise to command_code before reading it back in.)

+10 pts Use dlmwrite function
Immediately after the code in your program that sets the value of command_recv, use the dlmwrite function to store command_recv in a file called noisy_xmit_data.txt
Note that there is no .m extension on this file name. Use a comma as the delimiter.

+5 pts Clear command_recv
To prove that the next step is reading in the command_recv variable, clear the command_recv variable by setting it to [] immediately after issuing the dlmwrite command.

+10 pts Use dlmread function
Immediately after clearing command_recv, use the dlmread function to read the contents of the file noisy_xmit_data.txt into command_recv. Use a comma as the delimiter.

+15 pts Create wr_str_file function
Place the following line of code immediately after the code in your program that sets the value of command_str_recv. If all goes well, this file, called out_comm_str.txt will be the same as the in_comm_str.txt with which we began the entire communication process.

```
wr_str_file('out_comm_str.txt', comm_str_recv)
```

Note that the format of out_comm_str.txt will be the same as that of in_comm_str.txt
If there are no communication errors it appears as follows:

```
right
0
right
1
left
1
right
0
```

Write code for the wr_str_file function as described by the following definition and comments:

```
function wr_str_file(filename, str_matrix)
% Write strings in str_matrix out to file called filename.
%  File format: one string per row of file.
%  Strings in file are ascii with no quote marks.
```

In the wr_str_file function, use the following Matlab I/O functions: fopen, fprintf, and fclose.

+5 pts Run Script File
Run your script file by typing the name of the file without the .m
>> comm_io
Use a diary file to capture the output of comm_io.
Verify that each function works properly and that the content of files written by your program matches the results computed within the program. (In other words, compare the data printed out as your Matlab program runs with the data written to each file.)

Note: If you make any changes in your **comm_io.m** file, be sure to run the following Matlab command to insure that Matlab reads your file again the next time you run it:
>> clear all

E-mail your file (comm_io.m) and your diary file to your TA, (as two separate e-mails). In the Subject line of your e-mail, be sure to put Your Name, "ECE1020 Comp9," and the file name, (e.g. comm_io.m). Also, print out the files and hand them in to the TA or to the ECE1020 locker.