EECE 231: Files — Reading and Writing

Reading: Bielajew, Section 9.4

Matlab help for file read/write functions

OBJECTIVES

- Learn the concept of a file as a basic unit of storage.
- Reading and writing files in Matlab.

OUTLINE

DATA AND FILES

- We handled so far data we input from the keyboard
  - Also data we generated randomly
- Data is generated and collected in several ways.
  - Sensors collecting samples of interesting quantities.
  - Manual entry of interesting records.
  - Computations of interesting entities.
  - Speech, image, and video recordings of interesting events.
- Typically stored in digital format in the form of *files* of a corresponding formats.
- C++ programs, Matlab functions, and Matlab scripts are stored in text files.

FILES

- A file is a sequence of bytes with a header that contains meta information about the file.
  - such as file name, size, check sum for error correction, . . .
- Files allow easier and more reliable handling, storage, and transfer of data.
- Files allow more than one program to handle the same data.

FILES AND DATA FORMATS

- Data in files can take several formats.
  - Text files hold sequences of ASCII or UNICODE characters and follow a format that allows humans to read them easily.
  - Image files hold sequences of binary values and follow formats that allow image viewers to display them easily.
  - Video files follow formats that allow video players to change image frames in a timely fashion for the human eye.
- Every file format comes with interfaces to read from it and write into it.

FILES, DIRECTORIES AND STORAGE

- Files are stored in permanent storage devices such as
  - Hard disks, flash memory cards, compact disks, and tapes
- Files are organized in directories for easier management.
  - Each directory is itself a file whose content points to the files inside it.
  - Each system has root directories where file storage starts.
  - Nowadays the location of a file could be on a disk on the local computer, on a remote server, or somewhere on a cloud based file server.

FILE LOCATION

- On a local computer, the file name needs to refer to its position relative to the root storage device of the computer.
  - or relative to an important directory such as the *current working directory*.
- On the internet/intranet the file name is a *Uniform resource locator* (URL) that uniquely identifies the file we are working with.

SEQUENTIAL ACCESS FILES

- A sequential access file is a file containing a sequence of bytes.
  - A text file is a sequential access file
- It might contain records, but records in it do not have a fixed width.
  - The end of a record must be encoded itself by a special character or a pattern.
  - You can not read record n without reading the $n-1$ records before it.
- You always have to read it one byte at a time.
  - It is terminated by a special end-of-file (EOF) character.
- Changing contents is not very efficient.
  - It often requires rewriting the whole file.

ADVANTAGES OF SEQUENTIAL ACCESS FILES

- **Important:** It is a very useful method to store records of data of varying record length.
  - Such data is typically all read into memory, manipulated, and then written out.

OUTLINE

INTERFACES FOR FILES IN MATLAB

- Matlab supports interfaces to read files sequentially or based on user defined formats.
- Matlab also supports direct reading of several file formats into matrices, and writing matrices into files.
  - Spreadsheet files (excel, calc, comma-separated-value files),
  - Image files (jpg, png, svg,. . .),
  - Video files (mpeg, awv, *ldots*)
- Plan: underlined topics

TOPICS

- Writing and reading workspaces
- Writing and reading excel files
- A simple function to read homogeneous files: *textread*
- *fprintf* function: write to files
- *fscanf* function: read from files
- We will study also the `fgetl` and `feof` functions

WRITING THE WORKSPACE

- The Matlab workspace includes the current alive Matlab variables and their values.
- The command `save('myfile.mat')` saves all workspace data to the binary file `myfile.mat`.
  - The file is saved in the current directory.
- Command `save('myfile.mat', 'data1', 'data2')` saves variables `data1` and `data2` to file `myfile.mat`.
- Command `save('myfile.mat', 'data1', '-ASCII', '-v6')`
  - Saves the file using the ASCII format.
  - Saves it in a format that is compatible with Matlab version 6.
- Command `save('myfile.mat', 'data3', '-append')`
  - Appends the data from `data3` to the file.

READING THE WORKSPACE

- The `load('myfile.mat')` command loads the data from file `myfile.mat` into the workspace.
- Command `load` uses the same interface as the `save` command.
  - It can take a list of variable names and selectively load only those.
  - It accepts format, version, and append flags as well.

READING/WRITING EXCEL FILES

- The `xlsread` command takes the name of an excel sheet file.
  - Loads the data from the excel sheet into a Matlab array.
  - For instance, `g = xlsread('myfile.xls');` reads the data from file `myfile.xls` into matrix *g* while ignoring the non-numeric data
- Similarly function `xlswrite` takes a file name and an array variable.
  - Writes the values in the array to an excel sheet.
  - `xlswrite('myfile.xls', g);` writes the data in matrix *g* into the excel sheet file `myfile.xls`.

textread FUNCTION

- A simple function to read homogeneous files
  $[A, B, C, ...] = textread('filename','format');$
- Example: consider the numeric data file *numData.txt*:

```
1   2.2
3   5.1
6   3.6
10   11.2
```

- ```
  >> [a,b]=textread('numData.txt','%d %f');
  >> disp(a') % display transpose for clarity
       1     3     6    10

  >> disp(b')
      2.2000    5.1000    3.6000   11.2000
  ```
- Rows in the output variables match the format parameters.
- Produces an error if there is a mismatch
  ```

MANIPULATION FILES SEQUENTIALLY

- First, open the file
- Function `fileID = fopen('filename','r');` opens a file for reading and stores file ID in *fileID*
- Function `fileID = fopen('filename','w');` opens a file for writing and stores file ID in *fileID*
- Every `fopen` call should be matched by an `fclose` call after the reading/writing is done.

SEQUENTIAL FILE WRITE: *fprintf*

- Function `fprintf` similar to `sprintf`

```
fprintf(fileID, format-string, format-parameter1,
                               format-parameter2,...)
```

  It prints a formatted string *format-string* to the file whose ID is *fileID*

- Example:

```
s = 'bread';
price = 1.376;
f = fopen('newFile.txt','w');
fprintf(f, 'price of %s is %.2f today', s, price);
fprintf(f, '\n **');
fclose(f);
```

- Above code creates a new file called "newFile.txt" consisting of:

```
price of bread is 1.38 today
**
```

- Note: to see the new line on windows use *notepad* $++$ or add $\backslash r$ before $\backslash n$

SEQUENTIAL FILE READ: *fscanf*, *fgetl*, AND *feof* FUNCTIONS

- Function `fscanf`:
  $[A, count] = fscanf(fileID, format\text{-}string, size)$
  - *format-string* similar to that of *sprintf*
  - Input *size* is the number of elements to be read
  - Reads from file according to format strings and stores read data in *A*
  - Returns also the number *count* of read elements
- Function `fgetl(fileID)` reads and returns a line from file as a string
- Function `feof(fileID)` returns *true* if past the end of file and *false* otherwise

SEQUENTIAL FILE READ EXAMPLE

- Consider the file 'inputData.txt':

```
% table
time  value
  1    1.1
  2    2.2
  5    6.0
  6    5.0
  10   4.0
  11   8.1
  12   4.1
average value: 4.3571
```

SEQUENTIAL FILE READ EXAMPLE (CONTINUED)

- First read the comment in the header and then the 2 labels:

```
f = fopen('inputData.txt','r'); % Open file for reading

comment = fgetl(f); % Read first line
disp(comment)

[s1 count]= fscanf(f,'%s',1);
% fscanf reads the string 'time' and stores it in s1
%  1 specifies the number of strings to be read
% fscanf  also sets count to the number  of values read,
%  i.e., it sets count = 1 unless  the file consists of
%  one line (the one  read above), in which case count
%  takes the value  zero.

disp(s1)

[s1 count]= fscanf(f,'%s',1);
% fscanf reads the string 'value' and it stores it in  s2

disp(s2)
```

SEQUENTIAL FILE READ EXAMPLE (CONTINUED)

- Now read the numbers on row at a time and store them in the matrix T:

```
i=1;
while ~feof(f),  % feof(f) returns 1 if past end of file
    % The loop reads one row at each iteration
    [row count]=fscanf(f,'%d %f', [1 2]);
    % The 2 in [1 2] is the row length
    %   and 1 is the number of rows
    % fscanf stores the read row in the variable row
    % It also sets count to the number of values read
    % Thus,  count = 2 when  reading the table
    %   and count = 0 when the last line is reached
    %  (as it does not start with with an integer)

    if (count < 2) break; end;
    if i ==1, T = row;
    else  T= [T' row']';  % Save the row  in the table T
    end;
    i=i+1;
end;
disp(T)
```

SEQUENTIAL FILE READ EXAMPLE (CONTINUED)

- Fianlly read the average:

```
if ~feof(f),
  [avg count]=fscanf(f,'average value: %f',1);
  % Read the average

  disp(avg)
end;
fclose(f);
```

- Note that on the file "inputData.txt", the end of file will not be reached in the while loop (the break will stop the while loop).

EECE 231: FILES — READING AND WRITING

READING: BIELAJEW, SECTION 9.4

MATLAB HELP FOR FILE READ/WRITE FUNCTIONS