W1005
Intro to CS and Programming in MATLAB

Input / Output

Fall 2014
Instructor: Ilia Vovsha

http://www.cs.columbia.edu/~vovsha/w1005

Outline

- Input / Output Streams
- Loading / Saving data
- Formats
  - MATLAB data
  - Text files
  - Images

I/O (streams)

- A *stream* is a sequence of characters used for program input or output
- There are *standard* (default) streams (input, output, error)
- Typically print to the screen by default

sprintf()

- sprintf() converts (formats) data into a string:
  - Example: S = sprintf('myfile%d', 2);
  - The character following % specifies the format into which the supplied parameter is converted
  - Can specify different formats and supply multiple parameters
  - Example: S = sprintf('myfile%d%s', 2, 'test');
- Common formats: %d (integer) %s (string) %g (float)
- Example:

1. prefix = 'home_dir/mydata';
2. file_num = 2;
3. filename = sprintf('%s%d.txt', prefix, file_num);

I/O (formats)

- format <type> :
  - Controls output formats, and has no effect on computation
  - For better appearance of numbers with many decimal digits use 'format short g'
  - To remove annoying spaces (line-feeds) use 'format compact'
  - Possible to specify multiple formats as long as they don't contradict each other

I/O (user Input)

- input() is a command that prompts the user to enter some input
- To record the input, you need to supply a variable
- The input stream is closed when the user hits enter
- By default, the expected input is a MATLAB expression
- Examples:
  - color = input('Enter your choice');
  - color = input('Enter your choice', 's');
  - color = input('Enter your choice\n', 's');

I/O (MATLAB data)

- MATLAB has its own file extension ".mat"
- A .mat file is stored in binary format (unlike say text files it is pointless to view a .mat file)
- Storing data in a .mat file is convenient:
  - Can load and save data using built-in commands
  - Can load/save specific variables to/from the workspace
- Examples (load):
  - load file1; % Automatically checks for file1.mat and loads all vars
  - v1 = load('file1.mat', 'X'); % Loads the variable X from file1.mat

I/O (MATLAB data)

- Examples (save):
  - save file1.mat X; % Save variable X to file1.mat
  - save(filename, 'X'); % Save X to a file whose name is stored in the variable 'filename'
  - save file1.mat X Y Z; % Save X, Y, Z to file1.mat

Load / Save

- When multiple variables are saved in a .mat file, and then loaded into a single variable, they are saved as fields of a struct
- Example:

```
1. save file1.mat X Y Z;
2. S = load('file1');
3. isstruct(S);      % Struct with 3 fields: S.X, S.Y, S.Z
```

I/O (text files)

- Commands: fopen(), fclose(), fgetl(), fseek(), fprintf()
- More complex than loading .mat files. First need to open a file fopen(), then read/write the data fgetl(), fprintf(), then close the file fclose()
- To open a file, we create a 'file identifier':
  - FID = fopen('myfile.text', 'r'); % 2nd argument specifies operation
  - 'r' (read) , 'w' (write), 'a' (append)
  - In case of failure, FID = -1
  - Always check whether file was opened

I/O (text files)

- To close a file is easy:
  - FID = fopen('myfile.text', 'r');
  - SOPEN = fclose(fid); % returns -1 if failed to close
- Read a line from a file:
  - myline = fgetl(FID); % Read a single line
- Rewind file:
  - fseek(fid,0,-1); % Often necessary to traverse a file multiple times
- Write to a file:
  - fprintf() writes formatted data to file
  - fprintf(fid, FORMAT, ARRAY); % Format is a string

I/O (delimited data)

- Typically files have a specific format: numeric data separated by some delimiter (a character)
- MATLAB has built-in functions: dlmread(), dlmwrite()
- Examples:
  - M = dlmread('file1.txt', '\t') % Read tab delimited data into M
  - Choose any character(s) as your delimiter. Be wary of spaces or dots as delimiters
  - M = dlmread('file1.txt', ';' , [R1 C1 R2 C2])

    % 3rd parameter is the range (zero-based) from which the data is read
  - dlmwrite('file1.txt', M, '\t') % Write M to a tab-delimited file

I/O (images)

- Image are just matrices. Color images have an additional dimension (B,G,R values). Grayscale images just have pixel values
- Built-in commands: imread(), imwrite()
- Examples:
  - M = imread('myphoto.jpg'); % Read an image
  - imwrite(M, 'myphoto.jpg'); % Infers the format from the ext.
  - imwrite(M, 'myphoto', 'JPEG');

Exercise (in class)

- Loading and saving a sequence of files.
- Suppose you have a directory ‘home/mydata’ which contains 100 files all named ‘dataset_#.mat’. Each files stores a matrix M. Write a function that takes an input parameter N, and opens each file (from 1 to N), sorts the matrix M along rows, and save the sorted matrix to a new file named ‘newdata_#.mat’

Loading Data

- Three step process:
  1. Identify the type of data you have
  2. Find the relevant function(s)
  3. Choose the appropriate format
- Example: “Each file has strictly numeric data separated by delimiter…range of data is specified”
  - Step 2: dlmread(), dlmwrite()
  - Step 3: “RESULT = DLMREAD(FILENAME,DELIMITER,RANGE)”
  - Reads the range specified by RANGE = [R1 C1 R2 C2]

textscan()

- Read formatted data from a text file
- textread() is another (*deprecated*) function with similar functionality
- How does it work?
  - *C* = textscan (*FID*, '*format*')
  - Recall that to open a text file, we need to create an identifier:

    FID = fopen ('myfile.text', 'r');
  - *'format'* is a string specifying how each line should be read
  - 'C' is a cell array. The number of specifiers (*'format'*) determines the number of cells