Lecture 6
Advanced MATLAB: Data and File Management

Matthew J. Zahr

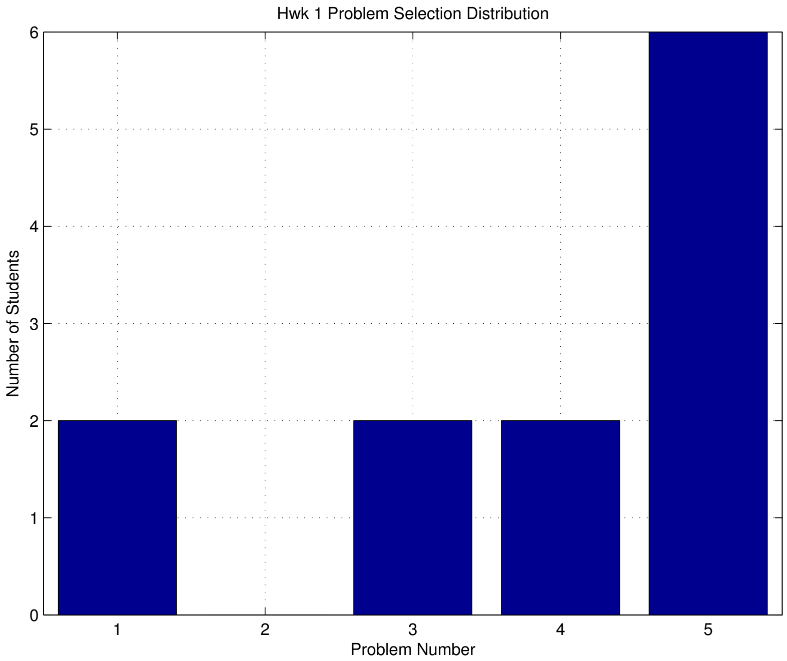
CME 292
Advanced MATLAB for Scientific Computing
Stanford University

9th October 2014

Announcements

- Homework 1 solutions to be posted tomorrow (on Coursework)
- Homework 2 due today
    - Email me if you need an extension
- Homework 3 uploaded; Due next Tuesday
- Lecture 6 of 9

Homework 1 Problem Selection



Outline

MATLAB Search Path

- *Search path* is a subset of all folders on the file system that MATLAB uses to efficiently locate files used with MathWorks products. All files in the folders on search path can be accessed by MATLAB.
- *Order* of folders on search path is important
    - When files in different folders (both in search path) with same filename exist, MATLAB uses the one in the folder nearest the top of the search path
    - Search path includes
        - Folders provided with MATLAB and other MathWorks products
        - MATLAB `userpath` (first on search path above folders supplied by MathWorks)
    - View entire *ordered* search path: `path`

Search Path Commands

Command	Description
`addpath`	Add folders to search path
`rmpath`	Remove folders from search path
`path`	View or change search path
`savepath`	Save current search path
`userpath`	View or change user portion of search path
`genpath`	Generate path string
`pathsep`	Search path separator for current platform

Outline

Text File Commands

Command	Description
`importdata`	Load data from file
`dlmread`	Read ASCII-delimited file of numeric data into matrix
`dlmwrite`	Write matrix to ASCII-delimited file
`textscan`	Read formatted data from text file or string
`type`	Display contents of file

`dlmread`

- Reads numeric data from the ASCII delimited file
  - `RESULT = dlmread(FILENAME)`
    - Delimiter *inferred* from file format
  - `RESULT = dlmread(FILENAME,DELIMITER)`
    - Delimiter specified by string `DELIMITER` (tabs are `'\t'`)
  - `RESULT = dlmread(FILENAME,DELIMITER,R,C)`
    - `R`, `C` specify the row/column in file of upper left corner of data (zero-based)
  - `RESULT = dlmread(FILENAME,DELIMITER,RANGE)`
    - `RANGE = [R1,C1,R2,C2]` specifies upper left and lower right corners of data (zero-based)
- When a delimiter is inferred from the formatting of the file, consecutive whitespaces are treated as a single delimiter. By contrast, if a delimiter is specified by the DELIMITER input, any repeated delimiter character is treated as a separate delimiter.

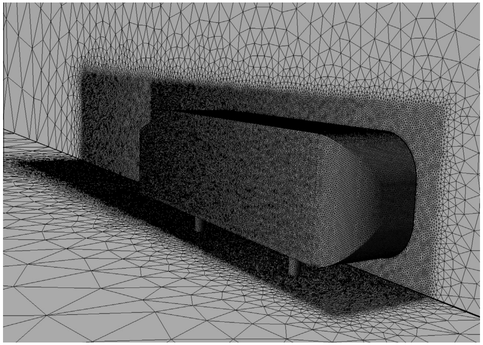
dlmwrite

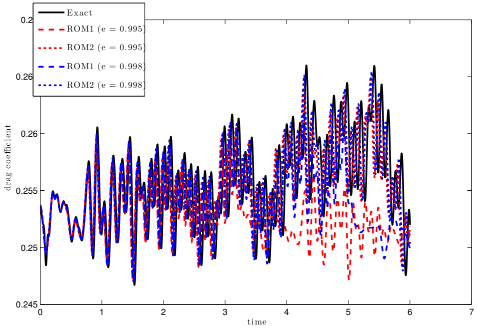
- Write numeric data in delimited format to ASCII file
  - `dlmwrite(FILENAME,M)`
    - Write matrix `M` to file, delimited by `,`
    - If `FILENAME` exists, it will be overwritten
  - `dlmwrite(FILENAME,M,DELIMITER)`
    - Delimiter specified by string `DELIMITER` (tabs are `'\t'`)
  - `dlmwrite(FILENAME,M,DELIMITER,R,C)`
    - `R`, `C` specify the row/column in file of upper left corner of data (zero-based)
- Force `dlmwrite` to append to existing file by using the `'-append'` flag
- Additional attributes that given to `dlmwrite` that will alter the format of the ASCII file
  - `'delimiter'`, `'newline'`, `'roffset'`, `'coffset'`, `'precision'`

Assignment

- Use `dlmread` to read the data from all 5 files in the directory `liftdrag`
- Each file contains the lift-drag history for the flow around the Ahmed body, a bench problem in the automotive industry
- The time history in each file corresponds to a different method of solving the CFD problem
- On the same axes, plot the *fifth* column of each file vs. the *second* column
    - This is the drag history (drag vs. time)

Assignment




Exact
ROM1 (e = 0.995)
ROM2 (e = 0.995)
ROM1 (e = 0.998)
ROM2 (e = 0.998)
drag coefficient
time
0.2
0.26
0.26
0.255
0.25
0.245
0
1
2
3
4
5
6
7

Low-Level File Commands

Command	Description
`fclose`	Close one or all open files
`feof`	Test for end of file
`ferror`	Information about file IO errors
`fgetl`	Read line from file, remove newline character
`fgets`	Read line from file, keep newline character
`fileread`	Read contents of file into string
`fopen`	Open file
`textscan`	Read formatted data from text file or string

Low-Level File Commands

Command	Description
`fprintf`	Write data to text file
`fread`	Read data from binary file
`frewind`	Move file position indicator to beginning of open file
`fscanf`	Read data from text file
`fseek`	Move to specified position in file
`ftell`	Position in open file
`fwrite`	Write data to binary file

Open/Close File

- `FID = fopen(FNAME)`
  - Opens the file `FNAME`
  - `FID` is a scalar integer valued double, called a file identifier
  - Use `FID` as the first argument to other file IO routines
  - If `fopen` cannot open the file, it returns -1
- `FID = fopen(FNAME,PERMISSION)`
  - Opens the file `FNAME` in the mode specified by `PERMISSION`
    - open for reading (`r`), writing (`w`), appending (`a`) - create if file does not exist
    - open for reading (`r+`), writing (`w+`), appending (`a+`) - do not create file
- `ST = fclose(FID)`
  - Closes the file associated with file identifier `FID`, obtained from `fopen`
  - `fclose('all')` closes all open files except standard input, output, and error

Read line from file (`fgetl`, `fgets`)

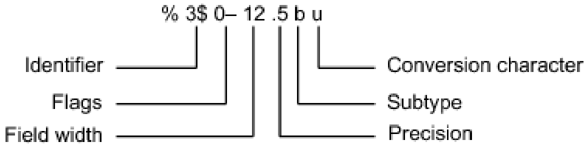
- `TLINE = fgetl(FID)`
  - Returns the next line of a file associated with file identifier `FID` as a MATLAB string (identifier incremented)
  - Line terminator is NOT included
- `TLINE = fgets(FID)`
  - Same as `fgetl` with line terminator included

```
fid=fopen('lec06_ex.m');
  while 1
    tline = fgetl(fid);
    if ¬ischar(tline), break, end
      fprintf(tline)
  end
fclose(fid);
```

- What happens with `fgetl` replaced with `fgets` in the above code?

Write to text file (`fprintf`)

- `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 with file identifier `FID` from `fopen`
  - Set `FID` to 1 to print to the screen (or exclude)



```
>> year = 8/3;
>> fprintf('%f year at Stanford\n',year)
2.666667 year at Stanford
>> fprintf('%20.6f year at Stanford\n',year)
            2.666667 year at Stanford
>> fprintf('%20.15f year at Stanford\n',year)
   2.666666666666667 year at Stanford
```

Conversion Characters (fprintf)

Conversion	Value Type
%d,%i	Signed integer
%u	Unsigned integer
%f	Floating point, fixed notation
%e	Floating point, exponential notation (e)
%E	Same as e using E
%g	Compact form of e
%G	Compact form of E
%c	Single character
%s	String of characters

```
>> v = 0.333;
>> fprintf('%f,%e,%E,%g,%G\n',v,v,v,v,v)
0.333000,3.330000e-01,3.330000E-01,0.333,0.333
```

Special Characters (`fprintf`)

Character	Meaning
`''`	Single quote (`'`)
`%%`	Percent sign (`%`)
`\n`	Newline
`\t`	Tab

```
>> fprintf('''Example''\t1%%\n')
'Example' 1%
>> fprintf('Example 2')
Example 2>>
```

feof, ftell, frewind, fseek

Consider the command `FID = fopen(FNAME)`. Then,

- `ftell(FID)` returns the *position* in the file
- `fseek(FID,OFFSET,ORIGIN)` repositions the file position indicator to the byte with the specified `OFFSET` relative to `ORIGIN`
- `frewind(FID)` resets `FID` to the beginning of the file `FNAME`
- `feof(FID)` returns true if end-of-file indicator has been set

Demo: `lec06_ex.m`

Writing/reading binary files (`fwrite`, `fread`)

- `count = fwrite(FID,A)`
  - Writes the elements of matrix `A` to the specified file
  - The data are written in column order
  - `COUNT` is the number of elements successfully written.
- `A = fread(FID)`
  - Reads binary data from the specified file and writes it into matrix `A`
  - Reads the entire file and positions the file pointer at the end of the file
- `A = fread(FID,SIZE)`
  - Reads the number of elements specified by `SIZE`
  - Valid entries for `SIZE` are:
    - `N` - read N elements into a column vector
    - `inf` - read to the end of the file
    - `[M,N]` - read elements to fill an M-by-N matrix, in column order (N can be inf, but M can't)

textscan

- Read formatted data from text file or string
  - C = `textscan(FID,'FORMAT',N)`
    - Reads data from the file, using the `FORMAT` (recall conversion characters: `%u`, `%i`, `%u`, `%f`, `%e`, `%E`, `%g`, `%G`, `%c`, `%s`) `N` times, where `N` is a positive integer
    - To read additional data from the file after `N` cycles, call `textscan` again using the original `FID`
    - Useful when format of file not uniform through the end of the file

Example

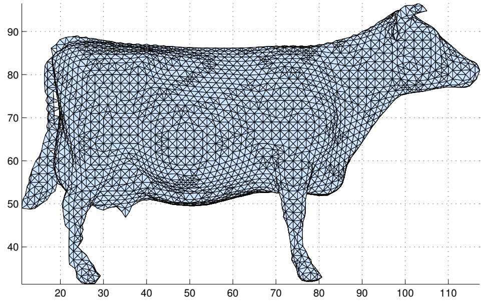
Node/element files from UC Berkeley Computer Graphics group. First line of each file contains header information (number of nodes/elements, etc). Nodes contained in columns 2 - 4 for nodes file. Elements contained in columns 2 - 5 of elements file.

```
% UC Berkeley Graphics group mesh format
fname = 'meshes/dragon';

fid = fopen([fname,'.node']); fgetl(fid);
nodes  = textscan(fid,'%d %f %f %f %d');
p = [nodes{2:end-1}]; fclose(fid);

fid = fopen([fname,'.ele']); fgetl(fid);
elems  = textscan(fid,'%d %d %d %d %d');
t = [elems{2:end}]; fclose(fid);
```

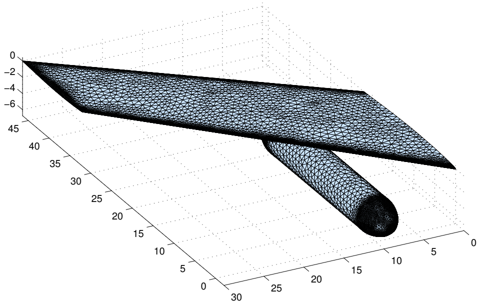
Assignment

- Extract the (surface) mesh in the FRG format (`'meshes/AGARDwtt.top'`) into two matrices
  - `p` - $n_v \times 3$ matrix contain xyz coordinates of each node
  - `t` - $n_e \times 3$ containing node numbers of each element comprising a triangle
- FRG file format
  - Line 1: Node header (ignore)
  - Lines 2 - $n_v + 1$: [node number, x-coord, y-coord, z-coord]
  - Line n_v+2: Surface element header (ignore)
  - Line n_v+3 - $n_v + n_e + 3$: Last 3 entries per row contain node number of given triangle
- Use `simpplot(p,t)` to plot the mesh. What is it?

Example/Assignment



(a) Dragon mesh - UCB



(b) Agard wing with tank - FRG

Spreadsheet Commands

Command	Description
`xlsfinfo`	Determine if file contains Microsoft Excel spreadsheet
`xlsread`	Read Microsoft Excel spreadsheet file
`xlswrite`	Write Microsoft Excel spreadsheet file

Read from Spreadsheets

- `[NUM,TXT,RAW]=xlsread(FILE,SHEET,RANGE)`
  - Reads the data specified in `RANGE` from the worksheet `SHEET`, in the Excel file specified in `FILE`.
  - The full functionality of `xlsread` depends on the ability to start Excel as a COM server from MATLAB.
- `[NUM,TXT,RAW]=xlsread(FILE,SHEET,RANGE,'basic')`
  - Uses basic input mode. This is the mode used on UNIX platforms as well as on Windows when Excel is not available as a COM server.
  - In this mode, `xlsread` does not use Excel as a COM server, which limits import ability.
  - Without Excel as a COM server, `RANGE` will be ignored and, consequently, the whole active range of a sheet will be imported.
  - Also, in basic mode, `SHEET` is case-sensitive and must be a string.

Write Spreadsheets

- `[STAT, MSG] = xlswrite(FNAME, M, SHEET, RANGE)`
  - Writes the data in matrix `M` to the file `FNAME` in the sheet specified by `SHEET` to the range of cells specified by `RANGE`
  - `SHEET` can be numeric specifying worksheet index or quoted string
  - `RANGE` is of the form `'X:Y'` where `X` indicates the upper left corner of the writable range and `Y` is the lower right corner (i.e. `'B2:D4'` is the 3×3 block of cells from row `B` to `D` and columns `2` to `4`)
- Requires ability to use Excel as COM server; otherwise, saves to CSV file

Image IO Commands

Command	Description
`imfinfo`	Information about graphics file
`imread`	Read image from graphics file
`imwrite`	Write image to graphics file

- `A = imread(FILENAME,FMT)`
  - Reads a grayscale or color image from the file specified by the string `FILENAME` in the `FMT` format
- `imwrite(A,FILENAME,FMT)`
  - Writes the image `A` to the file specified by `FILENAME` in the format specified by `FMT`
  - For grayscale, `A` is $m \times n$
  - For colorscale, `A` is $m \times n \times 3$
- Example: `get_rgb.m` from Homework 2
- Demo: `lec06_ex.m`

Outline

Operating System Commands

Command	Description
`clipboard`	Copy/paste strings to/from sys clipboard
`computer`	Information about computer
`dos`	Execute DOS command and return output
`getenv`	Environment variable
`perl`	Call Perl script
`setenv`	Set environment variable
`system`	Execute operating system command and return output
`unix`	Execute UNIX command and return output
`winqueryreg`	Item from Windows registry
`bang (!)`	Shell escape

System Calls

- Power of operating system available inside MATLAB
- Given stand-alone C/C++/Fortran code with files defining inputs and outputs
    - Ability to call executable from within MATLAB
    - Use MATLAB's file management to write input files and read output files
    - Provides *non-intrusive* alternative to integrating stand-alone code with MATLAB via MEX interface (Lecture 7)
    - Example: *PDE-constrained optimization*
        - Given executable that solves some PDE given text file input file and writes solution to binary files
        - Write optimization functions (objective, constraints, derivatives) that: write input files, use `system` to call executable, read binary outputs, and evaluate function
        - Call `fmincon` with optimization functions

Systems Calls - Syntax

- `[status,result] = SYSTEM('command')`
    - Calls upon the operating system to execute the given command. The resulting status and standard output are returned.

System Calls - Demo

- Demo: SDESIGN
- Opportunity to dig deeper in homework