# 13-Writing Data

text: Chapter 4.4-4.5

ECEGR 101

Engineering Problem Solving with Matlab

Professor Henry Louie


# Overview

- Writing Data to .txt
- Writing Data to .xls and .csv
- Writing Data to .mat


# Writing data to a file

To write data to a file

1. Open the file (fopen).
2. Write the variables to the file (fprintf).
3. Close the file (fclose).


# fopen

- fid = fopen(FILENAME, PERMISSION)

  FILENAME: name of the file to open, including extension (e.g. .txt). File must be in current directory. You can also specify the path to the file.

  PERMISSION (optional): opens FILENAME in the following mode

  'r' open file for reading

  'w' open file for writing; discard existing contents

  'a' open or create file for writing; append data to end of file

  See help fopen for more details


# fprintf

- fprint can be used to write data to a file
- Include fid as the first input

```
fprintf(fid, 'text %-12.5f more text', var)
```

file identifier


# fclose

To close the file after writing use fclose(fid)


# Example

Generate the square roots, squares, and cubes of all integers between 1 and 10. Save the output to a file as a table of data.

# Example

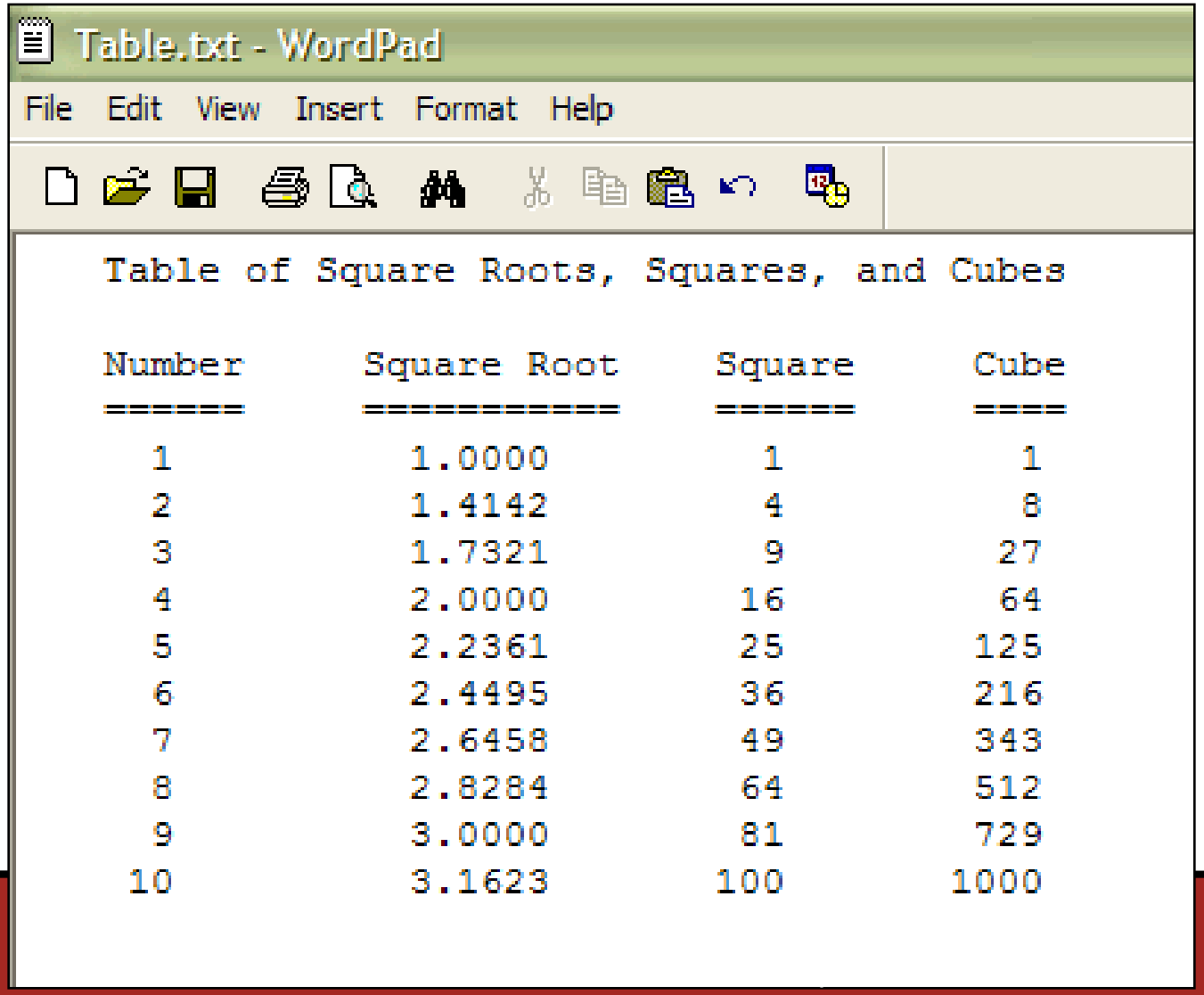
```
% Script file: table.m
%
% Creates a table of square roots, squares, and cubes.
%

% Generate the required data
x = 1:10;
square_root = sqrt(x);
square = x.^2;
cube = x.^3;

% Open a file for writing
fid = fopen('Table.txt', 'w');

% Print the title of the table
fprintf(fid, '  Table of Square Roots, Squares, and Cubes\n\n');
```

# Example

```
% Print column headings
fprintf(fid, ' Number    Square Root    Square    Cube\n');
fprintf(fid, ' ======    ===========    ======    ====\n');

% Create the output array
out = [x; square_root; square; cube];

% Print the data
fprintf(fid, ' %3d    %11.4f    %6d    %8d\n', out);

% Close the file
fclose(fid);
```

Important:
First row in - **out** as the first column in file
Second row in - **out** as the second column
:

# Example




Open the text files with WordPad instead of Notepad.


# Exercise

Write a MATLAB script to generate a table containing the sine and cosine of q for q between 0° and 90°, in 1° increments. The program should properly label each of the columns in the table. Save the table into a text file. Open the text file to make sure that you're getting a properly formatted table.

Note: you'll need to use WordPad to open the file (do not use Notepad).


# Exercise

```
% Generate a table of sine and cosine values
clear all
% Generate the angle values
theta = 0:90;
% Compute the sine
sinOUT = sin(theta*pi/180);
% Compute the cosine
cosOUT = cos(theta*pi/180);
% generate one variable with all the results
out = [theta ; sinOUT ; cosOUT];
% Open the file
fileID = fopen('Sin_Cos.txt', 'w');
% Save the results
fprintf(fileID, 'Theta       Sin        Cos\n\n');
fprintf(fileID, '%3d        %6.4f      %6.4f\n', out);
% Close the file
fclose(fileID);
```


# Exercise

| Theta | Sin | Cos |
|---|---|---|
| 0 | 0.0000 | 1.0000 |
| 1 | 0.0175 | 0.9998 |
| 2 | 0.0349 | 0.9994 |
| 3 | 0.0523 | 0.9986 |
| 4 | 0.0698 | 0.9976 |
| 5 | 0.0872 | 0.9962 |
| 6 | 0.1045 | 0.9945 |
| 7 | 0.1219 | 0.9925 |
| 8 | 0.1392 | 0.9903 |
| 9 | 0.1564 | 0.9877 |
| 10 | 0.1736 | 0.9848 |
| 11 | 0.1908 | 0.9816 |
| 12 | 0.2079 | 0.9781 |
| 13 | 0.2250 | 0.9744 |
| 14 | 0.2419 | 0.9703 |
| 15 | 0.2588 | 0.9659 |
| 16 | 0.2756 | 0.9613 |
| 17 | 0.2924 | 0.9563 |

...

# Exercise

Write a MATLAB script that asks the user to input a weight in pounds (lbs) and computes the equivalent weight in kilograms (kg). Display the input and the output on the screen as well as save it in a file as follows

(value in pounds) lbs = (value in kg) kg

For instance: **1 lbs = 0.45359237 kg**

Do not erase the file: every time you run the program, the output should be appended at the end of the tile. Open the file to make sure that it is correct.

# Exercise

```
% Exercise: Convert pounds into kilograms
% Ask the user for input
clc
pounds = input('Enter pounds:\n');
% Convert pounds to kilograms
kilograms= pounds/2.2;
% Open the output file
fid = fopen('Pounds2kilograms.txt', 'a');
% Print the result on the screen
fprintf('%4.2f pounds = %4.2f kilograms\n', pounds, kilograms);
% Save the same result in the file
fprintf(fid, '%4.2f pounds = %4.2f kilograms\n', pounds, kilograms);
% Close the file
fclose(fid);
```

Notice the permission 'a'


# Exercise

100.00 pounds = 45.45 kilograms
1.00 pounds = 0.45 kilograms
5.00 pounds = 2.27 kilograms
90.00 pounds = 40.91 kilograms
345.00 pounds = 156.82 kilograms
700.00 pounds = 318.18 kilograms
556.00 pounds = 252.73 kilograms
9.00 pounds = 4.09 kilograms
10.00 pounds = 4.55 kilograms
100.00 pounds = 45.45 kilograms
45.00 pounds = 20.45 kilograms
145.00 pounds = 65.91 kilograms
23.00 pounds = 10.45 kilograms

# Writing Data to .csv, .xls

- Data can be read from .csv, .xls, .xlsx files
- [data]=xlswrite(FILE, Variable, Sheet, RANGE)
  - data: variable containing numeric values
  - FILE: name of file, must include extension
  - Variable: the variable to write
  - SHEET: worksheet in the file to be written to (optional)
  - RANGE: range of data in the worksheet to be written to (optional)

Make sure the file you want to write to is closed

# Example

- Create a 10 by 10 matrix of random integers (use the randi command), and save the data to an Excel file named "RandomNumbers" in Sheet 2. Specify the range to be B2:K12

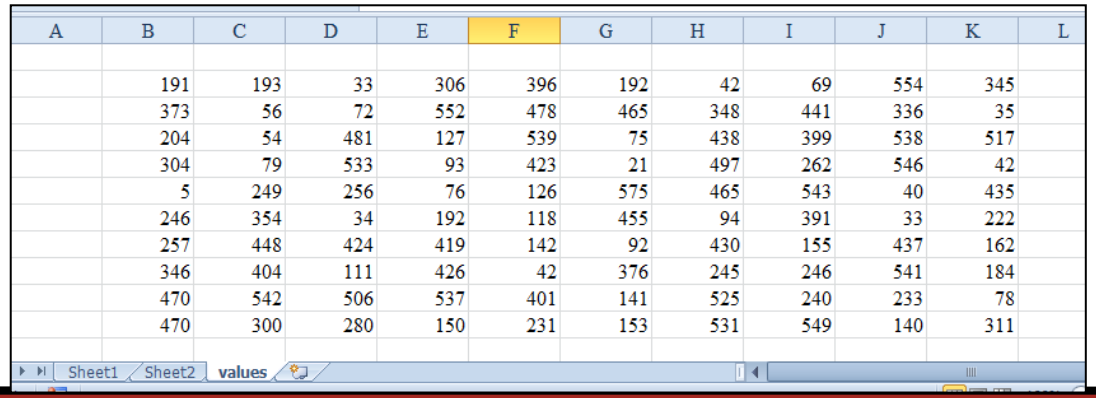
# Example

- Create a 10 by 10 matrix of random integers whose values are between 1 and 600 (use the randi command), and save the data to an Excel file named "RandomNumbers". Specify the range to be B2:K11, and the sheet to be "values"

# Example

```
>> numbers = randi(600,10)
numbers =
   191   193    33   306   396   192    42    69   554   345
   373    56    72   552   478   465   348   441   336    35
   204    54   481   127   539    75   438   399   538   517
   304    79   533    93   423    21   497   262   546    42
     5   249   256    76   126   575   465   543    40   435
   246   354    34   192   118   455    94   391    33   222
   257   448   424   419   142    92   430   155   437   162
   346   404   111   426    42   376   245   246   541   184
   470   542   506   537   401   141   525   240   233    78
   470   300   280   150   231   153   531   549   140   311
>> xlswrite('RandomNumbers.xls',numbers,'values', 'B2:K11')
Warning: Added specified worksheet.
> In xlswrite>activate_sheet at 284
  In xlswrite>ExecuteWrite at 256
  In xlswrite at 214
```

MATLAB created the sheet for us.



| A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | |
| | 191 | 193 | 33 | 306 | 396 | 192 | 42 | 69 | 554 | 345 | |
| | 373 | 56 | 72 | 552 | 478 | 465 | 348 | 441 | 336 | 35 | |
| | 204 | 54 | 481 | 127 | 539 | 75 | 438 | 399 | 538 | 517 | |
| | 304 | 79 | 533 | 93 | 423 | 21 | 497 | 262 | 546 | 42 | |
| | 5 | 249 | 256 | 76 | 126 | 575 | 465 | 543 | 40 | 435 | |
| | 246 | 354 | 34 | 192 | 118 | 455 | 94 | 391 | 33 | 222 | |
| | 257 | 448 | 424 | 419 | 142 | 92 | 430 | 155 | 437 | 162 | |
| | 346 | 404 | 111 | 426 | 42 | 376 | 245 | 246 | 541 | 184 | |
| | 470 | 542 | 506 | 537 | 401 | 141 | 525 | 240 | 233 | 78 | |
| | 470 | 300 | 280 | 150 | 231 | 153 | 531 | 549 | 140 | 311 | |

Sheet1 Sheet2 values


# Writing Data to .mat

- Variables and their content can be retrieved in the .mat format
- "save FILE" retrieves all variables saved in FILE
- Possible to specify particular variables (i.e. not all of the variables) to be retrieved from the FILE
- Also possible use a variable as the file name


# Example

- Create a vector of 100 Normally-distributed random variables with zero mean and standard deviation of one. Sort the vector in ascending order and save the sorted vector as "sorted" under the filename "sortedvalues". You will have to use the "sort" command.

# Example

only "sorted" is saved

```
>> unsorted=randn(100,1);
>> sorted=sort(unsorted);
>> save sortedvalues sorted
>> whos
  Name             Size             Bytes  Class     Attributes

  sorted         100x1                800  double
  unsorted       100x1                800  double

>> clear all; whos
>> load sortedvalues
>> who

Your variables are:

sorted
```

variables are cleared

sorted is loaded