University of Nottingham

Computer Programming with MATLAB

MM1CPM - Lecture 9

Advanced Input/Output

Dr. Roderick MacKenzie
roderick.mackenzie@nottingham.ac.uk
Autumn 2014

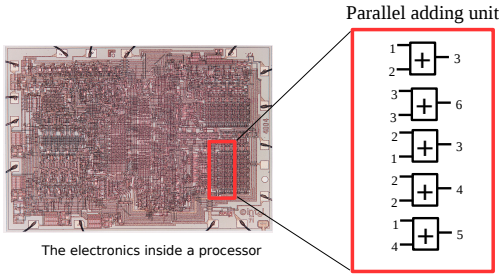
www.mm1cpm.com

Overview

- **Recap of last lecture**
  - Parallel computing
  - Functions
- Revision of input and output from the computer
- Advanced input and output to files

Recap: Parallel computing

We learnt about parallel computing, and how it can speed up our code.



Recap: Parallel computing in MATLAB

Non-parallel code:

```
a=1+1
b=1+2
c=2+2
d=4+4
e=7+7
f=1+2
g=3+3
h=1+1
i=1+1
j=1+2
k=2+2
l=4+4
.....
```

Parallel code:

```
x=[1 1 2 4 7 1 3 1 1 1 2 4]
y=[1 2 2 4 7 2 3 1 1 2 2 4]
z=x+y
```

- When ever you do matrix operations in MATLAB the computer uses parallel hardware to do them.
- They are therefore faster.

tic and **toc** commands.

Recap: Functions - input

- Information is passed to your function using arguments (the variables in brackets)

```
>z =our_function(1,2,3,4)
   z=2.5
```

```
function y =our_function(a,b,c,d)

   y=(a+b+c+d)/4

end
```


Recap : Functions - output

```
>y =our_function(1,2,3,4)
   y=2.5
```

```
function y =our_function(a,b,c,d)

   y=(a+b+c+d)/4

end
```

Overview

•Recap of last lecture

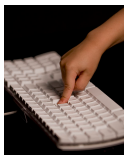
- Parallel computing
- Functions

•**Revision of input and output from the computer**

•Advanced input and output to files

Revision: Getting data from the keyboard

Keyboard



Can you name the command you would use to get data from the keyboard?

Revision: Getting data from the keyboard.

The input command:

```
%read your name as a string
name=input ('what is your name?','s');

%read your age as a string
age=input ('what is your age?');
```

The 's' tells matlab to expect a string

Revision: Controlling output to the screen

Controlling screen output

What two commands could you use write data to the screen?

Revision: The **disp** command

```
>disp('Hello my name is Rod');
```

Writing to the screen

Revision: The **sprintf** command

•The **sprintf** command gives us exact control over what is written to a string.

•Here is an example

```
a=sprintf('My name is %s and I live in flat number %d','Rod',9);
disp(a)
>>My name is Rod and I live in flat number 9
```

•The fields beginning with a % are called **format specifiers**.

- •They tell the computer how to format the output

fprintf - a new command

•**sprintf** stores the string in a variable i.e. ***out*** in this case:

```
out=sprintf('The value of pi is %.5f  %.10f',pi,pi);
disp(out);
>> The value of pi is 3.14159  3.1415926536
```

•This can be a bit clumsy if you don't need the data stored in 'out' again. **fprintf** can write data directly to the screen without the need to store it.

```
>>fprintf('The value of pi is %.5f  %.10f',pi,pi);

>> The value of pi is 3.14159  3.1415926536
```

•The **fprintf**, **sprintf** commands are also used in other languages
  •They are well worth learning

Revision: Loading data from the disk

Reading data from disk

Can you name the commands you would use to **load** data from a file on a disk?

Saving a file to disk

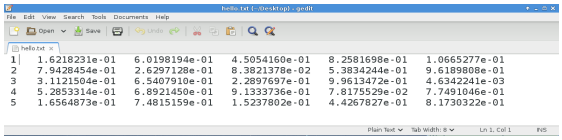
Saving data to a disk

Can you guess what command you would use to **save** data to disk?

There is a hint in the question.....

The **save** command!

```
a=rand(5,5)
save 'hello.txt' a -ascii
```



```
1   1.6218231e-01   6.0198194e-01   4.5054160e-01   8.2581698e-01   1.0665277e-01
2   7.9428454e-01   2.6297128e-01   8.3821378e-02   5.3834244e-01   9.6189808e-01
3   3.1121504e-01   6.5407910e-01   2.2897697e-01   9.9613472e-01   4.6342241e-03
4   5.2853314e-01   6.8921450e-01   9.1333736e-01   7.8175529e-02   7.7491046e-01
5   1.6564873e-01   7.4815159e-01   1.5237802e-01   4.4267827e-01   8.1730322e-01
```

•The option '-ascii' tells the computer that you want the file written out in a human readable format.

•In the lab try leaving this option off the command and see what happens.

Revision: Controlling what we read from a file

•Imagine you have a file containing multiple columns of data:

data.txt

1	2.4	2.64	1.64
2	4.8	5.28	4.28
3	7.2	7.92	6.92
4	9.6	10.56	9.56
5	12	13.2	12.2
6	14.4	15.84	14.84
7	16.8	18.48	17.48

•And you only wanted to plot two columns against one another.

•We could do it like this:

Revision: Controlling what we read from a file

•You would combine what we learnt about arrays and what we learnt about files:

```
clear                          %clear the screen
mydata=load('data.dat');       %load the data file
len=size(mydata)
len=len(1)
x=mydata(1:len,1);             %extract the first column
y=mydata(1:len,3);             %extract the third column
plot(x, y);                    %plot the data
```

•Try this out in your own time, you can copy and paste the code out of the pdf.

More control over files

•However data taken from real world instruments will never be formatted for you like this.

•It will almost always never be in a nice matrix

•It will normally be a mix of numbers and data

•Load and save will therefore not work.

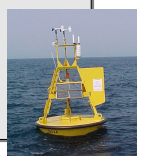
1	2.4	2.64	1.64
2	4.8	5.28	4.28
3	7.2	7.92	6.92
4	9.6	10.56	9.56
5	12	13.2	12.2
6	14.4	15.84	14.84
7	16.8	18.48	17.48

Here is an example..

Example: Reading data from a weather station

•Imagine we had a weather station recording data in this file format:

```
London                  .....
27/11/2013              London
17.00                   27/11/2013
5                       19.00
50                      3
1032                    50
London                  1032
27/11/2013              London
18.00                   27/11/2013
4                       20.00
55                      2
1032                    55
.....                   1032
```

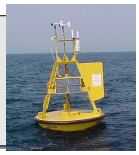


•Can we guess what the lines mean?

A closer look at the file format

•Each line has the following meaning

```
London          %location
27/11/2013      %date
17.00           %time
5               %temperature
50              %Humidity
1032            %pressure
.....
```



•**load** and **save** would not be able to deal with this data as it is **not in a matrix**.

•**We need more control over how we read/write files.**

Reading and writing data generated by sensors /scientific instruments

•As mechanical engineers you will have to make your code interact with all sorts of sensors and instruments which give you data in an odd format. Examples:

Oscilloscope

User: Autopilot wikipedia

Accelerometer

DaimlerChrysler AG

•**To do this we need to understand how computers interact with files.....**

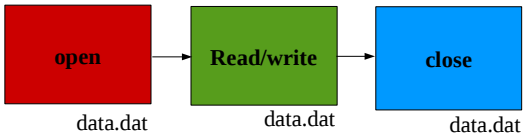
Something fundamental: How computers interact with files.

All computers interact with files in the following way.

•Think about opening a word document, you do the same thing without thinking about it.

•Open it, write text, then close it

How computers interact with files.



•You can think of a file as **a book**. You have to **open it before reading or writing in it**, then you have to **close it** before putting it back on the shelf.

•This is how all software on all computers interacts with files

•Including **supercomputers/your phone/my PC**

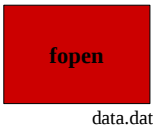
Writing to a file example in MATLAB

Let's look at how we would do this in MATLAB step by step.....

Opening a file in matlab (**fopen**)

•To open a file use the **fopen** command.

```
fopen('data.dat','w');
```



•Where '**data.dat**' is the name of the file to be opened.

•The '**w**' signifies that the computer should **w**rite data to the file.

Opening a file in matlab **(fopen)**

•A computer can have **many files open at the same time**.

•We therefore need a way of telling our program which file it should read/write to/from.

•This is why **fopen** returns a number to the variable **file_id**

•The number returned into file_id is called a file handle.

```
file_id = fopen('data.dat','w');
```

•Every time we want to tell the computer to access file data.dat we need to give it this number.

File handles examples

•Every time we open any file the computer returns a number to a variable called a file handle.

```
>>file_one = fopen('1.dat','w');
>>file_one
3
```

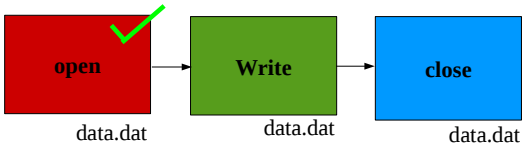
•In all further interactions with the file we use this file handle.

```
>>file_two = fopen('two.dat','w');
>>file_two
7
```

```
>>data_file = fopen('data.dat','w');
>>data_file
1
```

•We have opened the file now, what is the next step?

Writing to a file example in matlab



Writing data to the file line by line (**fprintf**)

•We looked at the **fprintf** before:

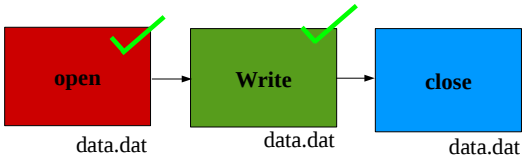
```
>>fprintf('The value of pi is %.5f  %.10f',pi,pi);
>> The value of pi is 3.14159  3.1415926536
```

•If we want to make **fprintf** print to a file instead of to the screen all we need to do is to put the **file handle** as the very first argument.

```
fprintf(file_id,'The value of pi is %.5f  %.10f',pi,pi);
```

•This will write '**The value of pi is 3.14159 3.1415926536**' to the file pointed to by file_id

Writing to a file example in matlab



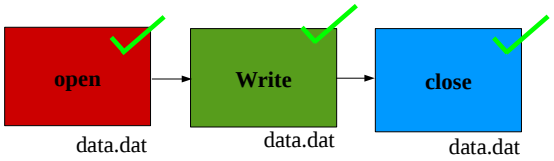
Closing the file when we have finished **fclose()**

•Just like a book, you need to close the file when you are finished

```
fclose(file_id);
```

•Tell the computer which file you want to close by passing it the file handle.

Writing to a file example in MATLAB



Putting it all together (writing to a file)

•The following script will write the text 'The value of pi is 3.14159 3.1415926536' to a file.

```
file_id = fopen('data.dat','w');        %open the file for writing
fprintf(file_id,'The value of pi is %.5f  %.10f',pi,pi); %write to the file
fclose(file_id);                                %close the file
```

YouTube Example

Reading from a file in MATLAB

•This is almost identical writing to a file....

Opening a file for reading (**fopen**)

•Reading from a file almost the same as writing to a file.

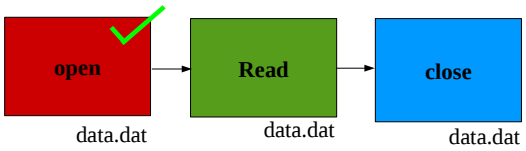
•First open the file for **r**eading (note the 'r')

```
file_id = fopen('data.dat','r');
```

•This will open the file and return a file handle to file_id.

•Every time we want interact with the file we need to give the computer the variable file_id so it knows which file to operate on.

Reading to a file in MATLAB



Reading from a file (**fsacnf**)

•fscanf takes all the same arguments as fprintf and sprintf that we have covered before.


```
my_text= fscanf(file_id, '%s', 1);
```

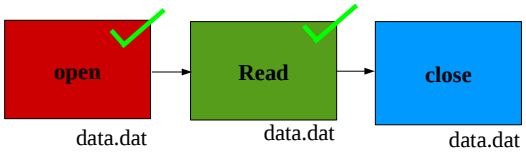
file_id: is the file handle pointing to file we are going to read.

'%s': defines the format of what we will read

my_text: will contain the data read from the file.

1: signifies the number of records to read

Reading to a file in MATLAB

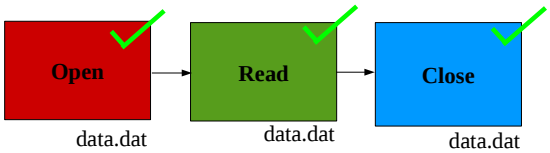


Close the file again (**fclose**)

•As before when we are finished we will just close the file.


```
fclose(file_id);
```


Reading to a file in matlab



Putting it all together (reading from a file)

•The following script will read one line form a file and print it to the screen.

```
file_id = fopen('data.dat','r');      %open the file for writing
my_text= fscanf(file_id, '%s', 1); %read from the file
disp(my_text)                         %write the text to the screen
fclose(file_id);                      %close the file
```

YouTube Example

Our weather station example

•Let's try to write some code that can read this data from a weather station and plot a graph of time against temperature.

```
London
27/11/2013
17.00
5
50
1032
London
27/11/2013
18.00
4
55
1032
.....
.....
London
27/11/2013
19.00
3
50
1032
London
27/11/2013
20.00
2
55
1032
```

Let's remind ourselves of the file format

•Each line has the following meaning

```
London          %location
27/11/2013      %date
17.00           %time
5               %temperature
50              %Humidity
1032            %pressure
```

•As with all complex tasks, break it up into small easy tasks...

Reading the first line

•This code will read the first line:

```
file_id = fopen('data.dat','r');
location= fscanf(file_id, '%s', 1);
fclose(file_id);
```


Reading the first line

•This code will read the first line:

```
file_id = fopen('data.dat','r');        %opens the file
location= fscanf(file_id, '%s', 1);     %reads the location
fclose(file_id);                        %closes the file
```


Reading the the remaining lines with **fscanf**

•Reading the other data from the file

```
file_id = fopen('data.dat','r');        %open the file
location= fscanf(file_id, '%s', 1);     %read the location
date= fscanf(file_id, '%s', 1);         %read the date
time= fscanf(file_id, '%f', 1);         %read the time
temp= fscanf(file_id, '%f', 1);         %read the temperature
humidity= fscanf(file_id, '%f', 1);     %read the humidity
presure= fscanf(file_id, '%f', 1);      %read the pressure
fclose(file_out);                       %close the file
```

•How many sets of data would this read?

•How could we make it read more? YouTube example

Reading all the date entries

•Exactly.... a loop

```
file_id = fopen('data.dat');
while (1)                   %start of while loop
        location= fscanf(file_id, '%s', 1);
        date= fscanf(file_id, '%s', 1);
        time= fscanf(file_id, '%f', 1);
        temp= fscanf(file_id, '%f', 1);
        humidity= fscanf(file_id, '%f', 1);
        presure= fscanf(file_id, '%f', 1);
end                         %end of while loop

fclose(file_id);
```

•Note we open and close the file out side the loop.
•What's wrong with this program?

Testing for the end of a file

The function

```
feof(file_id)
```

Will tell us if we have reached the end of the file. It returns **1** if we have and a **0** if we have not

Exit the loop when we get to the end of the file.

```
file_id = fopen('data.dat');
while (feof(file_id)==0)            %loop while we have not reached the
   location= fscanf(file_id, '%s', 1);       %end of the file
   date= fscanf(file_id, '%s', 1);
   time= fscanf(file_id, '%f', 1);
   temp= fscanf(file_id, '%f', 1);
   humidity= fscanf(file_id, '%f', 1);
   presure= fscanf(file_id, '%f', 1);
end

fclose(file_id);
```

feof(file_id) tests if we have reached the end of the file

YouTube example

Write the data we want into another file

```
file_id = fopen('data.dat');
file_out= fopen('out.dat','w');                 %open the file for writing
while (feof(file_id)==0)
   location= fscanf(file_id, '%s', 1);
   date= fscanf(file_id, '%s', 1);
   time= fscanf(file_id, '%f', 1);
   temp= fscanf(file_id, '%f', 1);
   humidity= fscanf(file_id, '%f', 1);
   presure= fscanf(file_id, '%f', 1);
   fprintf(file_out,'%fa %f\n',time, temp);     %write time and temperature to the file
end

fclose(file_id);
fclose(file_out);                               %close the file for writing
```

YouTube example

Summary

- In this lecture we revised **input** and **output**
  - **input**, **disp**, **sprintf**, **load** and **save**.
- We now know how to read and write to files line by line.
- We have covered **fprintf**, **fopen**, **fclose** and **fscanf**.
- You will find these commands in any computer language you choose to learn.
- You will very probably use these commands in your professional lives