University of Nottingham

Computer Programming with MATLAB

MM1CPM - Lecture 3

2D data arrays and advanced plotting

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

www.facebook.com/mm1cpm

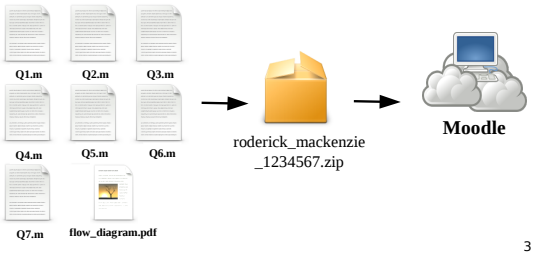
Overview

•In this lecture we will cover:

- •**Coursework**
- •Overview of last lecture
- •Updating and editing 1D arrays
- •2D data
  - •2D arrays
  - •Extracting data from 2D arrays
- •Advanced plotting

Coursework: How to hand it in...

•Please hand in a zip file containing one .m file for each question.
•The zip file should be named firstname_surname_student ID.zip



Coursework: The perfect answer

```
1 %Roderick MacKenzie roderick.mackenzie@notitngham.ac.uk
2 %answers to Q1
3 x=1.5
4 a=sin(x)
5 b=(sin(x))^2              %take power 2 of sin
6 c=x^5+x^4+x^3+x^2+x^1
7 d=exp(sin(x)*cos(x))
8 e=(x+1)*(x-1)*(x^2+1)*(x^3)*cos(x)
9 z=a+b                     %add a and b
10 f=(a+b)/(c+e)
```

•This would give you 100% for question 1
•Note the odd comment.

•We will mark your code by running the .m file.
•**Don't include** the **screen shots** of the plots or what MATLAB **prints out**.
•Please include your **e-mail address** so we can give you feedback.

Overview

•In this lecture we will cover:

- •Coursework
- •**Overview of last lecture**
- •Updating and editing 1D arrays
- •2D data
  - •2D arrays
  - •Extracting data from 2D arrays
- •Advanced plotting

Recap: Complex numbers

•We learnt that in MATLAB complex numbers can be represented by an ***i***
•All mathematical operations that work with ordinary numbers also work with complex numbers.

```
>a=7+i;          %define variable a
>b=8+8i;         %define variable b

>b=b^2           %raise c to the power of 2
>c=a+b           %adding
>d=a-b           %subtracting
>f=b/c           %division
```

Recap: Complex numbers

•MATLAB can do very complicated multiplications for you:

```
>(3+4i)*(7+i)*(3+4i)*(7+i)*(3+4i)*(7+i)*(3+4i)*
(7+i)*(3+4i)*(7+i)*(3+4i)*(7+i) <enter>
    ans=1.9361e9 + 2.5700e8i
```


Recap: Variables v.s. Arrays

$$y=\frac{\text{mark1}+\text{mark2}+\text{mark3}+\text{mark4}+...+\text{mark10}}{10}$$

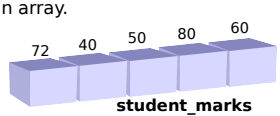
We could define ten variables then take the average:


```
> mark1=72 <enter>
> mark2=40 <enter>
> mark3=50 <enter>
> mark4=80 <enter>
> mark5=..... etc.. <enter>
>y=(mark1+mark2+mark3+mark4+mark5+mark6+mark7+
      mark8+mark9+mark10)/10
```

This looks like a lot of hard work....

Recap: Arrays

•A good way to handle large amounts of data is by using an array.



Arrays – good for large data sets (i.e. marks of students or audio data):

And this is how to make an array in MATLAB, we use square brackets around a list of numbers:

```
>student_marks = [ 72 40 50 80 60 ]
```


Recap: Defining arrays

•Defining arrays by hand

```
>time_in_seconds= [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14]
```

•Getting the computer to define arrays for you (less typing!):

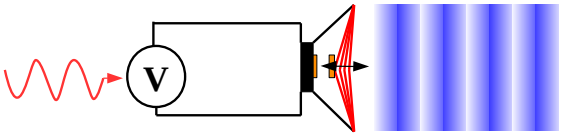
```
>time_in_seconds= linspace(0,14,15)
```

Start End Number of points

•Loading arrays from disk:

```
y = load('matlab_music_file.dat');
```


Recap: Arrays - music to my ears

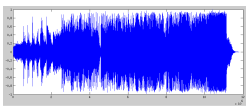


Sinusoid at 50 Hz

Sound wave at 50 Hz

•Sound data is just an array of numbers and we can get the computer to play music, by sending an array to the sound card.

```
> x=linspace(0,10000,10001)
> y=sin(x)
> sound(new_data,44100)
```



Overview

•In this lecture we will cover:

- •Coursework
- •Overview of last lecture
- •**Updating and editing 1D arrays**
- •2D data
  - •2D arrays
  - •Extracting data from 2D arrays
- •Advanced plotting

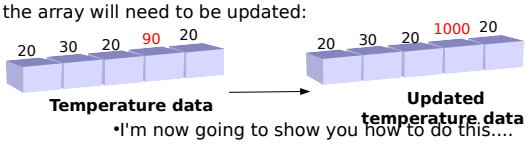
Updating and editing 1D arrays

•Very often in Engineering you will have defined an array to represent physical quantity:

```
>temperature = [ 20 30 20 90 20 ]
```

•Imagine elements in the array represents the temperature from five sensors in a jet engine.

•The temperatures values the sensors measure will change so the array will need to be updated:



•I'm now going to show you how to do this....

Updating and editing 1D arrays – example 1

•To update the array

```
>temperature = [ 20 30 20  90 20 ]
```

1st 2nd 3rd 4th 5th

•We simply type:

```
>temperature(4)=1000
```

•In English this command means replace the 4th element of array 'temperature' with the value 1000

•This will give:

```
>temperature = [ 20 30 20 1000 20 ]
```

1st 2nd 3rd 4th 5th

Updating and editing 1D arrays – example 2

•To update the array

```
>temperature = [ 20 30 20 1000 20 ]
```

1st 2nd 3rd 4th 5th

•We simply type:

```
>temperature(2)=2000
```

•This will give:

```
>temperature = [ 20 2000 20 1000 20 ]
```

1st 2nd 3rd 4th 5th

•Now we know how to update elements in arrays!

Reading values from arrays 1D

•Reading one value from an array is just the reverse process

•Imagine we have the array

```
>temperature = [ 18 19 20 21 10 25 20 30 22 23 ]
```

1st 2nd 3rd 4th 5th 6th 7th 8th 9th 10th

•And I wanted to know the value of the 7th element, I would simply type:

```
>temperature(7) <enter>

20
```

or the 4th element:

```
>temperature(4) <enter>

21
```

•Now we can read and write individual elements to an array

•We will use this later in the lecture

Overview

•In this lecture we will cover:

- •Coursework
- •Overview of last lecture
- •Updating and editing 1D arrays
- **•2D data**
  - **•2D arrays**
  - •Extracting data from 2D arrays
- •Advanced plotting

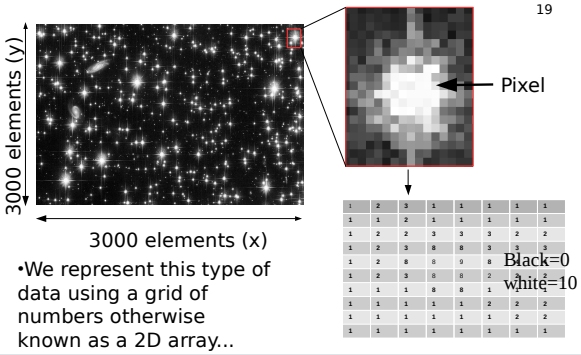
Limitations of 1D arrays.

•Until now we have only had arrays containing lists of numbers.

```
>age_of_students = [ 19 20 21 19 20 21 18 ]

>stock_market_data = [ 5000 5001 4999 ]

>temperature_values = [ 30 31 32 33 34 35 36 37 38 37
36 35 35 35]

>price_of_gold = [ 27 27.1 28 29 27 21.1 27 27.1 28 29
27 21.1 27 27.1 28 29 27 21.1 27 27.1 28 29 27 21.1 ]

>time_in_seconds= [0 3 6 9 12 15 18 21 24 27]
```

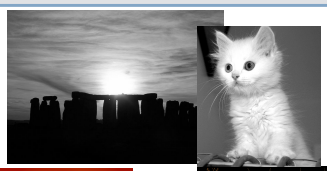
•However very often, the data you are interested in is not list of numbers.....

An example of 2D data

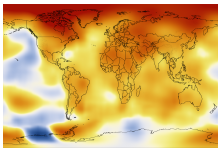


•We represent this type of data using a grid of numbers otherwise known as a 2D array...

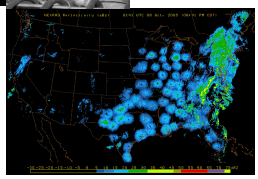
Examples of 2D data sets



2D images



2D map of predicted temperature changes due to global warming



Weather radar data (NASA)

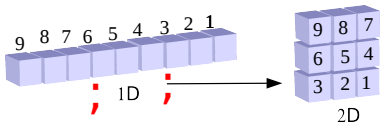
Making 2D arrays in MATLAB is easy....

If we had the 1D array defined as

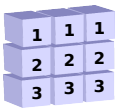
```
>numbers = [ 9 8 7 6 5 4 3 2 1 ]
```

We can turn it in to a 2D array by simply inserting a ; where we want a 'new line'.

```
>numbers = [ 9 8 7 ; 6 5 4 ; 3 2 1 ]
```



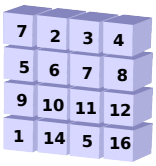
More examples of 2D arrays




```
>> a=[ 1 1 1 ; 2 2 2 ; 3 3 3 ]
```

;
semicolon

All you need to remember is the **semicolon** means start a new line in the array.




```
>> a=[ 7 2 3 4 ; 5 6 7 8 ; 9 10 11 12 ; 1 14 5 16]
```


But that needed a lot of typing...

Making arrays with less typing: ***zeros***

•The 'zeros' command will make a 2D array full of zeros.
>>a = zeros(4,4)

```
>> a=zeros(4,4)

a =

     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

```
>> a=zeros(2,3)

a =

     0     0     0
     0     0     0
```

•The numbers in the brackets specify the x an y size of the array.
•This is a very common command - we will use it later.

Making arrays with less typing: ***rand***

•The 'rand' command will make a 2D array full of random numbers between 0 and 1:
>a = rand(2,2)

```
> a=rand(2,2)

a =

    0.7577    0.3922
    0.7431    0.6555
```

```
> a=rand(2,3)

a =

    0.8235    0.3171    0.0344
    0.6948    0.9502    0.4387
```

•The numbers in the brackets specify the x and y size of the array.

Youtube example

•This is all very interesting but let's look at some real data.

The ***load*** command for 2D data

•Last lecture we used the ***load*** command to ***load*** a list of numbers into a 1D array.

```
> y= load ('my_data.dat')
        y= [10.0 20.1 10.4 40.5 ....
                50.6 80.9 10.1 20.1 ]
```

my_data.dat - Notepad

File Edit Format View Help

10.0
20.1
10.4
40.5
50.6
80.8
80.9
10.1
20.1

The ***load*** command for 2D data

•The load command also works just the same for 2D data:

```
> y= load ('my_data.dat')

      y= 10 20 30
         40 50 60
         70 80 90
```

my_data.dat - Notepad

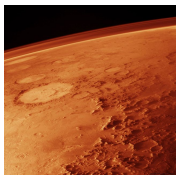
File Edit Format View Help

10 20 30
40 50 60
70 80 90

•Let's look at some real world 2D arrays...

Let's look at some real 2D data...

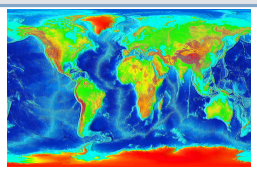
•The Mars Global Surveyor has been busy mapping the surface of mars since its launch in 1996.



•It has produced a 2D height profile map of the martian surface.

Mars height data

Height data on 2D map



Height data on 2D map

We can download a file containing this data from NASA:
mars_height_data.dat

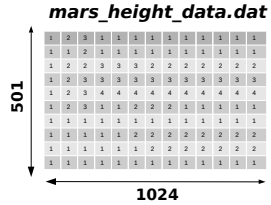
•Let's have a look at this data.....

Finding the size of the data

•The data is stored as a 2D ***grid of numbers***

•The numbers represent height in km.

mars_height_data.dat



•To load this data into a 2D array we simply type:

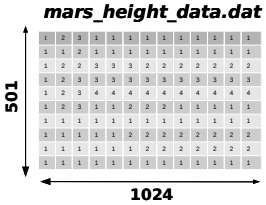
```
data=.....................
```


Finding the size of the data

•The data is stored as a 2D ***grid of numbers***

•The numbers represent height in km.

mars_height_data.dat



•To load this data into a 2D array we simply type:

```
>data = load('mars_height_data.dat');
```

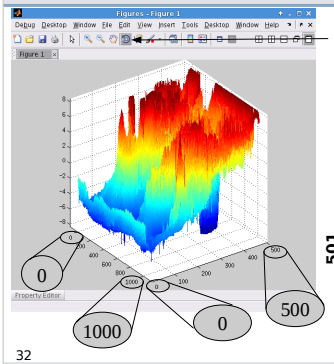
Plotting 2D data using the **surf** command

•The **surf** command performs a **surf**ace plot of a 2D array....

```
>data = load('mars_height_data.dat');

      data=6.32044   6.40884   6.49724...
           6.23204   6.32044   6.40884...
           6.76243   6.76243   6.85083...
           6.93923   7.02762   7.11602..
           6.67403   6.67403   6.76243..
           ..........................................
>surf(data)
```

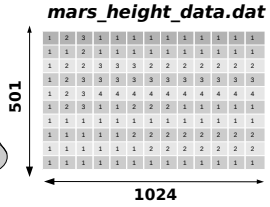
•Let's have a look at the result....

The surf command



•Use this tool to rotate and view the plot.

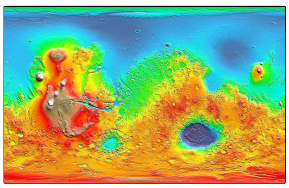
•We can also view the picture from a birds eye view by using the rotate tool.



The surf command

•A birds eye view of our data.

Height data on 2D map



Add a color bar scale

•Adding a color bar scale

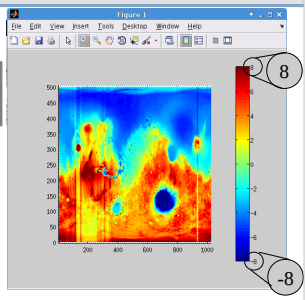
```
>data = load('mars_data.dat');
>surf(data)
>colorbar
```

•The height data is in **km**.

•The numbers on the x/y axis represent the **position** of the data in the **array** (not distance).

•How deep is the crater?

•How high are the mountains in the south?



Youtube

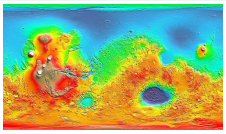
Finding the ***size*** of the data

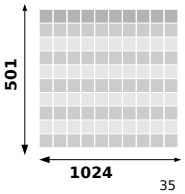
•Up to this point we have known how big our arrays are because we made them or I have told you the size.

•For data sets made by other people the first thing to do is to find out how big the data set is.

•We can do this using the size command ***size*** command:

```
>data = load('mars_height_data.dat');

>size(data)
ans =

      501      1024
```





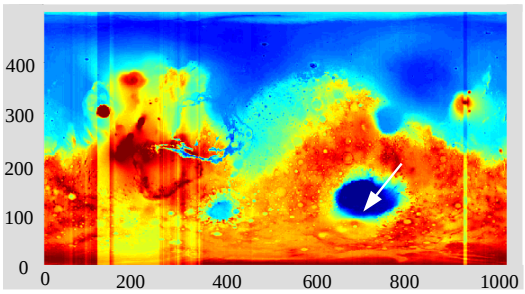
Overview

•In this lecture we will cover:

- •Coursework
- •Overview of last lecture
- •Updating and editing 1D arrays
- •**2D data**
  - •2D arrays
  - •**Extracting data from 2D arrays**
- •Advanced plotting

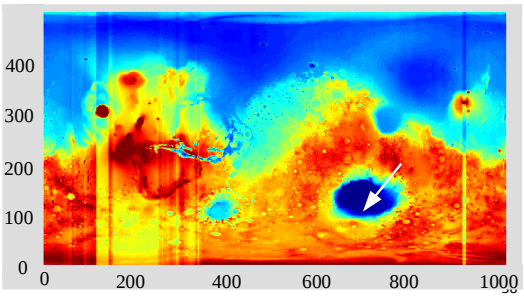
Coordinates in 2D arrays

- What coordinate is the crater at? Just treat it like a map.
  - **x=???, y=???**



Coordinates in 2D arrays

- The coordinate is
  - **x=720 y=120**



Extracting **one element** from a 2D array.

- To extract the depth of the crater from the array we just type:

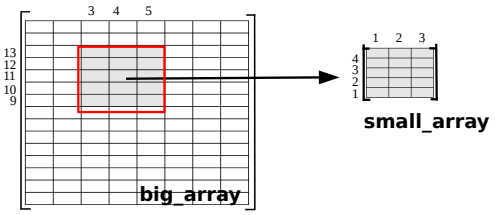
```
>data = load('mars_data.dat');
>data(120,750)                    %array_name(y,x)
      ans=-7.4074
```

- This works exactly the same as extracting data from a 1D array

- 7km is a deep crater!!

Extracting **small 2D arrays** from **big 2D arrays**.

- Imagine we wanted to just zoom in on one feature in the MAP.
- We could do it by first extracting the region of the array that we were interested in... then plotting it.

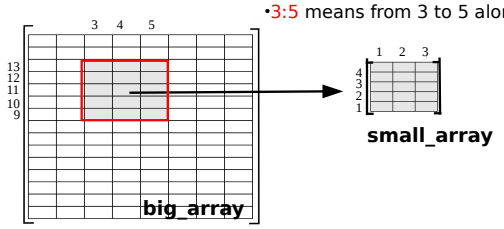


Extracting **small 2D arrays** from **big 2D arrays**.

We can do it like this:

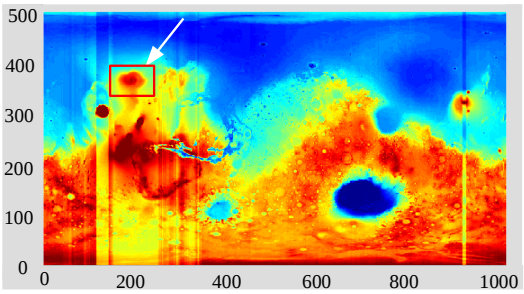
```
>small_array= big_array(9:13,3:5)
```

- 9:13 means from 9 to 13 along y
- 3:5 means from 3 to 5 along x



Let's look at a real example

- Between which coordinates does the mountain lie?
  - x=???:???, and y=???:???

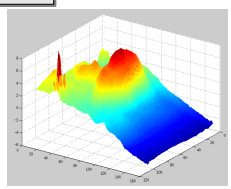


Let's have a closer look at this mountain

•The mountain lies between **x=150:250** and **y=300:450**

```
>data = load('mars_data.dat');
>mountain=data(300:450,150:250)
>surf(mountain)
```

•We can see that there is over 14 km between it's peak and the bottom(!).



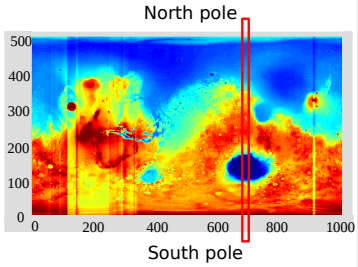
Youtube

Extracting 1D arrays from 2D arrays...

•Imagine we wanted to plot the **height profile from the north** of mars **through the crater** to the **south pole**.

•First we would need to find out where the crater lies.

•It lies on x=700.



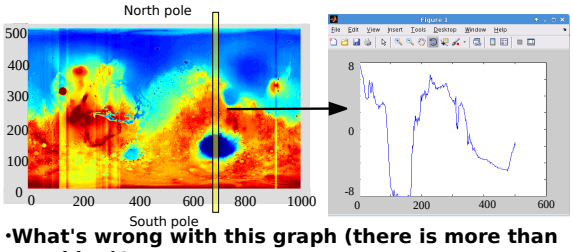
•To extract this strip of data we would type:

```
>strip=data(1:501,700)
```


Extracting strips of data from arrays

•Here is the whole program:

```
>data = load('mars_data.dat');
>strip=data(1:501,700)
>plot(strip)
```



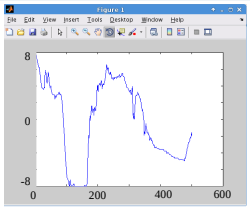
•What's wrong with this graph (there is more than one thing)?

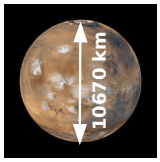
Overview

•In this lecture we will cover:

- •Coursework
- •Overview of last lecture
- •Updating and editing 1D arrays
- •2D data
  - •2D arrays
  - •Extracting data from 2D arrays
- **•Advanced plotting**

Generating an x-axis scale





10672 km

•I need to generate a scale for the x axis.

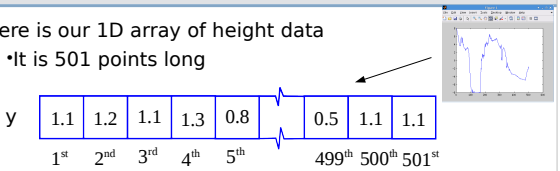
•Relating position in the array to actual distance in km.

•So this is what we do....

Generating an x-axis scale

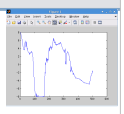
•Here is our 1D array of height data

•It is 501 points long



•We need to generate a second array acting as an x-scale... relating each point to an actual position in km.

Generating an x-axis scale



•Here is our 1D array of height data
  •It is 501 points long

y	1.1	1.2	1.1	1.3	0.8		0.5	1.1	1.1
	1st	2nd	3rd	4th	5th		499th	500th	501st
x	0	20	40	60	80		10630	10650	10670

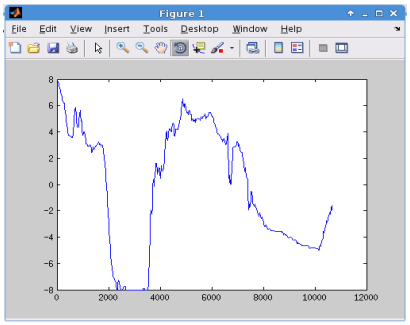
•New array containing actual position of data point in km.
•What command would we use to make this new array?

Generating an x-axis

•Here is the code

```
>data = load('mars_data.dat');      %load the data
>strip=data(1:501,700)              %extract a 1D strip
>x=linspace(0,10670,501)            %make our scale
>plot(x,strip)                      %plot x against y
```

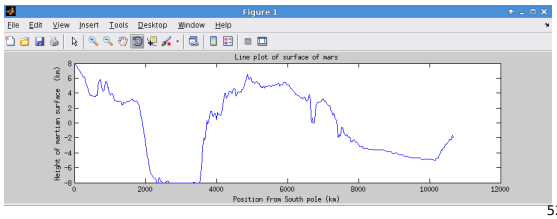

Better



•But we are still missing x-label, y-label and a title.

Adding labels to graphs

```
>data = load('mars_data.dat');
>strip=data(1:501,700)
>x=linspace(0,10672,501)
>plot(x,strip)
>xlabel('Position from South pole (km)')
>ylabel('Height of martian surface  (km)')
>title ('Line plot of surface of mars')
```



Summary

•In this lecture we will cover:

  •Coursework

  •Overview of last lecture

  •Updating and editing 1D arrays

  •2D data
    •2D arrays
    •Extracting data from 2D arrays

  •Advanced plotting