University of
Nottingham

Computer Programming with MATLAB

MM1CPM - Lecture 2

Complex numbers and 1D arrays

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

www.facebook.com/mm1cpm

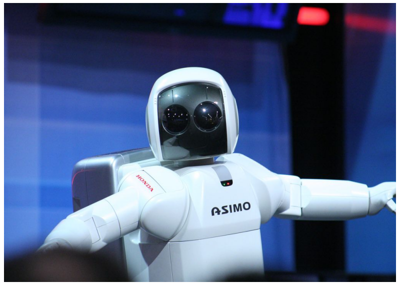
Outline of lecture

- **Recap of last lecture**
- Complex numbers
- Processing very very big amounts of data
  - Arrays
  - Plotting graphs
  - Making music
  - Extracting sub arrays from data
  - Building arrays

Recap: Computers in Engineering

•I showed you that being able to program a computer is an **essential** part of being a Mechanical Engineer. (I also tried to convince you that it is fun!)

•We learnt that writing computer code and remembering commands is **easy** but the tricky part is breaking the problem down into programmable chunks.



•We learnt that computers do **exactly what you tell them to do** – nothing more and nothing less.

Recap: scientific notation and built in functions

- Representing scientific numbers

In mathematics

$1.6 \text{ x } 10^{-19}$

$3 \text{ x } 10^{8}$

1.3806×10^{-23}

→

In MATLAB

```
>1.6e-19

>3e8

>1.3806e-23
```

- Built in functions and constants

```
>sin(0)              %sin of zero
>sin (1)             %sin of one
>sin (pi/2)          %sin of pi/2
>cos (pi/2)          %cos of pi/2
```

Recap: Doing mathematics in MATLAB

```
 >8*3           <enter>          %multiplying

 >7/10          <enter>          %dividing

 >7^3           <enter>          %raise to the power

 >3+7           <enter>          %adding

> 3-7           <enter>          %subtracting

 >(3+7)/4       <enter>          %brackets
```

Recap: variables

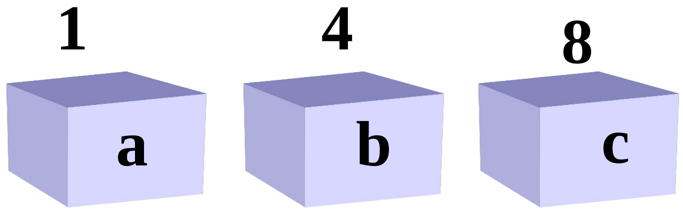
- In the lab we practiced evaluating expressions like this:

$$y=2a+3b+7c^2$$

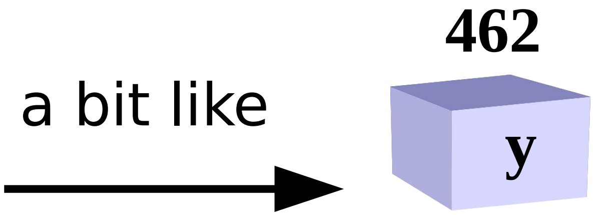
for a=1,b=4
and c=8

- In MATLAB we would type:

```
> a=1 <enter>
> b=4 <enter>
> c=8  <enter>
> y=2*a+3*b+7*(c^2)  <enter>
```



- We learnt that we can think of a variable a bit like a box in which we can store a number.



Outline of lecture

- Recap of last lecture

- **Complex numbers**

- Processing very very big amounts of data
  - Arrays
  - Plotting graphs
  - Making music
  - Extracting sub arrays from data
  - Building arrays

Making mathematics easy with MATLAB

I was having coffee with your mathematics teacher Dr. Richard Tew this week.

- He says that he has been teaching you:
  - **Complex numbers**

- Last lecture I said that learning MATLAB will **make your life as an engineer easier**.

- So I thought I would show you how to do complex numbers in MATLAB – this will enable you to double check your maths problems – **and make your life easier!**

Complex numbers

- You all should have met complex numbers in mathematics already, i.e.:

In mathematics

$a=34+42\boldsymbol{i}$

$b=7+\boldsymbol{i}$

$c=3+3\boldsymbol{i}$

Complex numbers

- You all should have met complex numbers in mathematics already, i.e.:

In mathematics

a=34+42***i***

b=*7*+***i***

c=*3*+*3****i***

In MATLAB

```
>a=34+42i
>b=7+i
>c=3+3i
```

- All you need to do is to remember to write the ***i*** but you know how to do that already!

Complex numbers: multiplying

- In mathematics if you wanted to multiply 34+42***i*** *by* 7+**i** we could do it like this

$$x=(34+42i)(7+i)$$

$$=34*7+34*i+42i*7+42i*i$$

$$=196+328i$$

- **This looks like a lot of work!**

Complex numbers: multiplying

```
>(34+42i)*(7+i)      <enter>
    ans=196 + 328i
```

- Just use the multiply * operator as you did last week for non-complex numbers

Complex numbers: multiplying

- 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
```

With very little effort on your part....

Complex numbers: multiplying

- Everything we have learnt so far in MATLAB will work with with complex numbers

```
>a=7+i;              %define variable a
>b=8+8i;             %define variable b
>c=1+i;              %define variable c

>c=c^2               %raise c to the power of 2
>d=a+b               %adding
>e=a-c               %subtracting
>f=d/c               %division
    ans=12-3i
```

Summary of complex numbers

- The only new thing you have to remember is that ***i*** represents a complex number

- **Multiplying**, **subtracting**, **adding**, **dividing** all work just the same as with normal numbers.

- Complex variables work just like normal non-complex variables.

- If you remember this MATLAB will do the work for you.

Outline of lecture

- Recap of last lecture

- Complex numbers

- **Processing very very big amounts of data**
  - Arrays
  - Plotting graphs
  - Making music
  - Extracting sub arrays from data
  - Building arrays

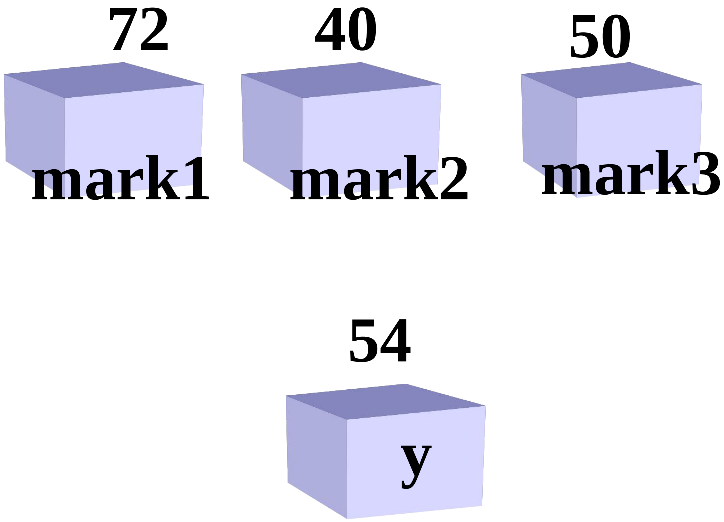
Imagine we were trying to calculate the average of three student's grades

$$y=\frac{mark1+mark2+mark3}{3}$$

for mark1=72,
mark2=40, and
mark3=50

- In MATLAB we would define three variables and then type the equation:

```
> mark1=72 <enter>
> mark2=40 <enter>
> mark3=50  <enter>
> y=(mark1+mark2+mark3)/3  <enter>
```

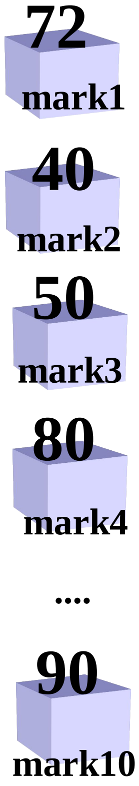


What if we had ten students and wanted to know the average mark?

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

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
```



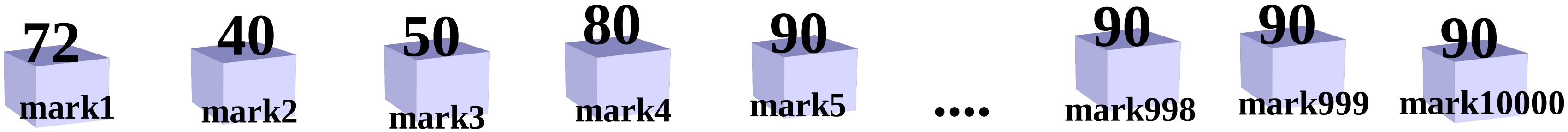
This looks like a lot of hard work....

But want if we had even more data….

- What if I was trying to calculate the average grade of every student in the university?

  - **Would I need to define 10000 variables one for each student?**

ChrisEngelsma



72 mark1

40 mark2

50 mark3

80 mark4

90 mark5

….

90 mark998

90 mark999

90 mark10000

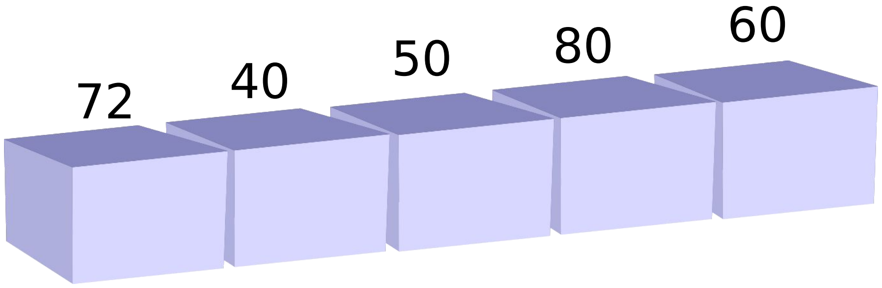
- No – we can use something called an **array**….

Arrays....

•Rather than having lots of variables (boxes) with different names.

• We can define one variable which can hold a whole list of marks.

•This is called an array, I have called this one **student_marks**.



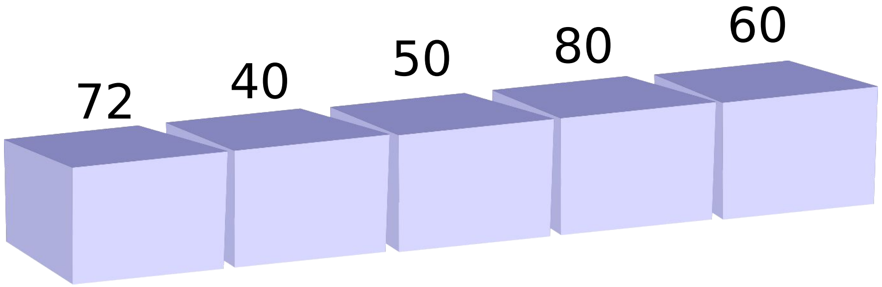
student_marks

Arrays....

•Rather than having lots of variables (boxes) with different names.

• We can define one variable which can hold a whole list of marks.

•This is called an array, I have called this one **student_marks**.

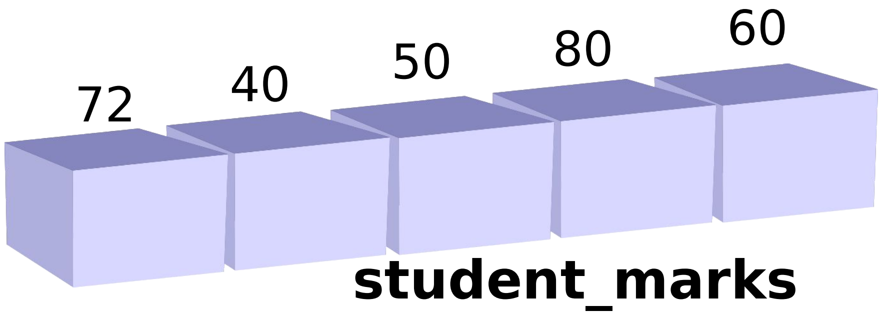


student_marks

An array is a variable which can hold a whole list of numbers.

Let's make an array in MATALB

Here is a picture of our array of student marks:



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 ]
```

[

]

That was easy,
more examples....

More array examples

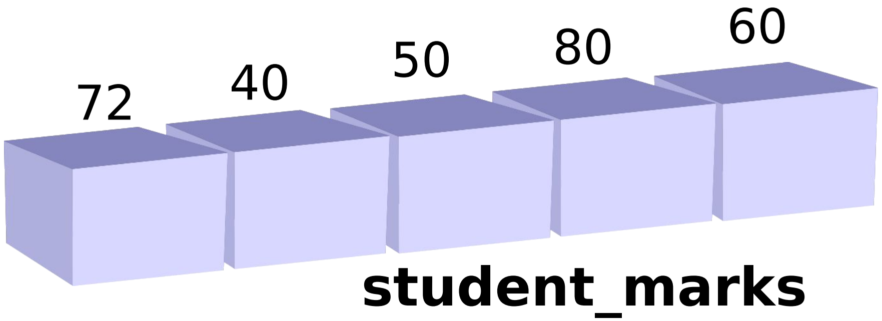
```
 >age_of_students = [ 19 20 21 19 20 21 18 ]

 >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]
```

Square brackets!!!!

- You can make a list of any numbers you like all you have to remember is to put square brackets **[ ]** around your data ....

Working with arrays

MATLAB has lots of built in functions to deal with arrays:



```
>student_marks = [ 72 40 50 80 60]              %make array
>max(student_marks)               %max mark
>min(student_marks)               %min mark
>mean(student_marks)              %mean mark
>std(student_marks)               %std
>sum(student_marks)               %sum
```

Youtube example

More array examples

- Making arrays which count up or down is a common thing to do in engineering.

```
>time= [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19]
```

- Typing in this array would require a lot of typing, so MATLAB has a command to automatically generate arrays for you!

The ***linspace*** command

- Instead of typing this:

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

- We could use the **linspace** command

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

- This makes an array from 0 to 14 with 15 data points.

- Arrays may sound like a handy time saving feature, but in fact it is **much much** more useful than it may first seem......

Using built in functions with arrays

- Functions such as **sin** and **cos** also work on arrays.

```
>x=linspace(0,20,1000)    <enter>

x = [0    0.0200    0.0400    0.0601    0.0801 ....
                  .......  19.9399   19.9600   19.9800   20.0000]
```

- Now let's calculate y=**sin**(x) for each data point. To do this we type:

```
>y=sin(x) <enter>
```

- That's it. MATLAB has calculated the **sin** of 1000 data points! Now let's plot the data using the plot command.

```
>plot(y)
```

Youtube example

All MATLAB commands will work on arrays

```
>x=linspace(0,20,1000) %make our array
>y=cos(x)              %take cos of the array
>y=sin(x)              %take sin of the array
>y=exp(x)              %take exp of the array
>y=tan(x)              %take tan of the array
>y=acos(x)             %take acos of the array
>y=sqrt(x)             %take sqrt of the array
```

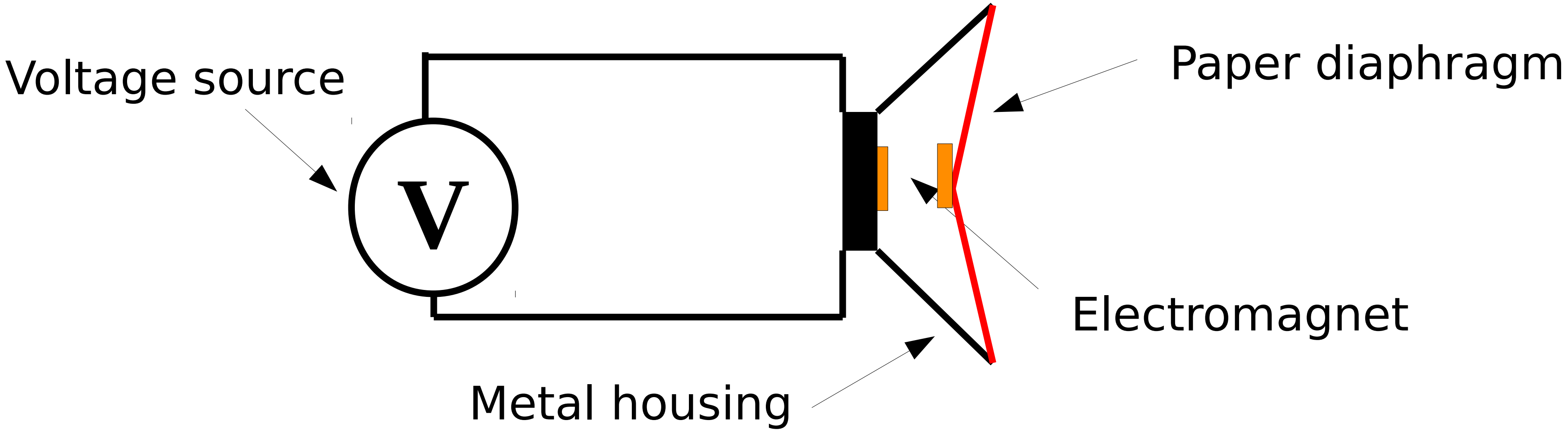
•This is really powerful stuff with one command we can calculate the sin, cos or tan of thousands or millions of numbers.

•But let's make this more exciting

Outline of lecture

- Recap of last lecture
- Complex numbers
- Processing very very big amounts of data
  - Arrays
  - Plotting graphs
  - **Making music**
  - Extracting sub arrays from data
  - Building arrays

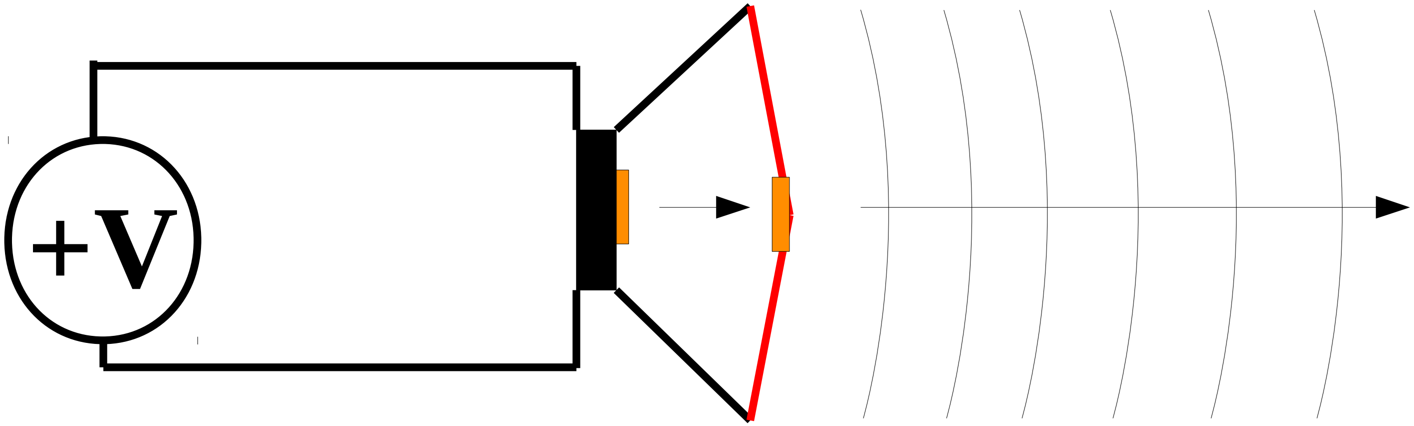
Think about a loud speaker



- A loud speaker is a **paper diaphragm** connected to an **electromagnet**.

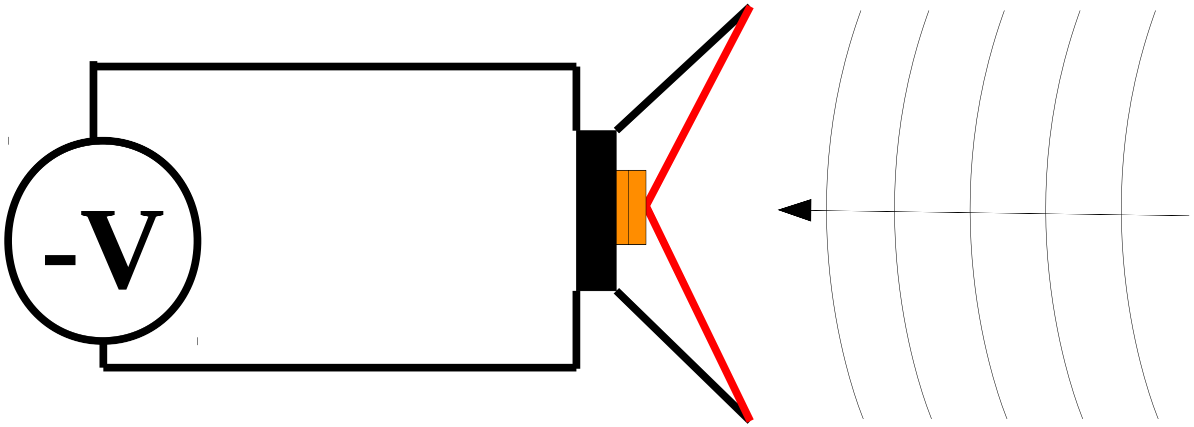
- A **voltage source** is usually used to power the **electromagnet**.

Producing sound waves



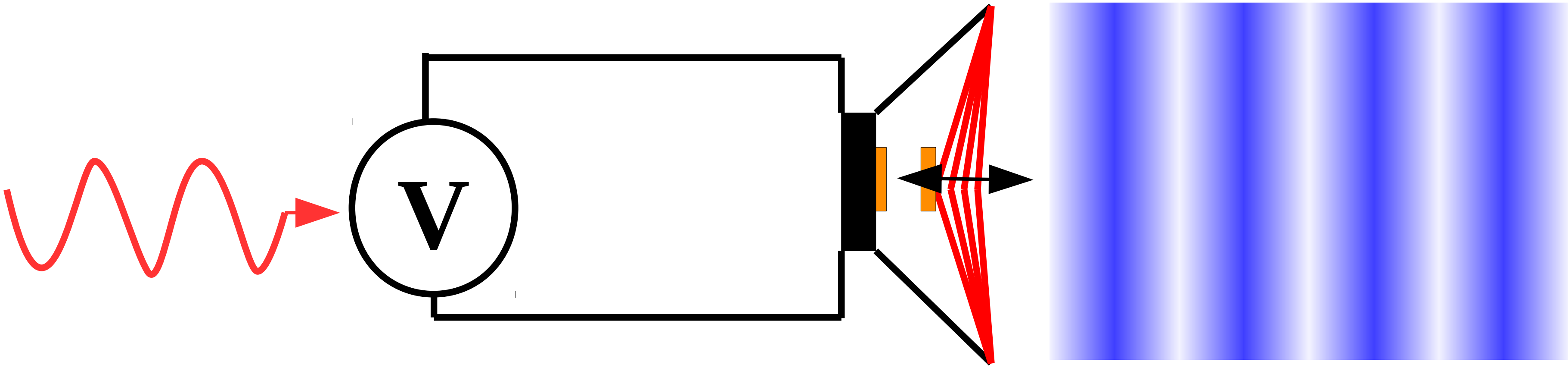
- When a **positive voltage** is applied, the diaphragm moves out producing a sound wave traveling **away** from the speaker.

Producing sound waves



- When a **negative voltage** is applied, the diaphragm moves in producing a sound wave traveling **towards** the speaker.

A sinusoidal voltage



Voltage at 50 Hz

Sound wave at 50 Hz

- When a **sinusoidal voltage** is applied at 50 Hz the speaker produces a **compression wave** at 50 Hz.
- We can produce arrays containing **sin** waves in MATLAB.
- If we can send this array to the computer's sound card we will be able to make audio tones MATLAB.

Making music in MATLAB

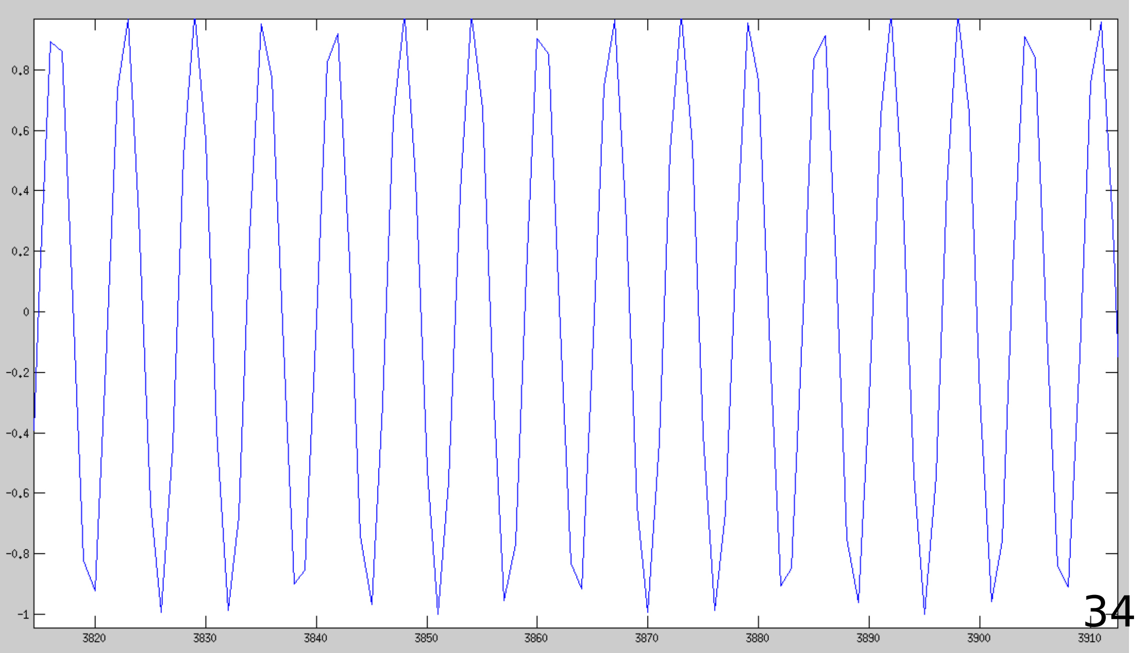
- First let's make a **sin** wave in MATLAB and plot it.

```
>x=linspace(0,10000,10001);
>y=sin(x)
>plot(y)
```

An array from 0 to 10000 with 10001 points

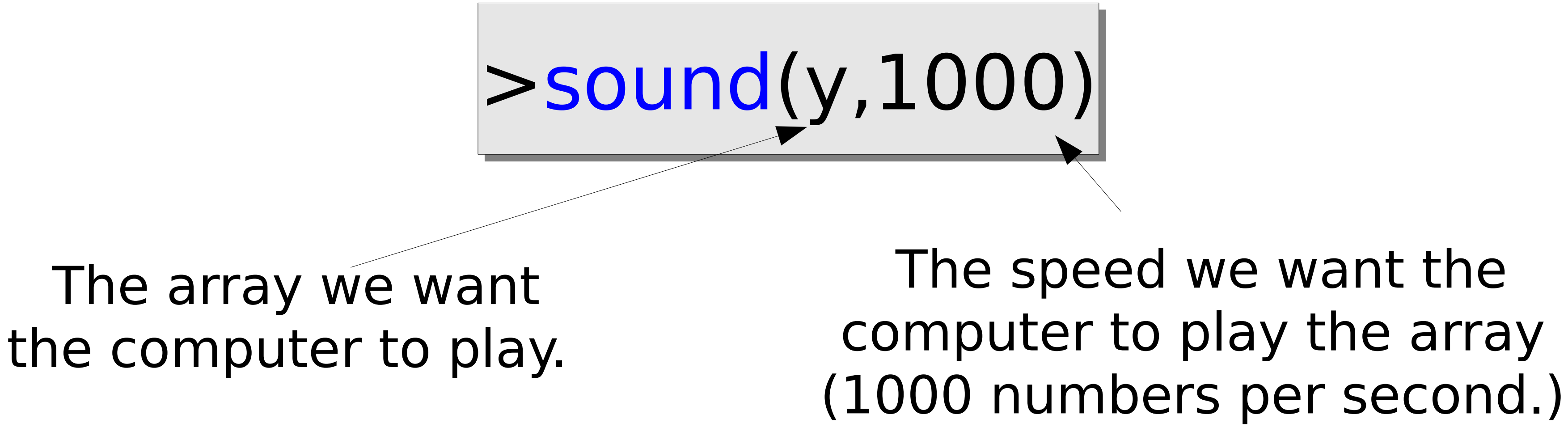
Plot it

Make the sin wave



Send this sound wave to the speakers

- Now let's send this **sin** wave to the speakers using the sound command.



```
>sound(y,1000)
```

The array we want the computer to play.

The speed we want the computer to play the array (1000 numbers per second.)

```
>x=linspace(0,10000,10001);
>y=sin(x)
>plot(y)
>sound(y,1000)
```

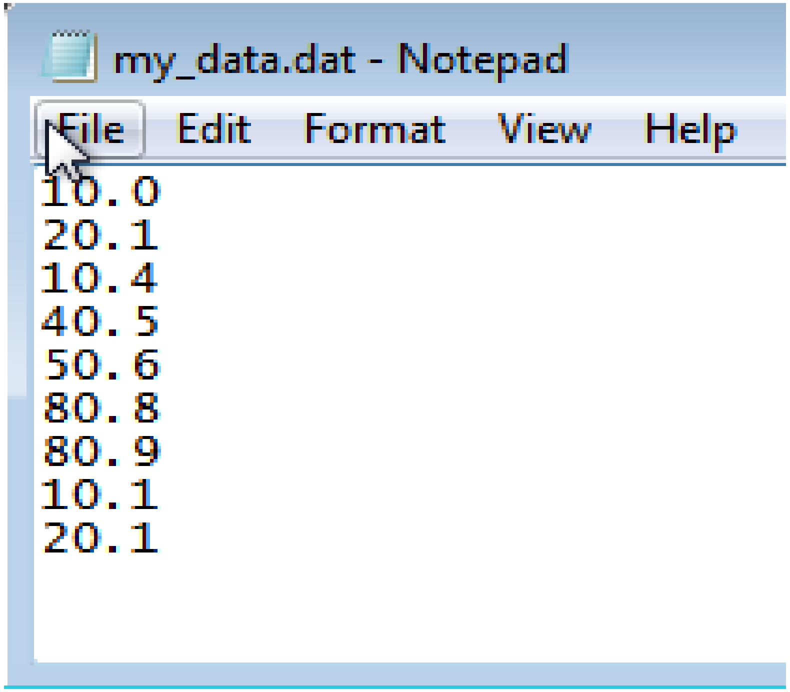
Youtube example

The ***load*** command

- That was music but I would probably not buy the CD!

- More interesting audio data
  - Arrays can also be loaded from a file on disk, we don't have to make them with the **sin** and **cos** command.

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

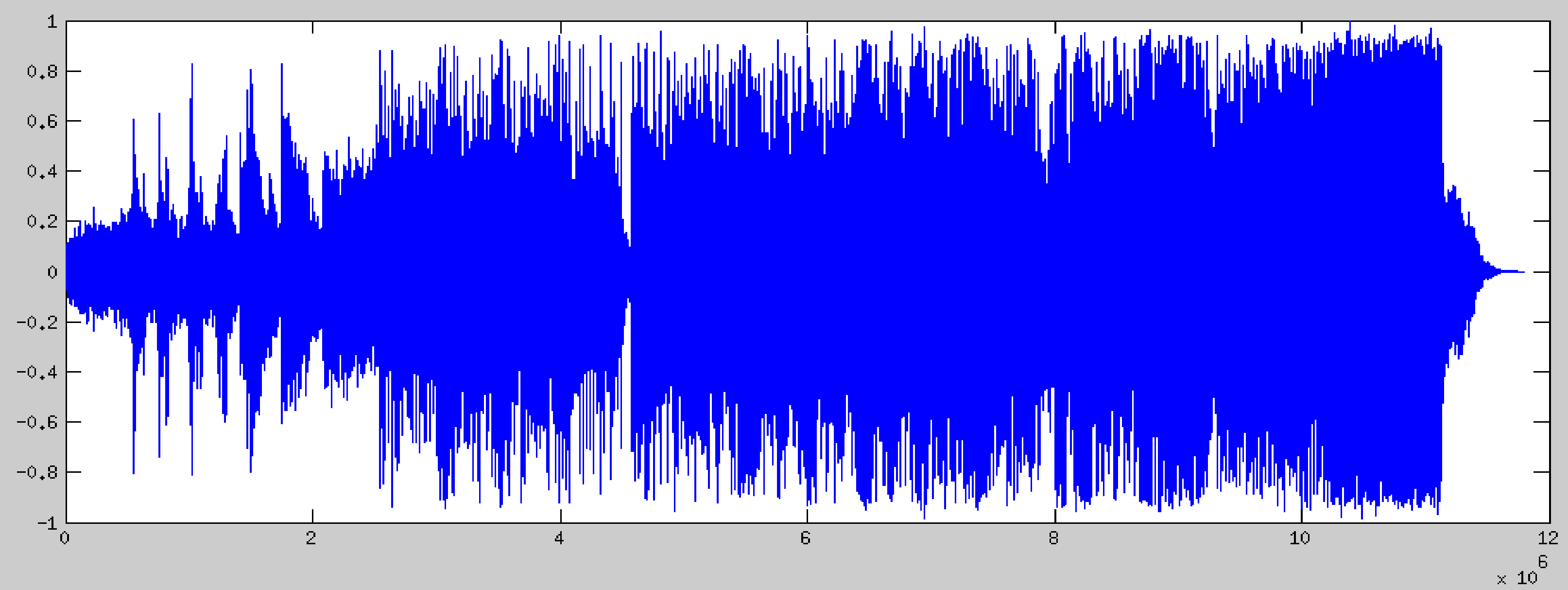


```
10.0
20.1
10.4
40.5
50.6
80.8
80.9
10.1
20.1
```

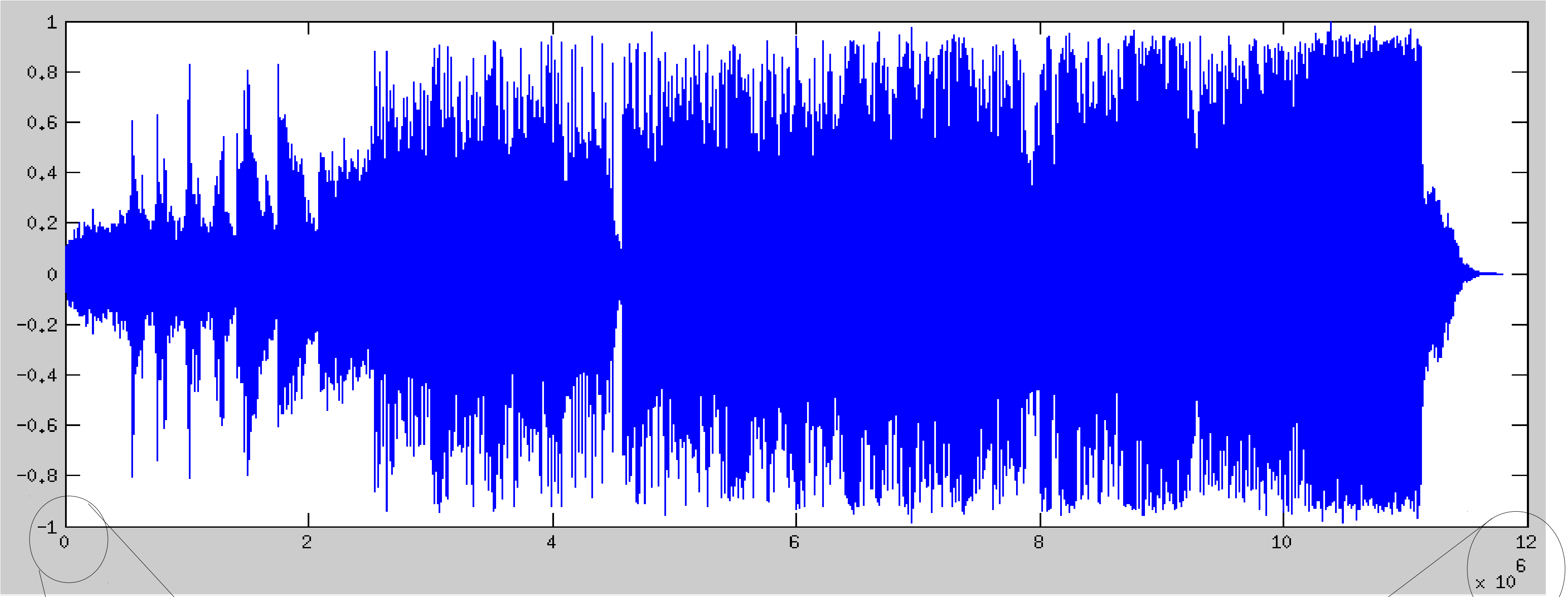
More interesting audio data

- Let's load an array of real audio data from disk and look at it's wave form...

```
> y= load ('audio_data.dat')             %load the file into array y
         y= [2.0462647e-04  -5.8219209e-04 .........
                  4.9949810e-04  -8.1901252e-04 -6.9956481e-04  ]
>plot(y)                                   %plot y
```



Our audio data in more detail

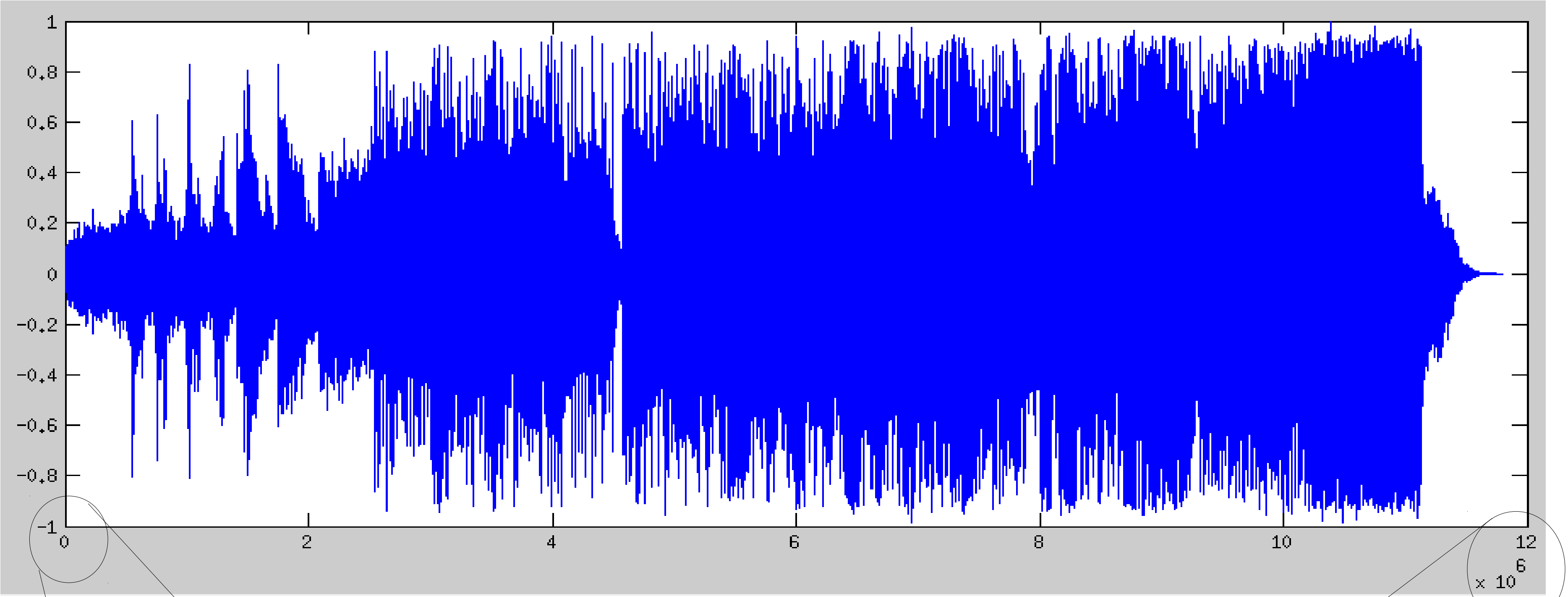


0

- How long is this array?
- Do you think this is music or speech?

12e6

Our audio data in more detail



0

12e6

- •This is CD quality data at 44100 samples per second – it's only 272 seconds long.
- •Try doing this in MS Excel and see how you get on!

Playing the music

- Let's look at this example in MATLAB.

```
>y=load ('list_of_numbers.dat')
>sound(y,44100)
>plot(y)
>clear sound                    %stop playing the music
```

Video

Outline of lecture

- Recap of last lecture
- Complex numbers
- **Processing very very big amounts of data**
  - Arrays
  - Plotting graphs
  - Making music
  - **Extracting sub arrays from data**
  - Building arrays

Array manipulation

- Very often in engineering the data set you collect will be very big but the event you want to study will only be short.

- Think of an earth quake. Most of the time your (seismometer) vibration detector will record nothing, which means you will have lots of uninteresting data.

- We must therefore know how to cut data out of arrays to form smaller sub sets of data.

Extracting data from an array

Imagine we had an array of student marks

```
>marks= [ 70 80 90 100 50 40 70 80 60 70]
           1  2  3  4   5  6  7  8  9 10
```

•In MATLAB, each position in the array is given a number

•If we wanted to extract the elements 4 to 6 and form a smaller array z, we would type:

```
>z=marks(4:6)
     [100 50 40]
```

Extracting data from an array second example

Imagine we had an array of temperature values

```
>temperature= [ 30 29 28 29 27 20 21 22 23 24]
                1  2  3  4  5  6  7  8  9  10
```

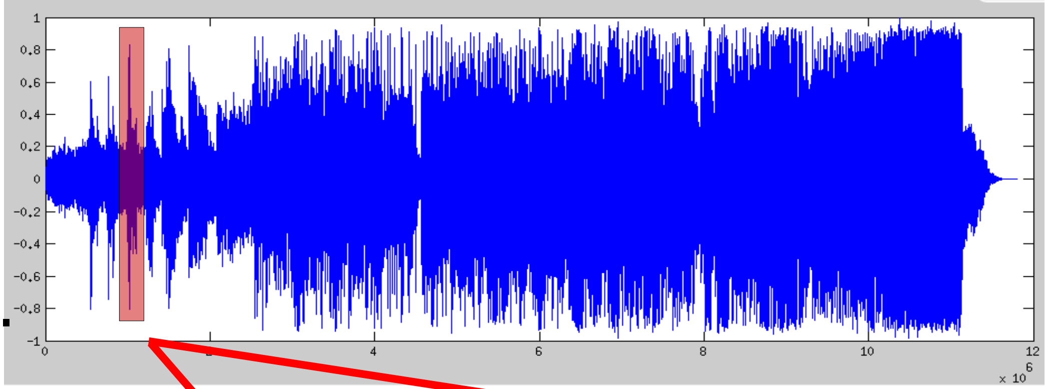
- If we wanted to extract the elements 1 to 5 and form a smaller array first_half, we would type:

```
>first_half=temperature(1:5)
     [ 30 29 28 29 27 ]
```

A more realistic example

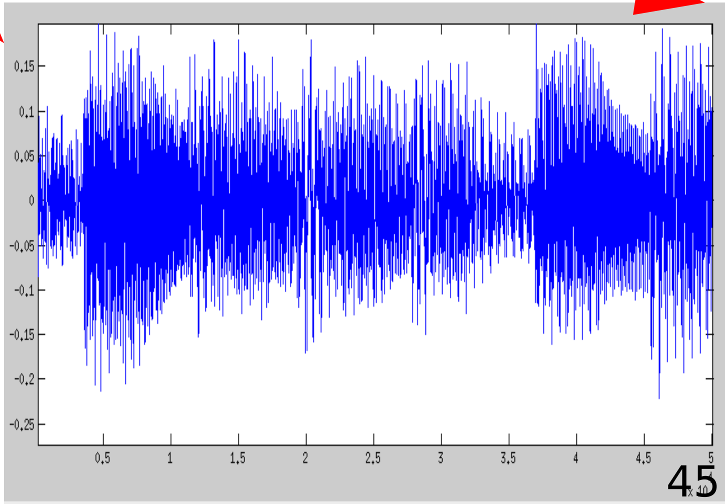
- The song we loaded earlier is a good example of a big data set, it has **11797632** items in the array y.

- Imagine we are only interested in the data between element **1150000** and **1200000** (the guitar).



- We could extract this data and plot it using the following code:

```
>y = load('matlab_music_file.dat');
>new_data=y(1150000:1200000)
>plot(new_data)
>sound(new_data,44100)
```



Video

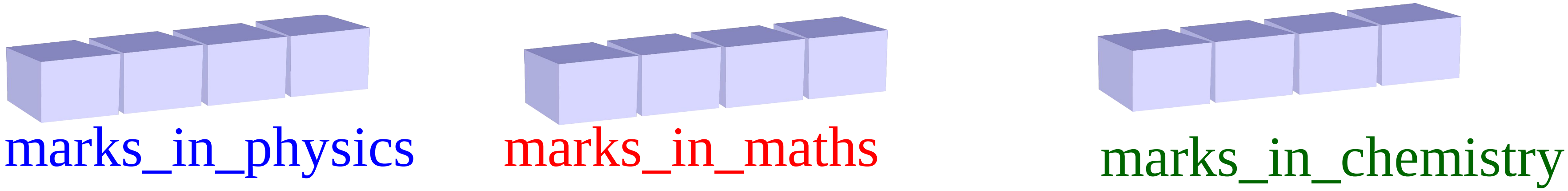
Outline of lecture

- Recap of last lecture
- Complex numbers
- **Processing very very big amounts of data**
  - Arrays
  - Plotting graphs
  - Making music
  - Extracting sub arrays from data
  - **Building arrays**

Building arrays

- Often when analyzing data you will need to join many small arrays together into one big array.

```
> marks_in_physics =[ 80 90 100 70 60 10 40 ]
> marks_in_maths = [ 30 90 10 80 60 10 30 ]
> marks_in_chem =[ 40 90 100 70 80 10 10 ]
```



- A simple example is calculating the average grade of 7 students over three subjects.

Building arrays

I could do it like this:

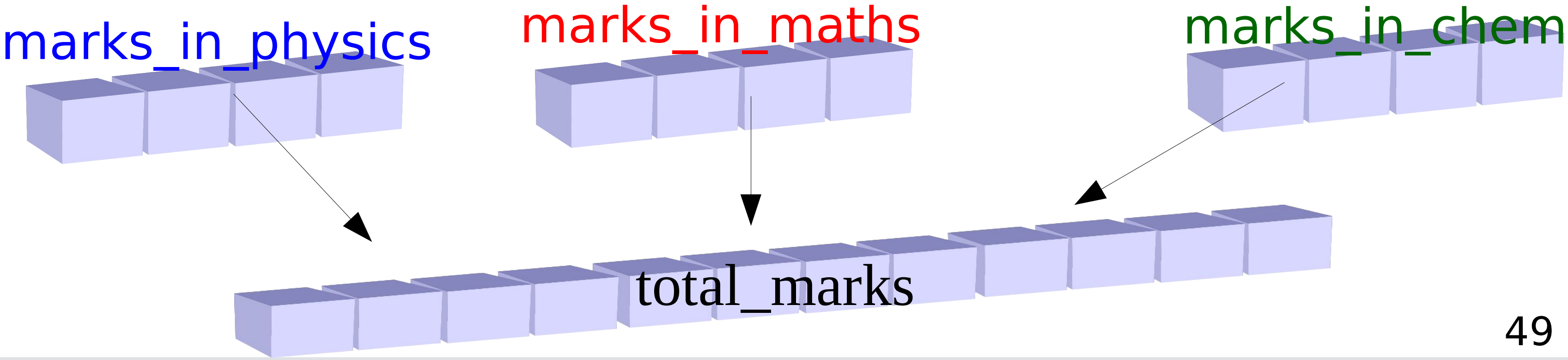
```
> marks_in_physics =[ 80 90 100 70 60 10 40 ]
> marks_in_maths = [ 30 90 10 80 60 10 30 ]
> marks_in_chem =[ 40 90 100 70 80 10 10 ]

>(avg(marks_in_physics)+avg(marks_in_maths
                                        +avg(marks_in_chem))/3
```

But the is another way of doing it...

Building arrays

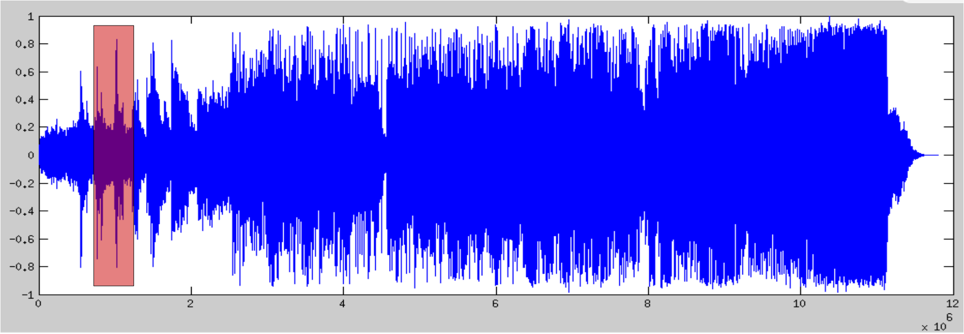
Or, I could joint the arrays together first then take the average

```
> marks_in_physics =[ 80 90 100 70 60 10 40 ]
> marks_in_maths = [ 30 90 10 80 60 10 30 ]
> marks_in_chem =[ 40 90 100 70 80 10 10 ]

>total_marks=[ marks_in_physics marks_in_maths marks_in_chem ]

>total_marks= [ 80 90 100 70 60 10 40  30 90 10 80 60 10 30
                                         40 90 100 70 80 10 10 ]
>avg(total_marks)
```



Building arrays: A MATLAB remix

In the previous audio example we cut the Guitar out of the intro and just played that. Using our new found knowledge we can now play the Guitar intro again and again.



```
>y = load('matlab_music_file.dat');      % load the audio data
>new_data=y(1150000:1200000)             % make a new array with only
                                           the Guitar in the intro
>remix=[new_data new_data new_data new_data new_data]
>sound(remix,44100)                     %play our looped data
```

Video

Summary

- Recap of last lecture
- Complex numbers
- Processing very very big amounts of data
  - Arrays
  - Plotting graphs
  - Making music
  - Extracting sub arrays from data
  - Building arrays