University of Nottingham

Computer Programming with MATLAB

MM1CPM - Lecture 7

Algorithms

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

www.mm1cpm.com

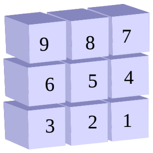
Overview

- **Recap of last lecture**
- Coursework 2
- Algorithms
  - Sorting numbers
  - Integrating with a computer
  - Differentiating with a computer

Recap: Arrays=Matrices

- **Arrays** are the same thing as **Matrices** – there is no difference at all!
- **Array** is the word used in computing. **Matrix** is the word used in Mathematics

$$a=\begin{vmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{vmatrix}$$



a

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

Recap: Summary of matrix operations

- Matrices are Arrays

- You've used most of these operations before, I have just told you that they also work on Matrices/arrays.

Operation	Sign	Example
Multiplying	*	c=a*b
Determinant	det	c=det(a)
Inverse	^	c=a^-1
Transpose	'	c=a'
Subtraction	-	c=a-b
Adding	+	c=a+b

Recap: if are quite important

- **If** statements are quite easy to understand and write.

- But if you make a mistake the consequences can be very serious and potentially kill people.

- There have been cases of errors like this in aircraft fly-by-wire systems..

Recap: if statements – the muffin example

- We then learnt about if-elseif-else statements for decision making, example:

```
weight=80                     %set weight of muffin
if (weight>40)                %if weight bigger than 40
     disp('Too heavy!')
elseif (mark<30)              %if weight bigger than 30
     disp('Too light')
elseif (mark==0)              %if weight equal to 0
     disp('No muffin!!!')
else                          %if none of the above
     disp('Perfect');
end
```

- Hopefully you had a chance to practice this in the lab.

Overview

- Recap of last lecture
- **Coursework 2**
- Algorithms
  - Sorting numbers
  - Integrating with a computer
  - Differentiating with a computer

Coursework

<u>Hand in date for coursework 2:</u>

Wednesday 10th December 15:00
to Moodle

Plagiarism

- Please remember this is individual work not group work.

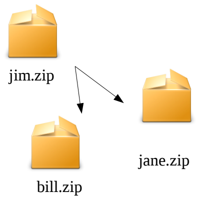
- Please, Please, Please do not hand in work that other people have done.

- At the very least it will result in a mark of 0 and lots of paper work for me.

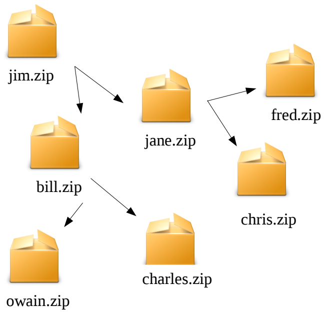
- Please don't give your zip files to your friends...

A Masters module in 2013

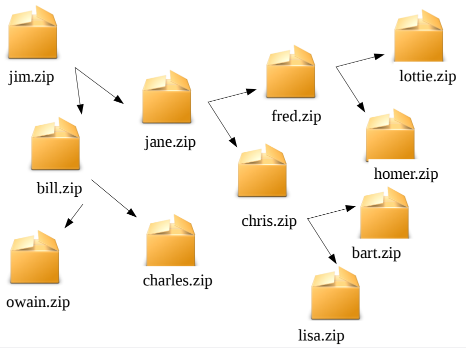
A Masters module in 2013



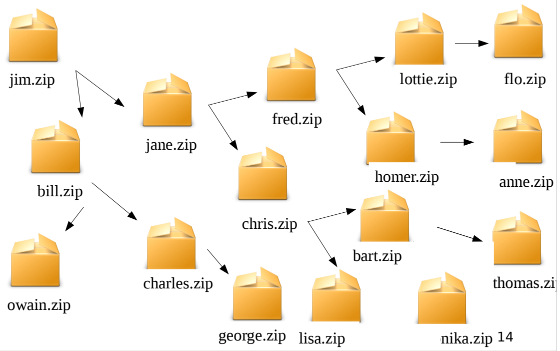
A Masters module in 2013



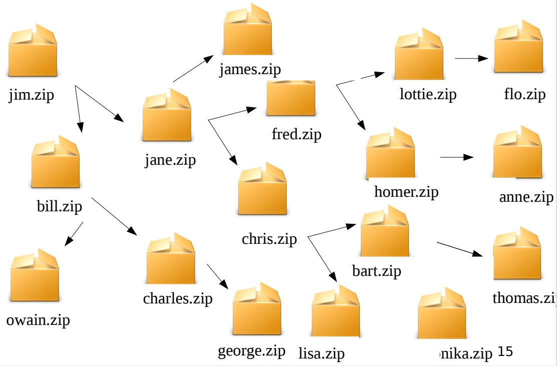
A Masters module in 2013



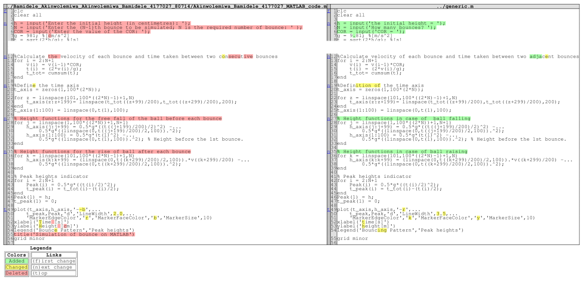
A Masters module in 2013



A Masters module in 2013



Plagiarism: Example 1



./Bamidele_Akinwolemiwa_Akinwolemiwa_Bamidele_4177027_80714/Akinwolemiwa_Bamidele_4177027_MATLAB_code.m

```
clc
clear all

h = input('Enter the initial height (in centimetres): ');
N = input('Enter the (N-1)th bounce to be simulated; N is the required number of bounce: ' );
COR = input('Enter the value of the COR: ');
g = 981; %[cm/s^2]
t = sqrt(2*h/g); %[s]

%Calculate the velocity of each bounce and time taken between two consecutive bounces
for i = 2:N+1
    v(i) = v(i-1)*COR;
    t(i) = (2*v(i)/g);
    t_tot= cumsum(t);
end

%Define the time axis
t_axis = zeros(1,100*(2*N));

for z = linspace(101,100*((2*N)-1)+1,N)
    t_axis(z:z+199)= linspace(t_tot((z+99)/200),t_tot((z+299)/200),200);
end
t_axis(1:100) = linspace(0,t(1),100);

% Height functions for the free fall of the ball before each bounce
for j = linspace(1,100*((2*N))+1,N+1)
    h_axis(j:j+99) = 0.5*g*((t((j+199)/200)/2)^2) -...
        0.5*g*((linspace(0,t((j+199)/200)/2,100)).^2);
    h_axis(1:100) = 0.5*g*(t(1)^2) -...
        0.5*g*((linspace(0,t(1),100)).^2); % Height before the 1st bounce.
end

% Height functions for the rise of ball after each bounce
for k = linspace(101,100*((2*N)-1)+1,N)
    h_axis(k:k+99) = (linspace(0,t((k+299)/200)/2,100)).*v((k+299)/200) -...
        0.5*g*((linspace(0,t((k+299)/200)/2,100)).^2);
end

% Peak heights indicator
for i = 2:N+1
    Peak(i) = 0.5*g*((t(i)/2)^2);
    t_peak(i) = t_tot(i)-(t(i)/2);
end
Peak(1) = h;
t_peak(1) = 0;

plot(t_axis,h_axis,'--b',...
    t_peak,Peak,'d','LineWidth',2.0,...
    'MarkerEdgeColor','r','MarkerFaceColor','b','MarkerSize',10)
xlabel('Time [s]')
ylabel('Height [cm]')
legend('Bounce Pattern','Peak heights')
title('Simulation of bounce on MATLAB')
grid minor
```

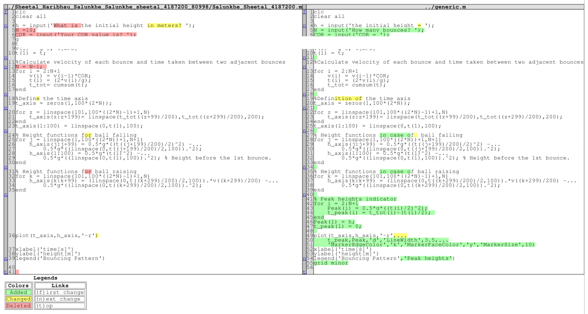
../generic.m

```
clc
clear all

h = input('the initial height = ');
N = input('How many bounces? ');
COR = input('COR = ');
g = 9.81; %[m/s^2]
t = sqrt(2*h/g); %[s]

%Calculate velocity of each bounce and time taken between two adjacent bounces
for i = 2:N+1
    v(i) = v(i-1)*COR;
    t(i) = (2*v(i)/g);
    t_tot= cumsum(t);
end

%Definition of the time axis
t_axis = zeros(1,100*(2*N));

for z = linspace(101,100*((2*N)-1)+1,N)
    t_axis(z:z+199)= linspace(t_tot((z+99)/200),t_tot((z+299)/200),200);
end
t_axis(1:100) = linspace(0,t(1),100);

% Height functions in case of  ball falling
for j = linspace(1,100*((2*N))+1,N+1)
    h_axis(j:j+99) = 0.5*g*((t((j+199)/200)/2)^2) -...
        0.5*g*((linspace(0,t((j+199)/200)/2,100)).^2);
    h_axis(1:100) = 0.5*g*(t(1)^2) -...
        0.5*g*((linspace(0,t(1),100)).^2); % Height before the 1st bounce.
end

% Height functions in case of ball raising
for k = linspace(101,100*((2*N)-1)+1,N)
    h_axis(k:k+99) = (linspace(0,t((k+299)/200)/2,100)).*v((k+299)/200) -...
        0.5*g*((linspace(0,t((k+299)/200)/2,100)).^2);
end

% Peak heights indicator
for i = 2:N+1
    Peak(i) = 0.5*g*((t(i)/2)^2);
    t_peak(i) = t_tot(i)-(t(i)/2);
end
Peak(1) = h;
t_peak(1) = 0;

plot(t_axis,h_axis,'-r',...
    t_peak,Peak,'d','LineWidth',3.5,...
    'MarkerEdgeColor','k','MarkerFaceColor','y','MarkerSize',10)
xlabel('time[s]')
ylabel('height[m]')
legend('Bouncing Pattern','Peak heights')

grid minor
```

Legends

Colors	Links
Added	(f)irst change
Changed	(n)ext change
Deleted	(t)op

Plagiarism: Example 2



./Sheetal_Haribhau_Salunkhe_Salunkhe_sheetal_4187200_80998/Salunkhe_Sheetal_4187200.m

```
clc
clear all

h = input('What is the initial height in meters? ');
N =10;
COR = input('Your COR value is? ');
g
t

t(1) = t;

%Calculate velocity of each bounce and time taken between two adjacent bounces
N = N-1;
for i = 2:N+1
    v(i) = v(i-1)*COR;
    t(i) = (2*v(i)/g);
    t_tot= cumsum(t);
end

%Define the time axis
t_axis = zeros(1,100*(2*N));

for z = linspace(101,100*((2*N)-1)+1,N)
    t_axis(z:z+199)= linspace(t_tot((z+99)/200),t_tot((z+299)/200),200);
end
t_axis(1:100) = linspace(0,t(1),100);

% Height functions for ball falling
for j = linspace(1,100*((2*N))+1,N+1)
    h_axis(j:j+99) = 0.5*g*((t((j+199)/200)/2)^2) -...
        0.5*g*((linspace(0,t((j+199)/200)/2,100)).^2);
    h_axis(1:100) = 0.5*g*(t(1)^2) -...
        0.5*g*((linspace(0,t(1),100)).^2); % Height before the 1st bounce.
end

% Height functions for ball raising
for k = linspace(101,100*((2*N)-1)+1,N)
    h_axis(k:k+99) = (linspace(0,t((k+299)/200)/2,100)).*v((k+299)/200) -...
        0.5*g*((linspace(0,t((k+299)/200)/2,100)).^2);
end







plot(t_axis,h_axis,'-r')


xlabel('time[s]')
ylabel('height[m]')
legend('Bouncing Pattern')

```

../generic.m

```
clc
clear all

h = input('the initial height = ');
N = input('How many bounces? ');
COR = input('COR = ');

t(1) = t;

%Calculate velocity of each bounce and time taken between two adjacent bounces

for i = 2:N+1
    v(i) = v(i-1)*COR;
    t(i) = (2*v(i)/g);
    t_tot= cumsum(t);
end

%Definition of the time axis
t_axis = zeros(1,100*(2*N));

for z = linspace(101,100*((2*N)-1)+1,N)
    t_axis(z:z+199)= linspace(t_tot((z+99)/200),t_tot((z+299)/200),200);
end
t_axis(1:100) = linspace(0,t(1),100);

% Height functions in case of  ball falling
for j = linspace(1,100*((2*N))+1,N+1)
    h_axis(j:j+99) = 0.5*g*((t((j+199)/200)/2)^2) -...
        0.5*g*((linspace(0,t((j+199)/200)/2,100)).^2);
    h_axis(1:100) = 0.5*g*(t(1)^2) -...
        0.5*g*((linspace(0,t(1),100)).^2); % Height before the 1st bounce.
end

% Height functions in case of ball raising
for k = linspace(101,100*((2*N)-1)+1,N)
    h_axis(k:k+99) = (linspace(0,t((k+299)/200)/2,100)).*v((k+299)/200) -...
        0.5*g*((linspace(0,t((k+299)/200)/2,100)).^2);
end

% Peak heights indicator
for i = 2:N+1
    Peak(i) = 0.5*g*((t(i)/2)^2);
    t_peak(i) = t_tot(i)-(t(i)/2);
end
Peak(1) = h;
t_peak(1) = 0;

plot(t_axis,h_axis,'-r',...
    t_peak,Peak,'d','LineWidth',3.5,...
    'MarkerEdgeColor','k','MarkerFaceColor','y','MarkerSize',10)
xlabel('time[s]')
ylabel('height[m]')
legend('Bouncing Pattern','Peak heights')
grid minor
```

Legends

Colors	Links
Added	(f)irst change
Changed	(n)ext change
Deleted	(t)op

Overview

- Recap of last lecture
- Coursework 2
- **Algorithms**
  - Sorting numbers
  - Integrating with a computer
  - Differentiating with a computer

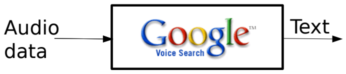
What is an algorithm?

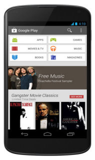
- Algorithms are, wikipedia: *A computational procedure for solving a problem.*
- *Or my definition: A handy piece of computer code that does something useful for you.*
- Algorithms are everywhere, you use them every day of your life.

Here are some real world examples....

Speech recognition algorithm

- It takes your voice and turns it into text

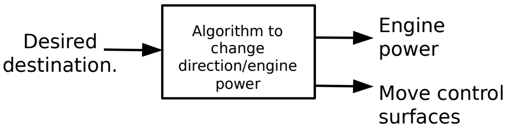



Copyright google

'A handy piece of computer code that does something useful for you.'

Cruse missile guidance algorithm

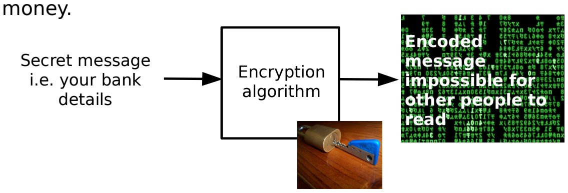
•It takes the desired destination and turns it into engine control signals and changes the ruder/ailerons to get there.



'A handy piece of computer code that does something useful for you.'

Encryption algorithm

•Encodes you bank details so people can't steel your money.


Secret message i.e. your bank details
Encryption algorithm
Encoded message impossible for other people to read

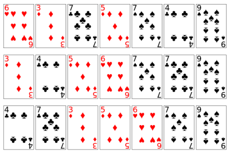
'A handy piece of computer code that does something useful for you.'

Overview

- Recap of last lecture
- **Algorithms**
  - **Sorting numbers – a classical algorithm**
  - Integrating with a computer
  - Differentiating with a computer

Sorting algorithm

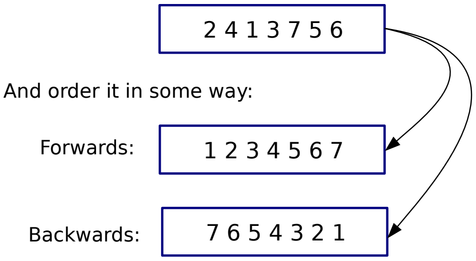
- These are one of the most common algorithms you will come across.
  - Sorting your **music** into alphabetical order.
  - Sorting phone numbers
  - Prioritizing patients for speed of treatment on the **NHS**
  - Sorting **scientific data**
  - Sorting playing cards


Dcoetzee

- Sorting data is a whole topic in it's self.

Sorting algorithms

•The general idea of a sort algorithm is to take a list of data i.e.:



2 4 1 3 7 5 6

And order it in some way:

Forwards: 1 2 3 4 5 6 7

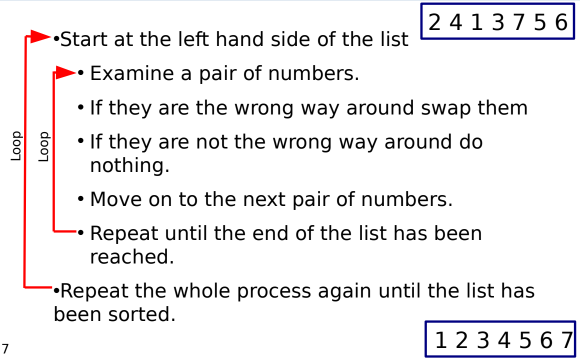
Backwards: 7 6 5 4 3 2 1

The bubble sort algorithm

- Look at this example of LEGO men sorting blocks using a bubble sort into height order.
- The heights of the LEGO represent the value of the number....

- Can you work out what is happening i.e. what algorithm are the LEGO men performing?

The bubble sort algorithm



2 4 1 3 7 5 6

•Start at the left hand side of the list

- Examine a pair of numbers.
- If they are the wrong way around swap them
- If they are not the wrong way around do nothing.
- Move on to the next pair of numbers.
- Repeat until the end of the list has been reached.

•Repeat the whole process again until the list has been sorted.

1 2 3 4 5 6 7

In detail

```
    Iteration 01:  15    8    1    5    3
    Iteration 02:  8    15    1    5    3
    Iteration 03:  8    1    15    5    3
    Iteration 04:  8    1    5    15    3
Do it again....
    Iteration 05:  8    1    5    3    15
    Iteration 06:  1    8    5    3    15
    Iteration 07:  1    5    8    3    15
    Iteration 08:  1    5    3    8    15
Do it again....
    Iteration 09:  1    5    3    8    15
    Iteration 10:  1    5    3    8    15
    Iteration 11:  1    3    5    8    15
    .......
```

Turning this into MATLAB

•Often the biggest challenge in writing computer code is first figuring out what the algorithm should actually do.

•Then writing computer code to solve the problem is easy

•Now let's turn this algorithm into MATLAB code step by step....

Break the problem down into small bits...

Let's first figure out how to swap two numbers in an array

```
numbers=[3  2  1  5  4]          %define an array to sort

i=1                              %use i to index the array.

temp=numbers(i);                 %store the i th element in temp
                                 % [3  2  1  5  4], temp=3

numbers(i)=numbers(i+1);         %copy the i+1 th element to the ith
                                 % [2  2  1  5  4] , temp=3

numbers(i+1)=temp;               %copy the old i th element to i+1
                                 %position
                                 % [2  3  1  5  4] , temp=3
```

Break the problem down into small bits...

Let's first figure out how to swap two numbers in an array

```
numbers=[3  2  1  5  4]     %define an array to sort

i=1                          %use i to index the array.

temp=numbers(i);            %temp=3
numbers(i)=numbers(i+1);  %numbers=[2  2  1  5  4]
numbers(i+1)=temp;          %numbers=[2  3  1  5  4]
```

But this only swaps the 1st and 2nd element we need it to swap all elements... so we need a **for** loop

Add a for loop.

```
numbers=[3  2  1  5  4]        %our array we want to sort

 for i=1:4                        %start of a for loop to count
                                  %through the list

     temp=numbers(i);              %store the i th element
     numbers(i)=numbers(i+1);  %copy the i+1 th element to the ith
     numbers(i+1)=temp;            %copy the old i th element to i+1
                                   %position
 end
```

This will swap all numbers in the array, what are we missing now?

The bubble sort algorithm implemented in MATLAB

We were missing an if statement, only swap the numbers if ***numbers*(i)** and ***numbers*(i+1)** are the wrong way around..

```
numbers=[3  2  1  5  4]        %our array we want to sort

for i=1:4                         %start of a for loop to count

    if (numbers(i)>numbers(i+1))          %if statement

    temp=numbers(i);                   %do the swap
    numbers(i)=numbers(i+1);
    numbers(i+1)=temp;

    end
end
```

Now the code will pass through the list or numbers once..... what is now missing?

The bubble sort algorithm implemented in MATLAB

```
numbers=[3  2  1  5  4]        %our array we want to sort

for ii=i:5                   %pass five times over the data
  for i=1:4                      %start of a for loop to count

     if (numbers(i)>numbers(i+1))        %if statement

       temp=numbers(i);                  %do the swap
       numbers(i)=numbers(i+1);
       numbers(i+1)=temp;

     end
  end
end
```

•Add another loop to run the swapping procedure a lot of times.

[YouTube Example](#)   That's it.

What have we learnt form the bubble sort algorithm?

1) Get a clear idea in your mind first what you want the algorithm to do – often this is harder than actually writing the MATLAB code it's self.

2) When trying to write MATLAB code for a tricky problem, break the problem down into smaller chunks.

3) Get these small chunks working first.

4) Then expand the MATLAB code bit-by-bit until you have solved your problem.

Overview

- Recap of last lecture
- Coursework 2
- Algorithms
  - Sorting numbers
  - **Integrating with a computer**
  - Differentiating with a computer

Numerical integration

How would you go about integrating:

$$y=\int \tan^{-1} x\,dx$$

Any ideas?

Numerical integration

Well I did't know, so I used google, and it told me this:

$$y=\int \tan^{-1} x\,dx = x\tan^{-1}x-\int \frac{x}{1+x^2}dx$$

$$=x\tan^{-1}x-\frac{1}{2}\int\frac{2x}{1+x^2}dx$$

$$=x\tan^{-1}x-\frac{1}{2}\ln\left(1+x^2\right)+C$$

Numerical integration

But, what if:

- The function was really very complicated and you could not figure out how to integrate it.
- Or if there was no analytically answer, like this integral:

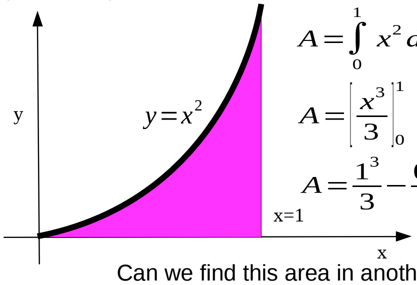
$$y=\int e^{-x^2}dx$$

- Or you just did not have time to sit around working out the integral (project deadlines)
- After today's lecture you will never have to do another analytical integral again.

Let's think about integration

Q) What is intergration?

A) It's just finding the area under the line



$$A=\int_0^1 x^2\,dx$$

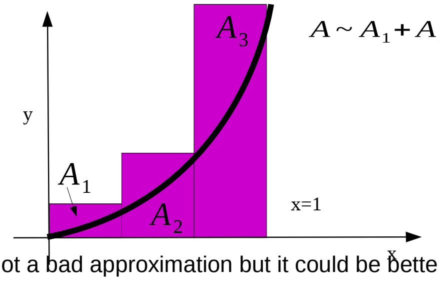
$$A=\left[\frac{x^3}{3}\right]_0^1$$

$$A=\frac{1^3}{3}-\frac{0^3}{3}=\frac{1}{3}$$

Can we find this area in another way?

Let's think about integration

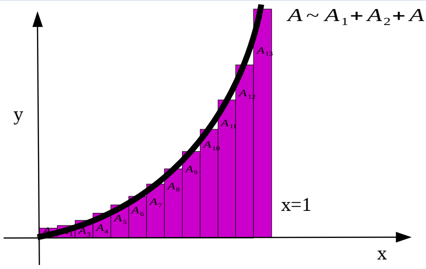
We can approximate under the line with a series of boxes, then just add up the area of the boxes.



$$A\sim A_1+A_2+A_3$$

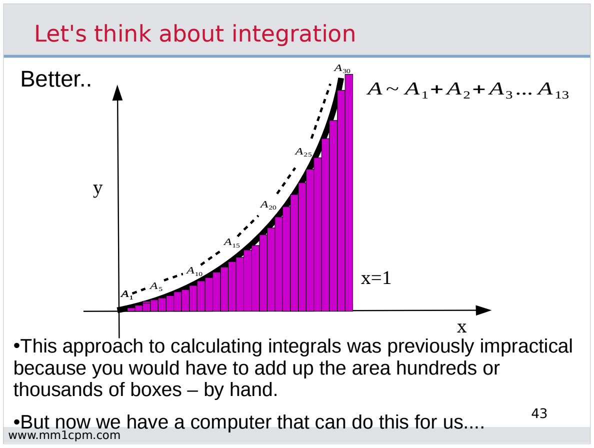
It's not a bad approximation but it could be better, let's make the boxes smaller..

Let's think about integration



$$A\sim A_1+A_2+A_3\ldots A_{13}$$

Smaller boxes = more accurate estimate of area



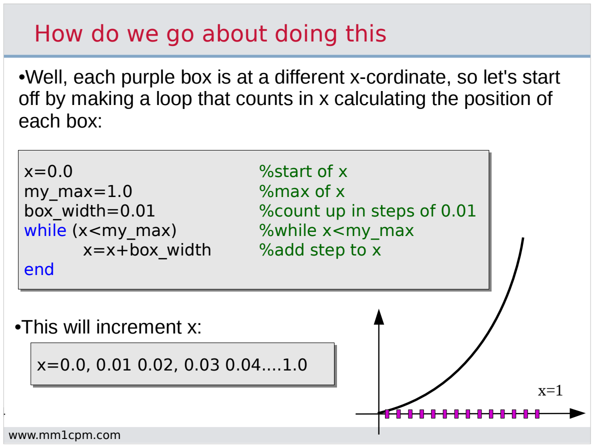
Let's think about integration

Better..

$A \sim A_1 + A_2 + A_3 \ldots A_{13}$

A_1
A_5
A_{10}
A_{15}
A_{20}
A_{25}
A_{30}
y
x
x=1

- This approach to calculating integrals was previously impractical because you would have to add up the area hundreds or thousands of boxes – by hand.
- But now we have a computer that can do this for us....



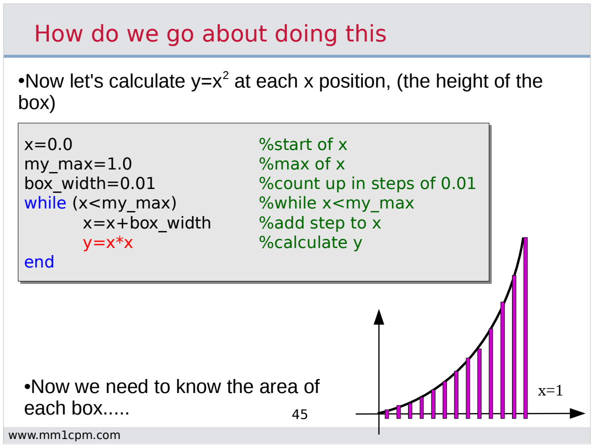
How do we go about doing this

- Well, each purple box is at a different x-cordinate, so let's start off by making a loop that counts in x calculating the position of each box:

```
x=0.0                   %start of x
my_max=1.0              %max of x
box_width=0.01          %count up in steps of 0.01
while (x<my_max)        %while x<my_max
     x=x+box_width      %add step to x
end
```

- This will increment x:

```
x=0.0, 0.01 0.02, 0.03 0.04....1.0
```

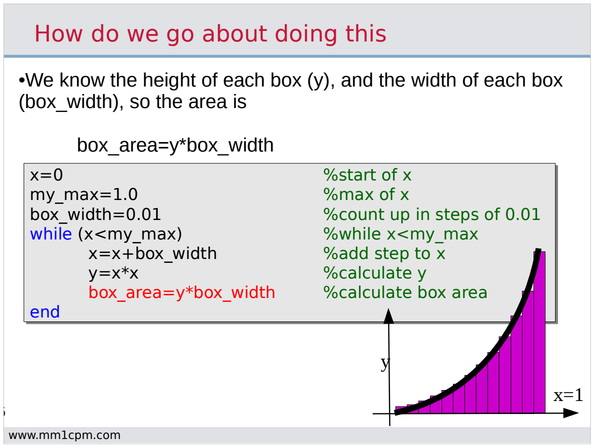


How do we go about doing this

- Now let's calculate $y=x^2$ at each x position, (the height of the box)

```
x=0.0                   %start of x
my_max=1.0              %max of x
box_width=0.01          %count up in steps of 0.01
while (x<my_max)        %while x<my_max
     x=x+box_width      %add step to x
     y=x*x              %calculate y
end
```

- Now we need to know the area of each box.....

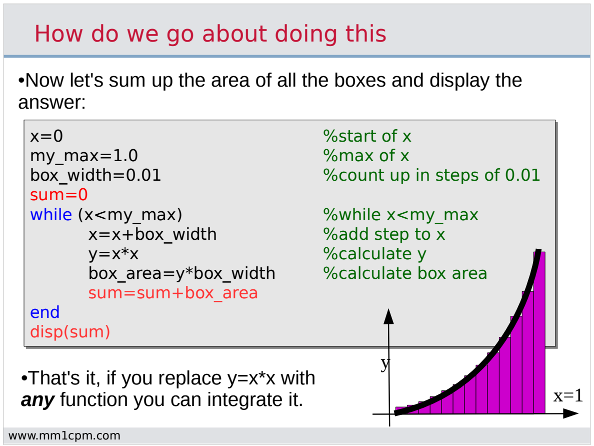


How do we go about doing this

- We know the height of each box (y), and the width of each box (box_width), so the area is

box_area=y*box_width

```
x=0                          %start of x
my_max=1.0                   %max of x
box_width=0.01               %count up in steps of 0.01
while (x<my_max)             %while x<my_max
     x=x+box_width           %add step to x
     y=x*x                   %calculate y
     box_area=y*box_width    %calculate box area
end
```

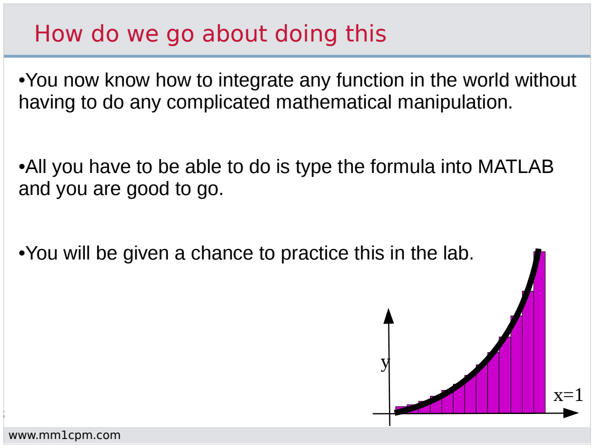


How do we go about doing this

- Now let's sum up the area of all the boxes and display the answer:

```
x=0                          %start of x
my_max=1.0                   %max of x
box_width=0.01               %count up in steps of 0.01
sum=0
while (x<my_max)             %while x<my_max
     x=x+box_width           %add step to x
     y=x*x                   %calculate y
     box_area=y*box_width    %calculate box area
     sum=sum+box_area
end
disp(sum)
```

- That's it, if you replace y=x*x with ***any*** function you can integrate it.



How do we go about doing this

- You now know how to integrate any function in the world without having to do any complicated mathematical manipulation.
- All you have to be able to do is type the formula into MATLAB and you are good to go.
- You will be given a chance to practice this in the lab.

Overview

- Recap of last lecture: if statements
- Coursework 2
- **Algorithms**
  - Sorting numbers
  - Integrating with a computer
  - **Differentiating with a computer**

Numerical differentiation

•In mathematics you learn how to do differentiation.

•For example you know that the derivative of x^2 with respect to x is 2x

$$\frac{\partial}{\partial x} x^2 = 2x$$

You can also solve more complex problems...

Numerical differentiation

More complex problems like this....

$$\frac{\partial}{\partial x} \sin(x)\cos(x)$$

Use chain rule ← $\boxed{\frac{\partial}{\partial x} uv = \frac{\partial u}{\partial x} v + \frac{\partial v}{\partial x} u}$

$$= \cos(x)\cos(x) - \sin(x)\sin(x)$$

$$= \cos^2(x) - \sin^2(x)$$

Substitute in an identity ← $\boxed{\sin^2(x) + \cos^2(x) = 1}$

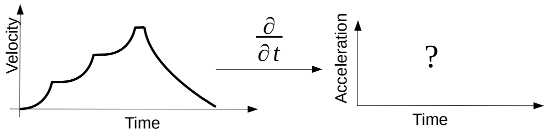
$$= 1 - 2\sin^2(x)$$

However this looks like a lot of work...

Numerical differentiation

•Your problems are going to get more complex. Often the problems you are interested in can't be differentiated in this way:

•For example, if you collect velocity data from a sports car:



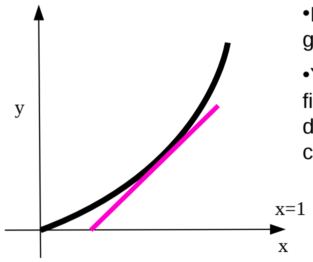
•How would you calculate the **gradient** of this **velocity** data to get **acceleration**?

•You can't differentiate data like this with any of the methods you previously learnt.

Numerical differentiation

•To differentiate real wold data we need a different approach.

•Let's go back to what differentiation really means.

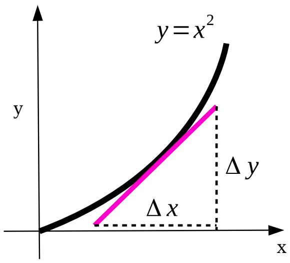


•It's just finding the gradient of a line.

•You did this at school by first taking your ruler and drawing a tangent to the curve.

Numerical differentiation

•Then you calculated the slope of the tangent by calculating Δx and Δy.



$$slope = \frac{\Delta y}{\Delta x} = \frac{dy}{dx}$$

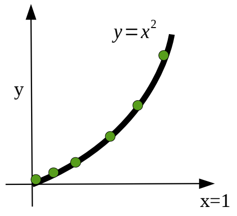
•The slope is the derivative – that's it.

•So if we can get the computer to do this we can differentiate any function.

•Let's have a go at writing an algorithm to do this.

Numerical differentiation

- We first need to generate the curve in MATLAB

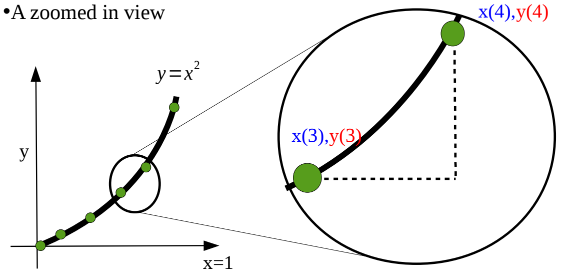


```
x=[0.0 0.2 0.4 0.6 0.8 1 ]
y=[ 0 0 0 0 0 0 ]
for n=1:6
 y(n)=x(n)^2
end
disp(y)
  [ 0.0 0.04 0.16 0.36 0.64 1.0 ]
```

- We now have an array of x and y points representing the line.
- The y position in each point can be accessed with y(n) and the x position of each point can be accessed with x(n)

Numerical differentiation

- A zoomed in view

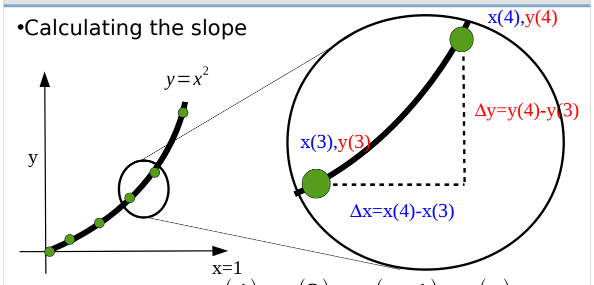


- The position y(n) and x(n)

x =[0.0 0.2 0.4 0.6 0.8 1]
y=[0.0 0.04 0.16 0.36 0.64 1.0]

Numerical differentiation

- Calculating the slope

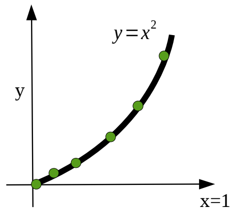


$$slope=\frac{\Delta y}{\Delta x}=\frac{y(4)-y(3)}{x(4)-x(3)}=\frac{y(n+1)-y(n)}{x(n+1)-x(n)}$$

- Let's program this into MATLAB

Numerical differentiation

- The MATLAB code

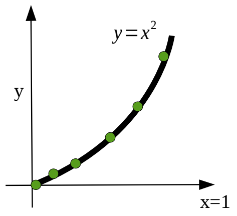


```
x=[0.0 0.2 0.4 0.6 0.8 1.0 ]
y=[0 0 0 0 0 0 ]
for n=1:6
 y(n)=x(n)^2
end

for n=1:6
 dx=x(n+1)-x(n)
 dy=y(n+1)-y(n)
 slope=dy/dx
end
```

That's it. The program will now calculate the slope at each point.

Numerical differentiation

- Generate an array of gradient..



```
x=[0.0 0.2 0.4 0.6 0.8 1.0 ]
y=[0 0 0 0 0 0 ]
slope=[0 0 0 0 0 0 ]
for n=1:6
 y(n)=x(n)^2
end

for n=1:6
 dx=x(n+1)-x(n)
 dy=y(n+1)-y(n)
 slope(n)=dy/dx
end
```

Overview

- Today we have covered:
  - Recap of last lecture
  - Algorithms
    - »Sorting numbers
    - »Integrating with a computer
    - »Differentiating with a computer