University of Nottingham

Computer Programming with MATLAB

MM1CPM - Lecture 5
for and while loops

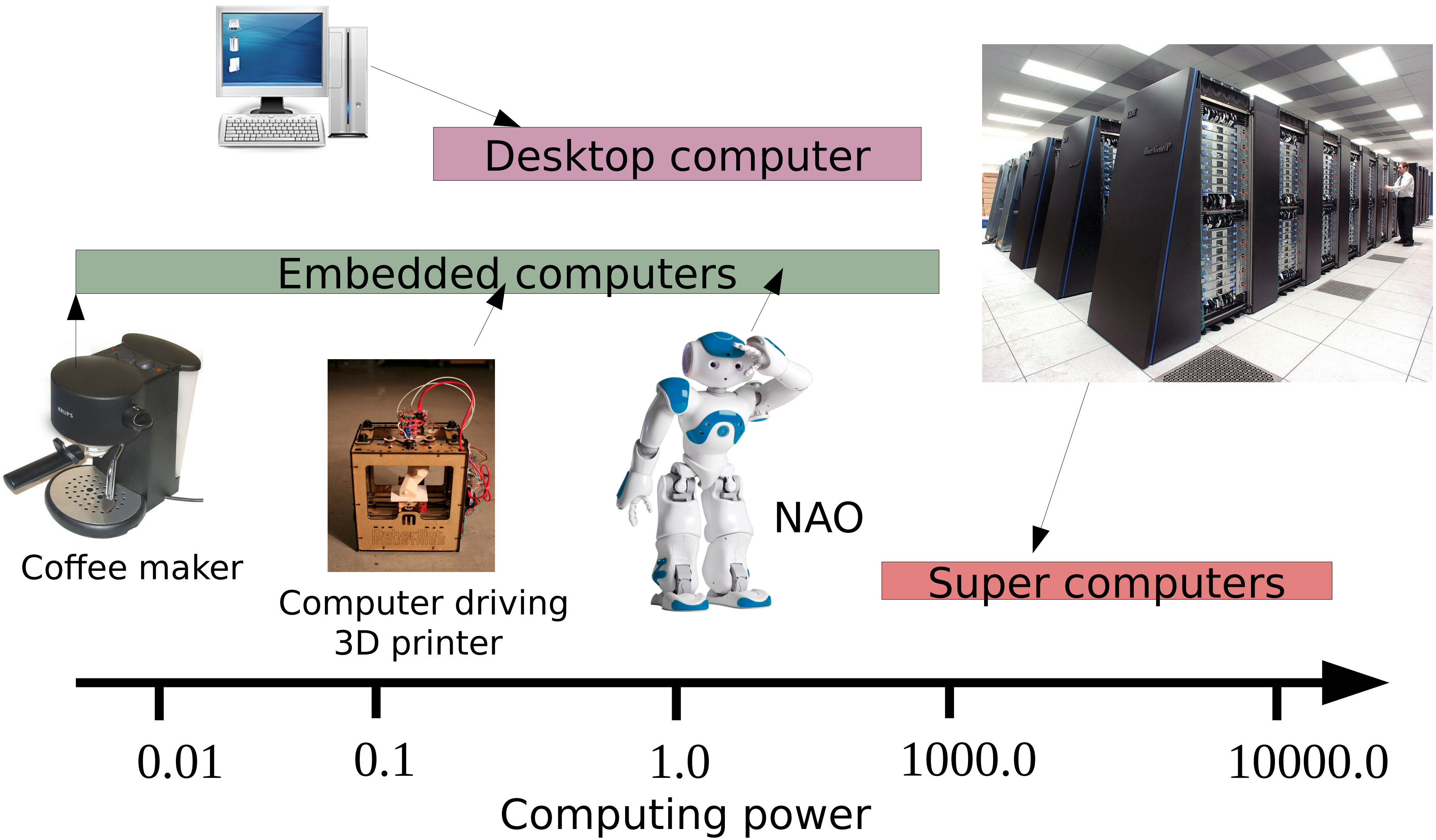
Dr. Roderick MacKenzie
roderick.mackenzie@nottingham.ac.uk
Autumn 2014

www.mm1cpm.com

Outline

- **Recap of last lecture**
- Loops
  - ***for*** loops
  - ***while*** loops
- Nested loops

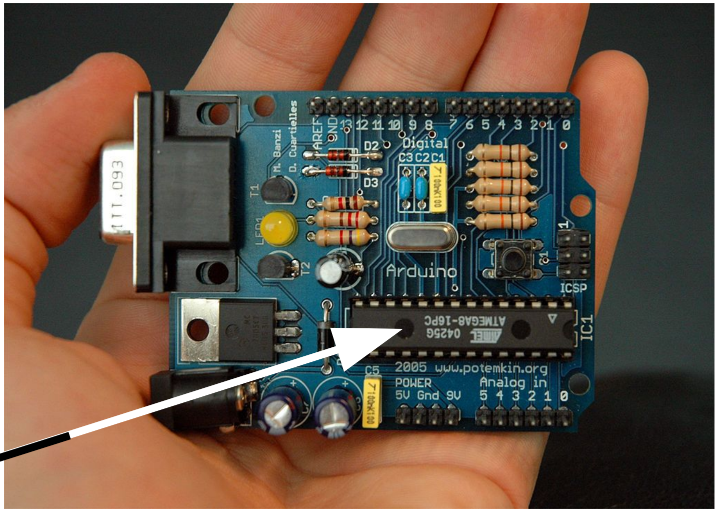
Recap: different types of computer



1.0= the power of a standard desktop PC

Recap:Computers on a chip

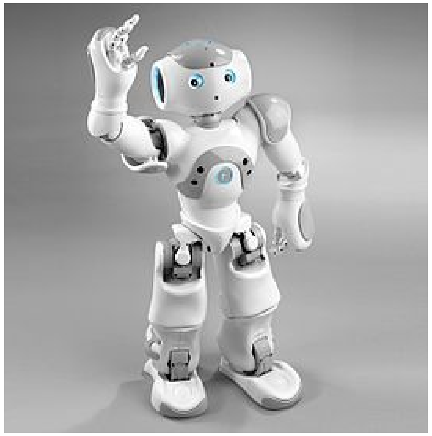
- To reduce cost all components (memory, processor and some storage) computers in embedded devices are often integrated onto a single chip.
- These are the type of computes you will most come into contact with during your professional life.



Processor

Main memory

Storage



Recap: displaying to the screen

- Last lecture we learnt about ***disp***laying text on the screen

```
disp('Computer programming can make me rich!')

Computer programming can make me rich!
```

- We learnt that 'strings' are variables that can hold text.
- All text in MATLAB is surrounded by 'single quotes'

```
a='Computer programming can make me rich!'
disp(a)

Computer programming can make me rich!
```

Recap: Building strings

- sprintf (string print format) can be used to build a string in a given format.

- disp can then be used to display it.

```
a=sprintf('speed=%f m/s fuel left=%f L altitude=%f m ',500, 5000, 1e4);
disp(a)

    speed=500 m/s fuel left= 5000 L altitude=10000 m
```

- The fields beginning with a % are called **format specifiers**.
    - They tell the computer how to format the output

Recap: Keyboard ***input***

Often your program needs to ask the user a question which requires a numeric answer:

How much fuel is needed?

In MATLAB we would do this with the input command

```
answer=input ('How much fuel is needed?');
```

ASCII code

- All computers store text as numbers

a A b

97 65 98

- ***PowerOn*** in ASCII would be

[80 111 119 101 114 79 110]

- Important for talking to robots in mecatronics.

Outline

- Recap of last lecture
- **Loops**
    - for loops
    - while loops
- Nested loops

Linear programs

- Until now our programs have run from the first command to the last command in order:

```
%start of program
x=1                                   %executes 1st
y=2                                   %executes 2nd
x=x*2                                 %executes 3rd
z=x+y                                 %executes 4th
%end of program
```

- This is called a ***linear program*** because you can think of it being executed in a straight ***line***.

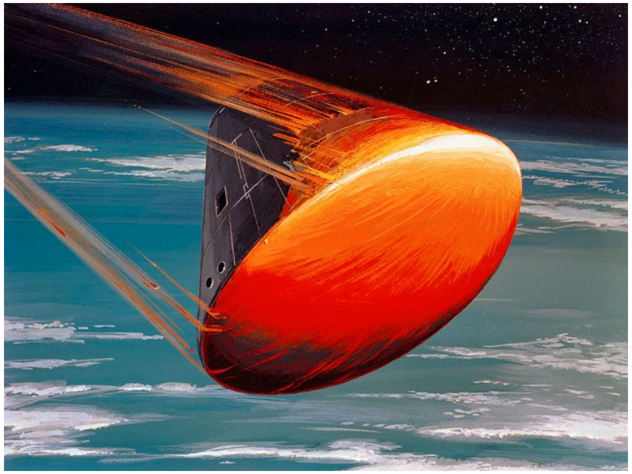
The limitations of linear programs

- Imagine you are designing a program to decide when to open a spaceship's parachute that is reentering the earth's atmosphere.

- If you deploy the parachute too early it will burn up.

- What's the problem with this program?

Images from NASA



Line 1: Start
Line 2: answer=is_speed_slow_enough_to_open_parachute
Line 3: **if** the answer=**yes** open_parachute
Line 4: **if** the answer=**no** do_nothing
Line 5: End

The limitations of linear programs

Genesis space craft

A $260 million hole in the desert - Not so good!

Using a **loop**

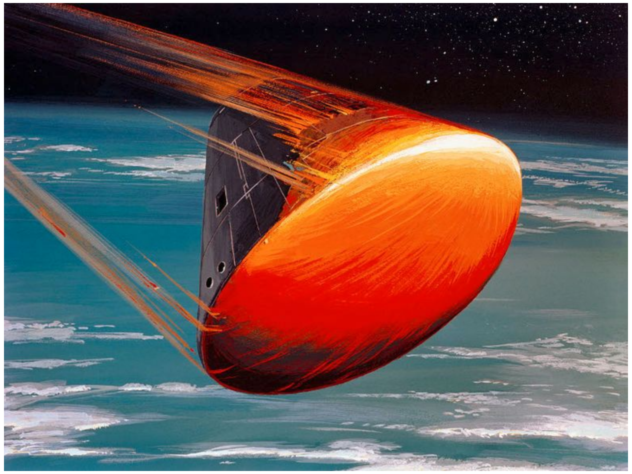
A better program would be:

Loop

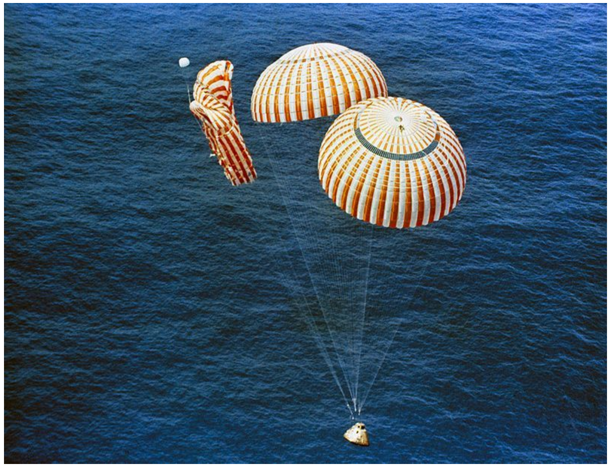
Line 1: Start
Line 2: answer=is_speed_slow_enough_to_open_parachute
Line 3: if the answer=yes open_parachute
Line 4: if the answer=no do_nothing
Line 5: go back to line 2
Line 6: End

This is called a loop.

Images from NASA



How long will this program run for?



Loops in engineering

- Loops are a way of getting the computer to repeat commands again and again.

- The concept is very powerful because it means you only have to write code once and it can run forever...

DaimlerChrysler AG

Loop

Line 1: answer=**has_the_car_hit_an_object**
Line 2: if the answer=yes **open_airbag**
Line 3: if the answer=no **do_nothing**
Line 4: go back to line 1

Your car will have code in it that does this.

Outline

- Recap of last lecture
- Loops
  - **for loops**
  - while loops
- Nested loops

Simple examples of *loops* in MATALB

•Imagine we want to tell MATLAB print 'Hello!!' on the screen 100 times. We could write a script like this:

```
%Print Hello!! x100 to the screen.
disp('Hello!!');
disp('Hello!!');
disp('Hello!!');
disp('Hello!!');
disp('Hello!!');
disp('Hello!!');
disp('Hello!!');
.... repeat 93 more times..
disp('Hello!!');
```

```
Hello!!
Hello!!
Hello!!
Hello!!
Hello!!
Hello!!
Hello!!
........
Hello!!
```

But there is a quicker way...our first loop......

Repeating code using a **for** loop

- We can use a **for** loop to repeat code

Start the for loop

Loop

```
%start of program
for

        disp('Hello!!')          %print to the screen

end
%end of program
```

End the for loop

- All code between the ***for*** and the ***end*** will be repeated.

- What have I forgotten?

Repeating code using a **for** loop

Count using the variable n from 1 to 100

Loop

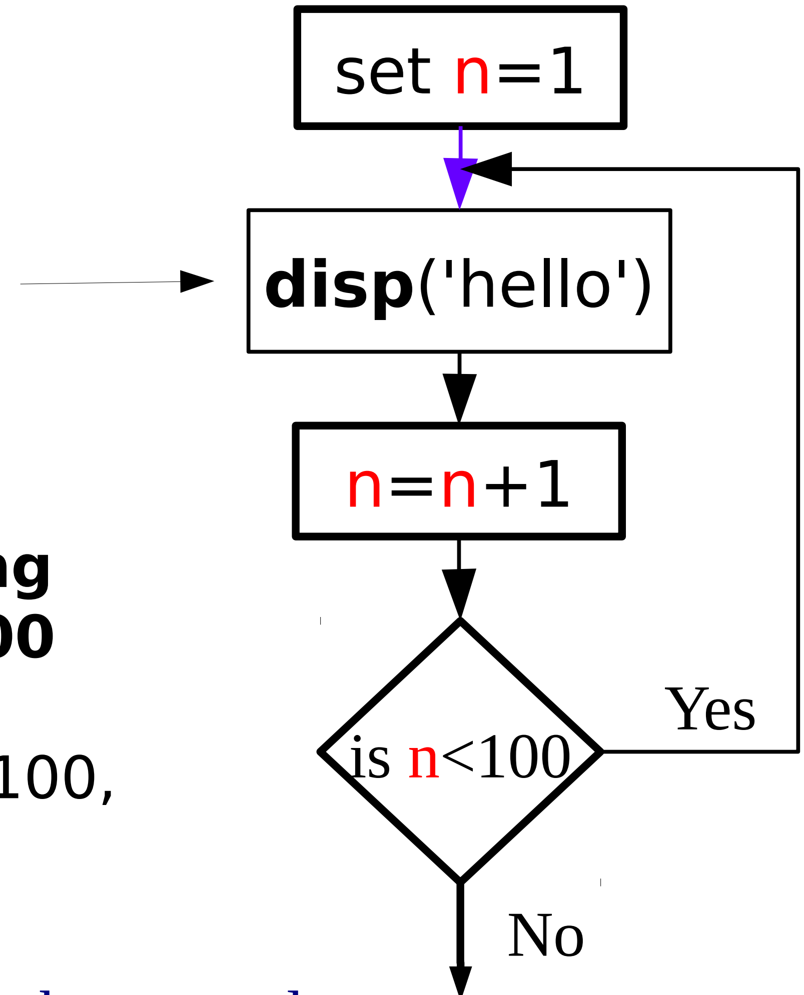
```
%start of program
for n=1:100

        disp('Hello!!')         %print to the screen

end
%end of program
```

- This is a long winded way of saying repeat the loop 100 times
- Let's have a look at this for loop in a bit more detail.....

A for loop in detail

Loop

```
for n=1:100
        disp('Hello!!')
end
```



- **n=1:100** means **Count using the variable n from 1 to 100**

- When n is no longer smaller 100, the ***for*** loop finishes.

Youtube example

The **for** loop in MATLAB

Within the loop we can access n to see what the current count is using the ***disp*** command.

Count using the variable n from 1 to 40

Loop

```
for n=1:40

        disp('Hello!')
        disp(n)

end
```

```
Hello!
1
Hello!
2
Hello!
3
Hello!
4
Hello!
5
Hello!
6
Hello!
7
......
Hello!
40
```

The **for** loop in MATLAB

- You don't have to start the ***for*** loop at 1 you can start it at any number you like.

Loop

```
for n=50:100

        disp('Hello!')
        disp(n)

end
```

```
Hello!
50
Hello!
51
Hello!
52
Hello!
53
Hello!
54
Hello!
55
Hello!
56
......
Hello!
100
```

Youtube example

Another **for** loop example

- Sum the numbers from 100 to 200.

- If we did not know about loops, we could do it like this

```
%Script to sum numbers from 100 to 200
sum=0

n=100
sum=sum+n

n=101
sum=sum+n

n=102
sum=sum+n

n=103
sum=sum+n

.............. repeat 95 more times...

n=200
sum=sum+n
```

Another **for** loop example

But an easier way would be to do this

Loop

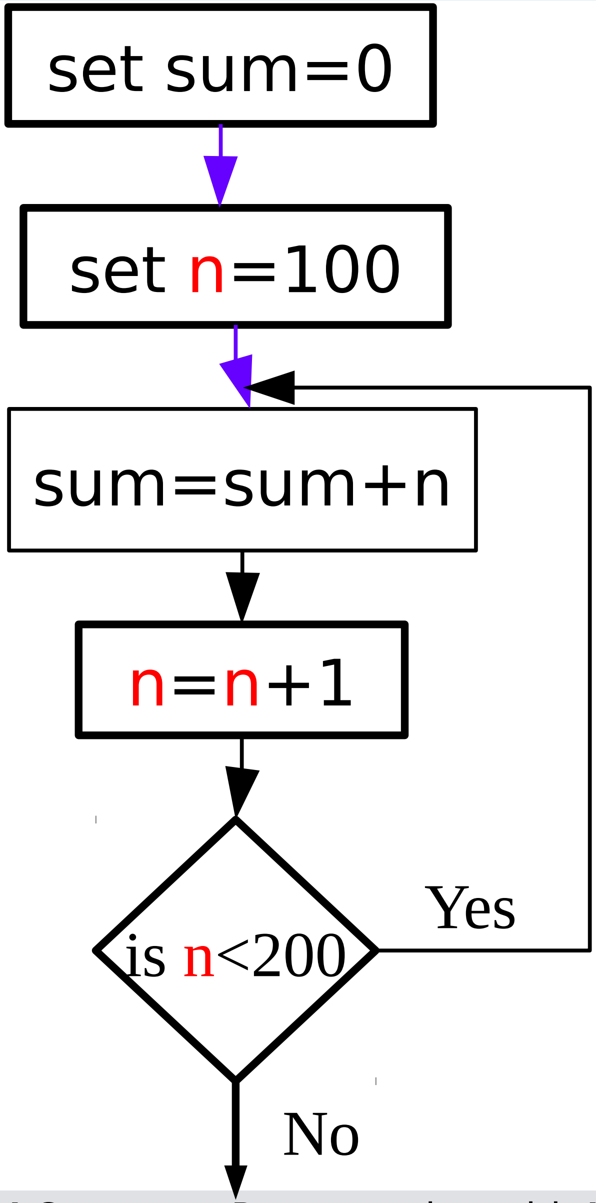
```
sum=0                  %Set variable sum to zero

for n=100:200          %count with n from
                       %100 to 200

    sum=sum+n;         %add variable n to variable sum

end                    %end of for loop, if n has not
                       reached 200 go back to the top.

disp('Finished!');     %Print finished to the screen
```

'n' is just a normal variable, you can give it any name you want, 'a', 'b', 'fred' etc..

A for loop in detail

```
sum=0
for n=100:200
    sum=sum+n;
end
```

Loop



Your go!

```
S=9.5            %set variable S to 9.5
x=1              %set x to 1
x=0.5*(x+S/x) %calculate new value of x from old value of x
x=0.5*(x+S/x) %calculate new value of x from old value of x
x=0.5*(x+S/x) %calculate new value of x from old value of x
x=0.5*(x+S/x) %calculate new value of x from old value of x
x=0.5*(x+S/x) %calculate new value of x from old value of x
.....44 more times
x=0.5*(x+S/x) %calculate new value of x from old value of x
disp('Finished!')
```

- On paper rewrite this program in a shorter form using a **for** loop.
- If you finish quickly try to use your calculator to figure out what mathematical operation the program is performing. Is this an efficient program?

Your go: Answer

Youtube example

```
S=9.5
x=1
for n=1:50
        x=0.5*(x+S/x)
end
disp('Finished!')
```

```
S = 9.5000
x = 1
x = 5.2500
x = 3.5298
x = 3.1106
x = 3.0823
x = 3.0822
x = 3.0822
x = 3.0822
x = 3.0822
.....41 more times
x = 3.0822
```

- The program is calculating the square root of S, but the most important thing is that it got you thinking about loops.

- If you did not get it correct (don't worry) there are lots of more examples in the lab sheets for today.

Outline

- Recap of last lecture
- **Loops**
    - for loops
    - **while loops**
- Nested loops

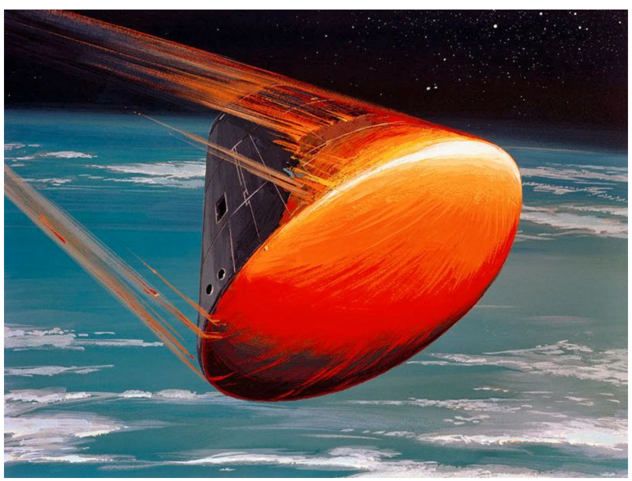
while loops

- Would you want to ride in this space craft?

```
%start of program
for n=1:10
    answer=is_speed_slow_enough_to_open_parachute
    if the answer=yes open_parachute
    if answer=no do_nothing
end
%end of program
```

NB this is a mixture of english and MATLAB (psudo-code).

Image from NASA



28

Another hole in the desert!

• Often we don't know how long our loop will have to run for. Look at the first example again......

Genesis space craft

while loops

- To get around this problem we can use a 'while' loop
- A while loop will continue to loop 'while' a condition is true for example.
- Here is an psudo-code example:

Loop

```
while space_craft_is_in_the_air=yes
    answer=is_speed_slow_enough_to_open_parachute
    if the answer=yes open_parachute
    if the answer=no do_nothing
end
```

- This loop would continue to run as long as the space craft is in the air.

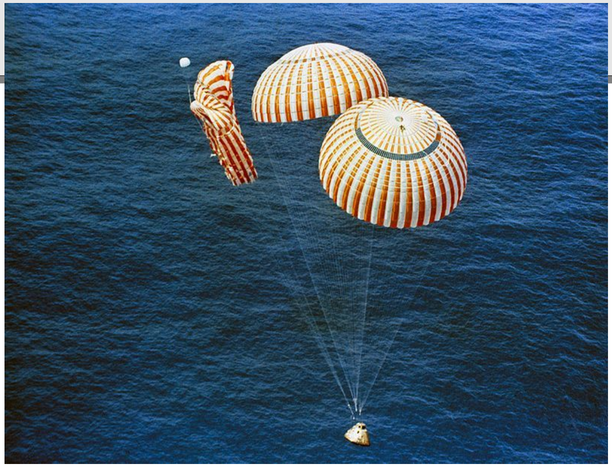


Image from NASA

while loops example in MATLAB

- Here is a real example of a while loop in MATALB.
- This will continue to run ***while*** n is smaller than ten.

Loop

```
n=1                     %set sum to zero
while (n<10)            %start of while loop

    disp(n)             %print sum to the screen
    n=n+1               %add one to sum

end                     %go to the top of while loop if sum<10
```

- The result would be:

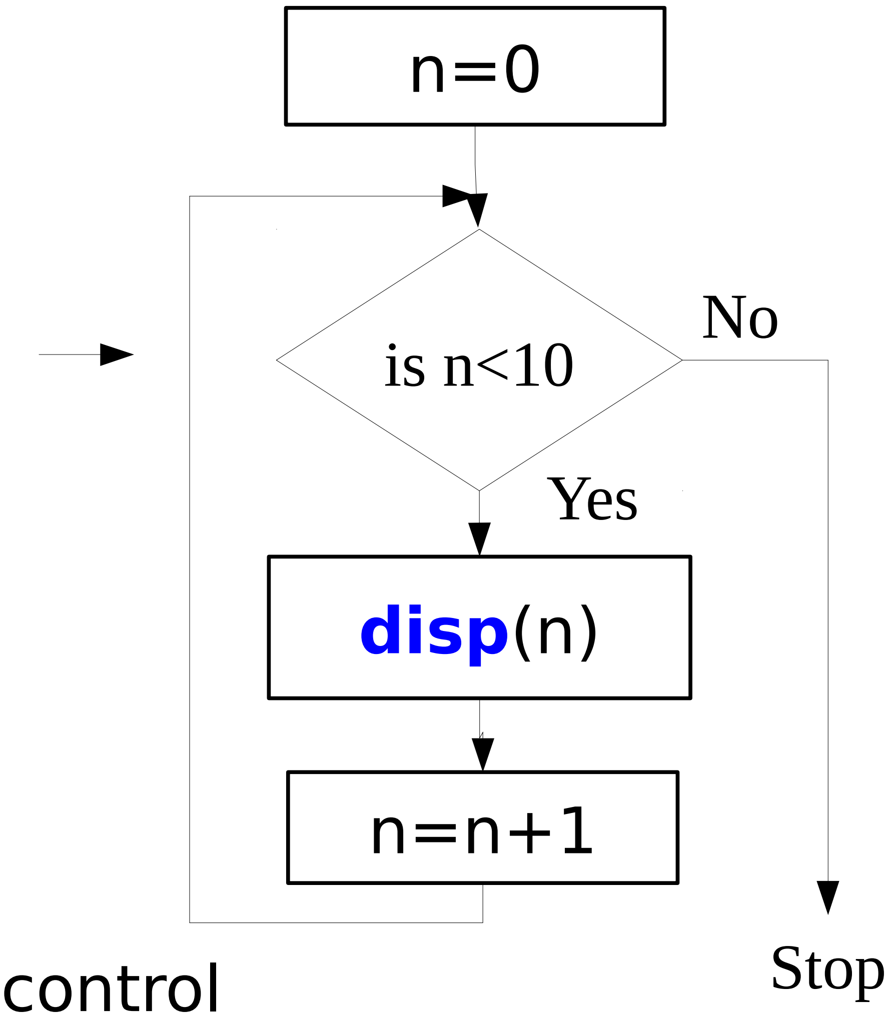
```
1
2
3
.....
10
```

the program would count to ten

A while loop in pictorial form

Loop

```
n=0
while (n<10)

   disp(n)
   n=n+1

end
```



While loops give you more control over the loop than ***for*** loops.

Another while loop example

- **for** loops are good for counting in **steps of 1**
- **while** loops are better for counting in **any size step**.
- Look at the following example which increments the variable **t** by 0.5:

```
t=0
while t<10.0

        disp('The time is ')
        disp(t)
        t=t+0.5

end
```

```
The time is
0.0
The time is
0.5
The time is
1.0
The time is
1.5
The time is
2.0
....
The time is 10.0
```

Youtube example

Conditions

- So far we have used the smaller than < comparison statements to decide if the **while loop** should continue to run.

```
while t<10
.....
end
```

```
while n<10
.....
end
```

- MATLAB also supports other tests, such as equal to, bigger and not equal to etc...

Conditions

Test	Description	Example
<	Less than	while(y<5)
>	Greater than	while(y>5)
<=	Less or equal to	while(y<=5)
>=	Greater or equal to	while(y>=5)
==	Equal to	while(y==5)
~=	Not equal to	while(y~=5)

Note the double equals

- There are questions in the today's example sheet on this.

Outline

- Recap of last lecture
- Loops
    - **for** loops
    - **while** loops
- **Nested loops**

Loops in loops (Nested loops)

- Often in engineering you will need to put one loop in side another loop

Outer loop

Inner loop

```
for x=1:5  %count using x from 1 to 5

    for y=1:5  %count using y from 1 to 5
    a=sprintf('x=%d y=%d',x,y)
    disp(a)
    end

end
```

Nested loops and 2D arrays

```
for x=1:5

        for y=1:5
        a=sprintf('x=%d y=%d',x,y);
        disp(a)
        end

end
```

x=1 y=1	x=2 y=1	x=3 y=1	x=4 y=1	x=5 y=1
x=1 y=2	x=2 y=2	x=3 y=2	x=4 y=2	x=5 y=2
x=1 y=3	x=2 y=3	x=3 y=3	x=4 y=3	x=5 y=3
x=1 y=4	x=2 y=4	x=3 y=4	x=4 y=4	x=5 y=4
x=1 y=5	x=2 y=5	x=3 y=5	x=4 y=5	x=5 y=5

Can anyone think what this could be used for?

Nested loops and 2D arrays

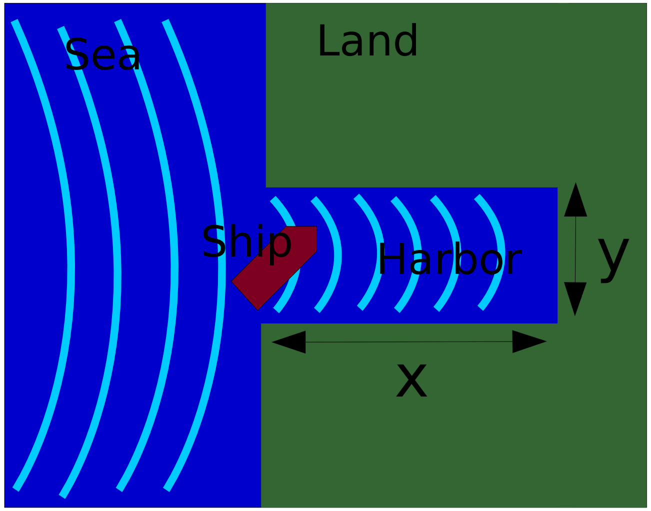
1	2	4	5	6	7
2	6	3	4	4	7
3	6	8	2	4	6
4	5	5	4	3	1
5	5	6	3	2	2
	1	**2**	**3**	**4**	**5**

•Nested loops can be used to scan over 2D arrays....an example.....

But why would I want to do this?

- Often as an engineer you will need to visualize an equations in 2D space.
- Imagine, you are designing a new **60x60 m** harbor for a ship.
- Using fluid mechanics you have worked out that the waves in the harbor obey the following equation:

$$h(x,y)=10\sin(x0.8+t)\sin(y0.2)\exp((x-60)*0.05)$$

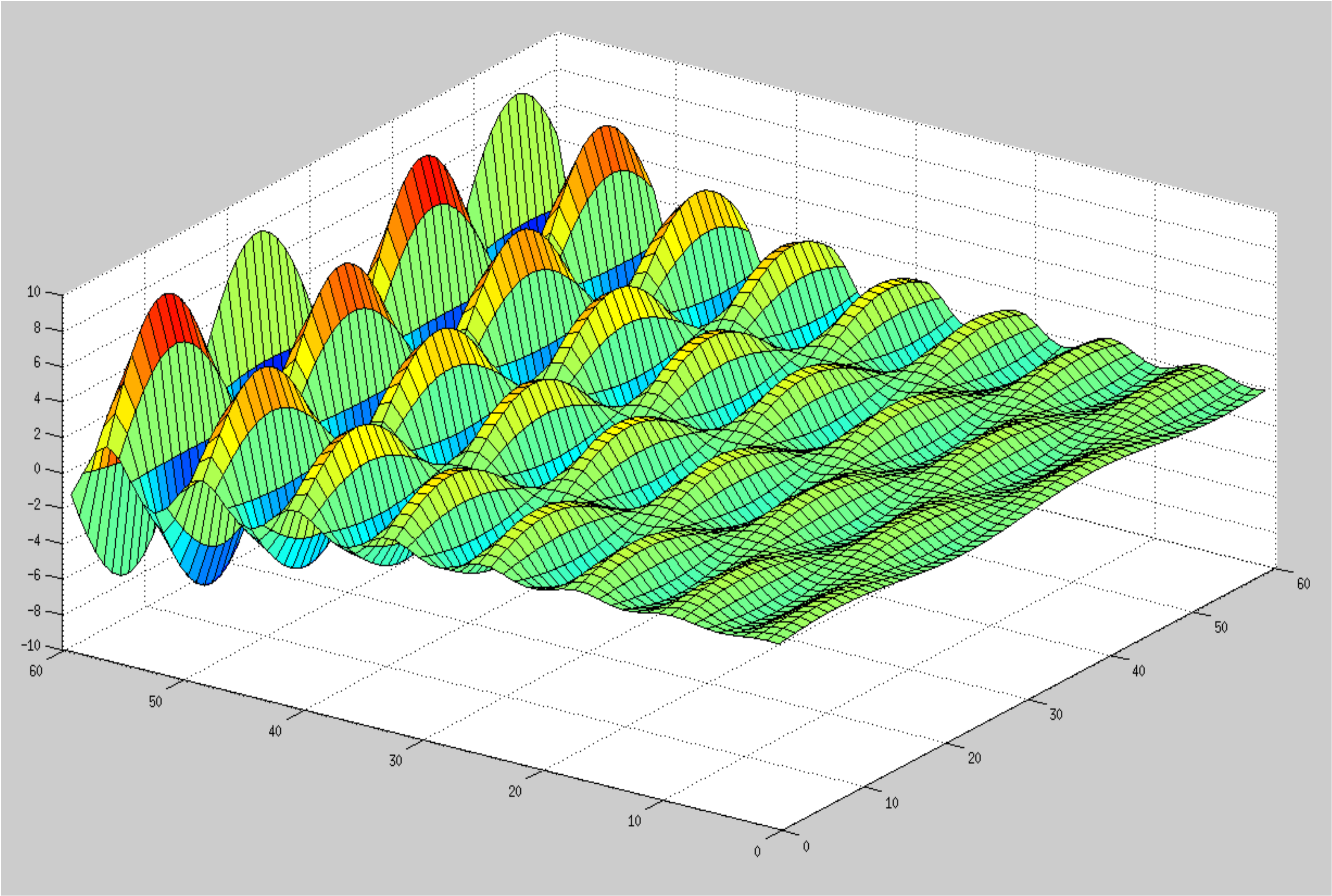


Q: What is the maximum height of the waves at the back of the harbor?

First steps to simulating the real world

```
t=0.0              %set time equal zero

h=zeros(60,60)     %make a 2D array 60x60
                   %of zeros representing the harbor

for x=1:60         %loop over x

   for y=1:60      %nested loop over y
   h(x,y)=10*sin(x*0.8+t)*sin(y*0.2)*exp(-(60-x)*0.05);
   end             %end of nested loop over y

end                %end of loop over x
surf(h)            %plot the wave profile in 3D
```

Simulating the real world



- Now you can evaluate any function in 2D!

Mixing for and while loops

- But at time=0 the waves may not be at their maximum at the back of the harbor.

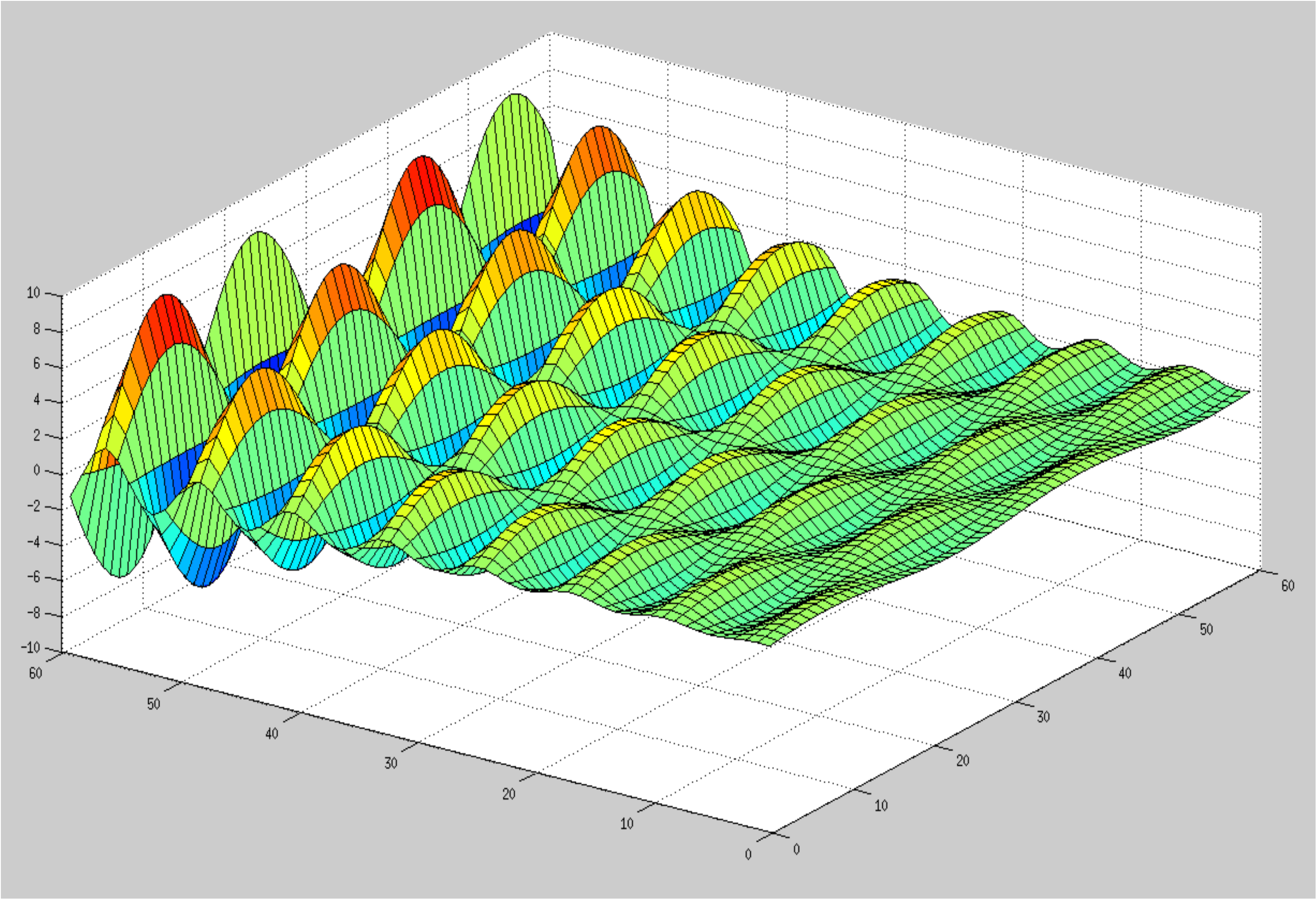
- So let's look at the problem as a function of time by adding a another loop which counts over time.

- If we want to increment time (t) in steps of 0.01s what type of loop would we use?

Simulating the real world

```
t=0.0                %set time equal zero
h=zeros(60,60)            %make a 2D array 60x60
                          %of zeros representing the harbor
while(t<20)
  for x=1:60              %loop over x

     for y=1:60           %nested loop over y
     h(x,y)=10*sin(x*0.8+t)*sin(y*0.2)*exp(-(60-x)*0.05);
     end                  %end of nested loop over y

  end                     %end of loop over x
surf(h)                   %plot the wave profile in 3D
t=t+0.01
pause(0.001)
end
```

Simulating the real world



Youtube example

Summary

- Recap of last lecture
- Loops
    - **for** loops
    - **while** loops
- Nested loops

Debugging: The pause command

•Now that your code is getting more complicated it is more likely that you will make mistakes

•The ***pause*** command is very handy because it can either slow the program down to so you can see variables changing i.e.:

```
t=0
while t<10.0
        disp('The time is ')
        disp(t)
        t=t+0.5
        pause(1)                %pause for one second
end
```

•If you don't put the (1) on the end, pause will wait for a key press - also helpful.

Debugging: Make the problem smaller

- Often if you make a mistake in a complex program it will still run but do something unexpected.

- The best way to find these types of mistakes is to start commenting out code '%' until it starts doing what you think it should do.

- Often this will reveal the bug.

- My other top tip, is to print out all the variables and just check that they are sensible values.

  - You can do this with the ***who*** command or by just typing the variables name.

Summary

- Recap of last lecture
- Loops
    - **for** loops
    - **while** loops
- Nested loops