Faculty of Engineering
MM1CPM

The University of Nottingham

Worksheet 5 – Basic loops

Basic for loops

Q1: Using copy and paste write a script to print the word "Hello world!" to the screen 20 times, now save this as q1.m. Shorten the script to three lines of code using a **for** loop.

Q2: Copy the script you made in Q1 to a file called q2.m . Now edit the script to print the following to the screen:

Hello world!
1
Hello world!
2
Hello world!
3
.....
Hello world!
100

Q3: Copy q2.m to q3.m . Change the script to sum the numbers between 1 and 100. What is the sum?

Q4: Write a script to sum the numbers between 100 and 200.

Q5: Many useful mathematical tasks can be done using iteration. Iteration is often used for difficult to solve mathematical problems in engineering where it is impossible to calculate the answer in one go. When iteration is used, one usually first guesses the answer to a problem. Then one applies a mathematical equation to get a better guess of the answer. The improved guess is then fed back into the mathematical equation again and again to improve the guess. After a number of iterations one ends up with a good answer to the problem.

a) An example of an iterative algorithm is given below. Copy and paste the following code into a new MATLAB script.

S=9
x=100
x=0.5(x+S/x)*
x=0.5(x+S/x)*
x=0.5(x+S/x)*
x=0.5(x+S/x)*
x=0.5(x+S/x)*
x=0.5(x+S/x)*
x=0.5(x+S/x)*
x=0.5(x+S/x)*
x=0.5(x+S/x)*
x=0.5(x+S/x)*
x=0.5(x+S/x)*
x=0.5(x+S/x)*
x=0.5(x+S/x)*
x=0.5(x+S/x)*
x=0.5(x+S/x)*

x=0.5(x+S/x)*

What mathematical operation this code doing? If you can't guess, try changing 'S' to other numbers and see what happens. Did you know that this is used in many computers. After how many iterations did the code find the correct answer?

b) Convert this program into a **for** loop. And play with the value of 'x' and 'S' – what effect do these values have on the rate of convergence? Why could this code waste computing time?

Basic while loops

Q6:
a) Write a new script called q6.m which will count from 0 to 10.0 in steps of 0.5 using a **while** loop. Use the variable t to count. Hint: There is an example of how to do this in the notes. Display the value of t each time the loop runs.

b) Now edit the script to start counting at -10.0, and to count in steps of 0.01.

c) What happens if you delete the line of code 't=t+0.1', and replace it with the code '`disp(t)`'? Why is this? Hint 'ctrl+c' will stop the program running.

Q7: Write a new **while** loop to sum the numbers between 0 and 10.0. Hint: There is an example of how to do this in the notes.

Q8: Write a script using a **while** loop to count from 1000 to 2000 in steps of 2.

Q9: Write a script using a **while** loop to sum the numbers divisible by three between 3 and 1000000. Hint: You need to count in steps of three.

Q10: Write a script to ask the user to enter two variables, 'start' and 'finish'. Then use a **while** loop to calculate the product (i.e. multiply) of the whole numbers between these two values.

Nested loops

Nested loops simply mean one loop inside another, they are used all the time in Engineering where you are handling 2D data such as microscope images or video data. This section of the example sheet will let you practice using nested loops.

Q11:
a) Make a new script and initialize two variables, x and y as zero. Then uses **sprintf** and **disp** to write the following text to the screen.

“x=0.0000, y=0.0000”

b) Place the the **sprintf** and the **disp** commands within a **for** loop that counts from 1 to 10 using the variable x. What happens to the variable x and what happens to the varable y?

c) Now place the entire **for** loop counting in x (including the **sprintf** and **disp**) command within another **for** loop counting in y from 1 to 10. What now happens to x and y? This is your first

nested loop! Nested simply means one inside the other.

Q12:
a) Copy the script file q11.m to a file called q12.m. Now define an array of 10 by 10 of zeros called 'data' at the beginning of the script. Delete the **sprintf** and **disp** lines, and replace them with the line of code 'data(x,y)=sin(x)'. What does this line of code do?

b) Now edit the script so that the array data is plotted after the **for** loops have finished. Use the command **surf**(z, 'linestyle', 'none') to plot the data. What does the 2D graph look like?

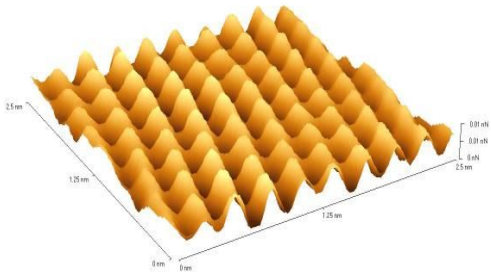

Figure 1: A very sensitive microscope image of atoms on the surface of a metal.

c) Replace the **sin**(x), with **sin**(x)**sin**(y) what happens now? This pattern could represent atoms on the surface of a metal – see figure 1.

Q13: Copy the script file q12.m to a new file called q13.m and change and make both for loops count from 1 to 20 instead of 1 to 10. Also change the size of the array 'data' to be 20x20. Now replace ***sin****(x)****sin****(y)* with $e^{-a(x-10)^2-a(y-10)^2}$, where a is a variable set to 1.0. Reduce the value of the variable slowly until it reaches 0.01. What does the a control on the graph? This shape is called a 2D Gaussian. It is found all over nature, for example when light first leaves a laser ,it has a very narrow beam which can be modeled with a=1.0, as the light travels through air it spreads out and the beam profile can be better modeled with a=0.01. Now make up three of your own 2D functions to plot using the code for this example as a base.

Q14: In this question we are going to use nested **for** and **while** loops together to simulate a ball being dropped from a height of h=10 meters and bouncing on a surface at height h=0 m.

a) Initialize the following variables with the corresponding values:

Variable	Value	Meaning
h	10	Ball height
g	-9.81	Gravity
dt	0.1	Time step
v	0.0	Velocity

b) The following equations of motion can be used to calculate the height and velocity of the ball at time Δt in the future.

$$h = h + \left(v + \frac{g\Delta t}{2}\right)\Delta t$$

$$v = v + g\Delta t$$

Implement these equations in MATLAB, copy and paste both of them four times, one after each other. What happens to the v and h? This represents the ball falling. Now delete all but one pair of

the equations and place a **for** loop around them that counts from 1 to 1000. What is the minimum height the ball reaches? What is wrong with this height?

c) At the beginning of the script initialize a variable called count to one. Just after you calculate the new velocity and height in the **for** loop add one to count.

d) Initialize a variable called t (for time) at the beginning of the program. Just after you calculate the new velocity and height add Δt to t. This will keep track of the current simulation time.

e) Append each new value of t and h in two arrays called store_t and store_h, then at the end of the program, plot store_t against store_h. This should give you a plot of the ball height against time.

f) Change the **for** loop into a while loop that only runs while h>0, rerun your script. What happens to the ball now?

g) Place your new **while** loop within a **for** loop which counts from 1 to 10. This will make the ball bounce 10 times. When the ball hits the ground, it's velocity changes direction and looses 10% of it's velocity. Therefore after the end of the **while** loop multiply the velocity by -0.9. Also note that the **while** loop will only run if h>0.0. Therefore just after multiplying the velocity by -0.9, set the height of the ball to 0.01 m. If you run your script now you should get a plot of a ball which bounces forever.

h) Try changing the value of -0.9 to other values – what happens to the ball?

j) Move the plot command from the end of the script to within the **while** loop and make MATLAB pause (0.001) seconds between each iteration of the loop. If you run your code now you should get an animation of a bouncing ball!

Q15: In this question we will learn about simple image processing in MATLAB. In lecture 2 you learnt to load data from text files using the ***load*** command. MATLAB also has commands to load and display jpeg images. Download the file called 'cat.jpg' from moodle, copy this to your z: drive and double click on it to check you have the correct file. In MATLAB, make a new script called cat.m and save this to your z: drive.

Figure 2: The file cat.jpg, you can download this from moodle

a) To load the image into an array 'a' and display it, type the following commands into your script.

```
a = imread('cat.jpg')
image(a)
```

The command ***imread*** will load the image cat.jpg into an array 'a' and ***image*** will display it.

b) Using the ***size*** command, find out what dimensions the array 'a' has? Is it 2D or 3D?

c) Computers and televisions display ***all*** colors using a combinations of Red, Green and Blue light. If you have ever looked really closely at an old TV screen, you will have seen the little red, blue and green dots which make up the image, this can be seen in figure 3. If only the red dots are turned on, the image will be red. If only the green or blue dots are turned on the image will be green or blue respectively. By changing the relative intensity of the dots millions of colors can be made.

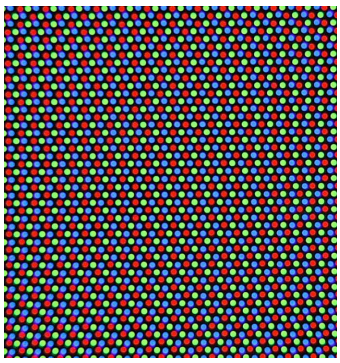

Figure 3: The red, green and blue dots used to make up a television picture.

The command ***imread*** in your script reads a color image and then produced three 2D arrays stacked on top of each other, forming a 3D array. The first 2D array represents red pixels, the second 2D array represents green pixels and the third array represents blue pixels. The intensity of the colors are stored using values from 0 to 255, where 0 represents the color being turned off and 255 represents the color being at its maximum intensity.

If you wanted to change the pixel in the array 'a' to a pure red pixel, you would type

```
a(10,10,1)=255      %set the red pixel at 10,10 to the maximum intensity
a(10,10,2)=0        %set the green pixel at 10,10 to the minimum intensity
a(10,10,3)=0        %set the blue pixel at 10,10 to the minimum intensity
```

Add these lines to your script and try to find this red pixel in the image (hint: look at the top left) Or for example if we wanted to change the pixel at 10,14 to green we could add the following commands to our script

```
a(10,14,1)=0        %set the red pixel at 10,10 to the minimum intensity
a(10,14,2)=255      %set the green pixel at 10,10 to the maximum intensity
a(10,14,3)=0        %set the blue pixel at 10,10 to the minimum intensity
```

Again, add these commands to your script and see what happens, try to find the green pixel in the image.

d) By changing the array 'a' pixel by pixel, draw a very tiny smiley face on the top left of your image.

e) By looking at the image and using the zoom tool, find the location of of the cat's right eye. Then write two nested **for** loops which draw a green box over the cat's right eye. (Hint, the color green is given by Red,Green,Blue=0,255,0).

f) The cat has a blue and a green eye. Change your script you wrote in part e to change the color of the right hand eye to blue. You can do this by simply setting the value of the blue pixel at each location to the value of the green pixel. The color of the right hand eye won't perfectly match that of the left but it will definatly be more blue than it was before.

The above example is a very simple example of image processing, we will do more advanced image processing in later example sheets.

Q16 (a tricky but fun question): Often when working on large projects in industry you will be presented with a piece of code that you have never seen before nor really understand exactly what it does. The best strategy for trying to understand such pieces of code is to first pick out the commands that you understand then try to identify the **for** and **while** loops. Then try to make a sensible guess as to what the program may do. Look at the piece of code on the right.

```
clear;
x_pos=1;
x=-1
y=-1.0
while (x<1.5)
   y_pos=1;

   y=-1.5;

   while (y<1.5)
      z=0;
      c=x+i*y;
      count=0;

      while ((abs(z)<2)&&(count<255))
         z_next=z*z-c;
         z=z_next;
         count=count+1;
      end

   out(x_pos,y_pos)=abs(z);

   y_pos=y_pos+1;
   y=y+0.005;
   end

x_pos=x_pos+1;
x=x+0.005
end

colormap jet;
surf(out,'EdgeColor','none','LineStyle','none')
view(2)
```

a) How many **for** and **while** loops can you see?

b) Just by looking at the code try to guess what the minimum and maximum values of x and y that will be reached.

c) In steps of what size are 'x_pos' and 'y_pos' incremented? What is the purpose of these variables?

d) What is the maximum number of times the inner most **while** loop is run?

e) Is 'c' a real or imaginary number? The variable 'c' is reinitialized every time the second while loop runs. What values will 'c' be initialized to?

f) In your head try to figure out what the various loops might do – it's quite tricky to do this so, I don't expect you to guess exactly what it does. If after having looked at a piece of code for a while and you are still unsure what it does the best way to find out is to run it! Make a new script file called mystery_code.m . Then copy and paste the code into the script file. Click run and a plot should be generated. Do you recognize the image? Have you seen it before on any psychedelic pictures? Does this help you figure out what the code is doing? Now add comments to the mystery_code.m file containing guesses as to what each line does. Try playing with the various constants in the script to see what they control.