ELEE 4200

Autonomous Mobile Robots

Homework No:0

MATLAB Refresher

Author 1 :
Name : Wang Jingya
T.No.: 02157119

Author 2:
Name : Liang Mi
T.No.: 02157118

Instructor:
Dr. Mohan Krishnan

September 12, 2019

1 Abstract

To refresh our Matlab skills, and also preparing for drawing the path of a robot, the homework asked us to draw Olympic rings

2 Introduction

- We should plot a image which looks like Olympic rings,

  There are some requirements as follows:

  1.The radius of each circle should be random, and it should be calculate by radius = 4*scale, scales is a number between 0 to 1. Additionally, the scales should be stored in a matrix ;

  2. The center of each circles should be fixed, what's more, all the centers should be stored in a matrix which has 5 rows and 2 clolumn;

  3. The whole shape of the figure should be looked as a "W";

  4. Not every circles should be intersect. The upper line circles should be connected to the under lines, how ever, the parallel line's circle should not be connected.For example, in Figure 1:

  The blue one must intersect with the yellow one, and the yellow one must be intersect with the black one, however, the blue one should not be connected with black one.

Figure 1: Olympic rings

5. The most importantly, we should use function in our main.m files.

3 Methodology

Our group have come up with two methods, and both have one code to solve the problem:

- A.The code written by Jingya wang:

  Logic is:

  (1)Firstly, Identifying the circles' centers.

  The requirement is the center of each circle is fixed, additionally,the centers should be output as a matrix;

  so I create a function, in order to be called in the main.m file,and assuming the centers by ourselves. and then, storing all the data in a matrix.

  (2)Secondly, producing another function which is used to create a radius matrix.

  But there is an additional requirement is that the radius should be calculated by the equation:

  radius = 4 * scale;

  and the scale should be in the interval from 0 to 1;

  Therefore, the radius function will act as two roles:

  1.) Judging whether all the elements in scale matrix is around 0 to 1;

  2.) Outputting the radius as a matrix;

  (3)Thirdly, creating another function which is used to calculate, and the equation is:

  $$x = x0 + R * cos(\theta)$$
  $$y = y0 + R * sin(\theta)$$

  [1].

  (4)Fourthly, in the main.m file judging whether the circles are intersecting;

  The requirements are specified written in the introduction section, so I just write the basic idea here. The basic perception is comparing the distance between two circles' center with the sum of their radius to decide whether it is intersect.

For example, Look at the Figure 2:

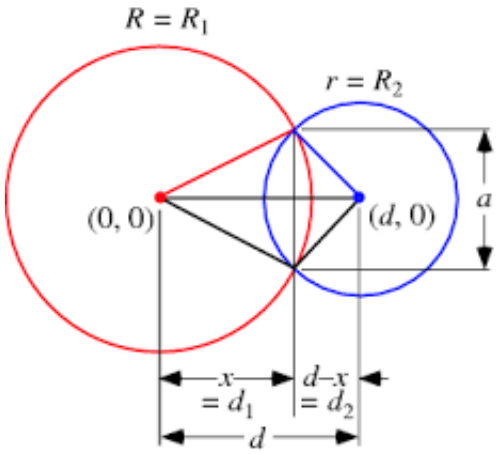


Figure 2: Two circles intersect

if R1 + R2 <d, it means the two circles are intersecting.

if R1 + R2 >d, it means the two circles are staying away.

Therefore, using this basic theory to judge if the blue circle connect with the yellow one, the yellow one connect with the black one, the black one connect with the green one, and the green one connect with the red one.

(5)Fifthly, Plotting all the images, to satisfied the requirement that show the circles one by one, I have three solutions:

1. Use the Matlab command of subplot(x,y,n); To add one circle each step and print on one page.

2. Use the Matlab command of figure(n) and hold on ; To print five figures and add one in each figures.

3. Use the Matlab command of pause(seconds); When print the figures to let it hold on seconds.

- B.The code written by Liang Mi:

  Logic is:

As far as I'm concerned, because of the requirement,I should draw the circle one by one. When a circle was being drawn, testing whether it meet the requirements. If the circles are suitable then draw the next circle. If the circles are not suitable, like not intersect, then clearing all the circles have drawn and restarting the "for" loop to build the first circle again. The loop will continue and restart until getting all the five circles in a suitable condition.

In my code the "main.m" is a "for"loop in a "while" loop. The "for" loop is used to build center, drawing the circle and test circles has been built by three functions in the loop. After once "for" loop, a circle will be built and tested. If the test failed, the circles has been drawn will be clean and restart the "for" loop. "While" loop is used to build an environment that the program will been circulating until getting a right result and restart the "for" loop.

There are three function in the loop.

The first one is "function center radius".

It is used to build the center and radius.

The second is "function plot".

It is used to draw the circle.

The third is "function test".

It is used to test if the circles are suitable. The function will return a data. After the text if the result is right, the return data will stop the "while" loop and show the result. If it is wrong, the return data will break the "for" loop immediately and restart the "for" loop.

4 Results

A.The result of Jingya Wang's code:
There are three version of my code; (1.)The first plotting method: The first one is to use **subplot command**; So we would see five figures on one page, and each figure has one more circle than the previous one.

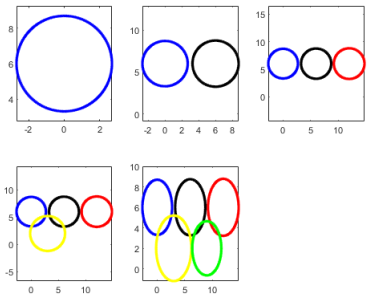
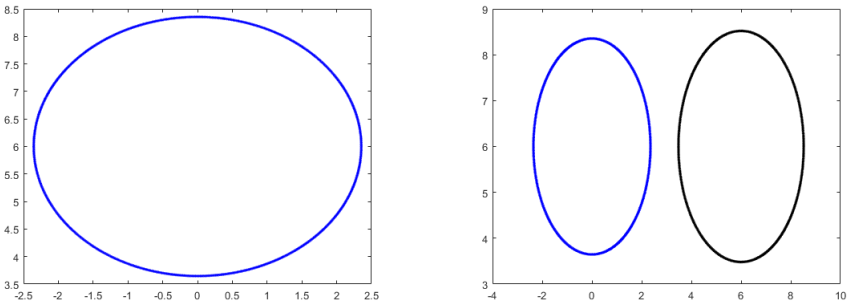


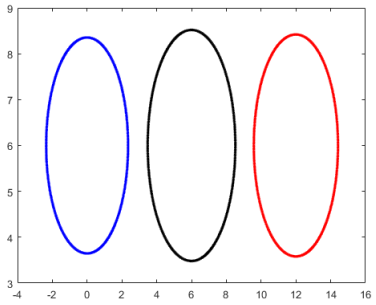
Figure 3: Solution1's figure

(2.)The second plotting method: The second method is to use **figure(n)** command with **hold on** command. So we would see five page figures and on each page, there is a circle more than the previous one.

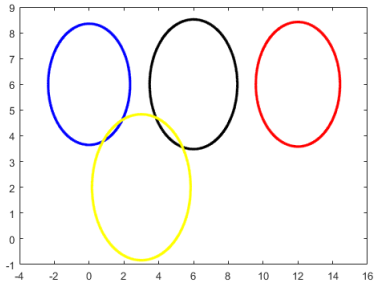


(a) Solution2's figure1 (b) Solution2's figure2

Figure 4: Solution2 's figure1 and figure2

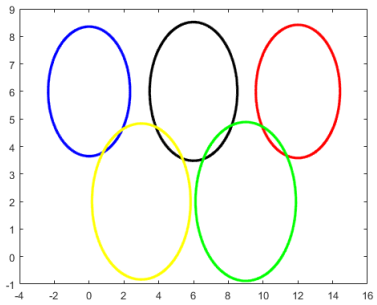


(a) Solution2's figure3

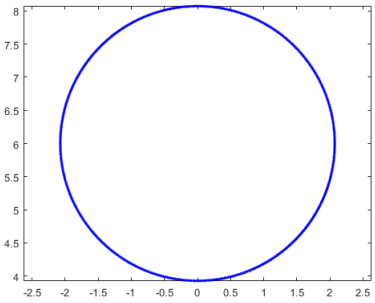


(b) Solution2's figure4

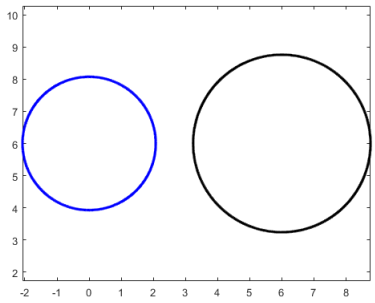
Figure 5: Solution2 's figure3 and figure4



(3.)The third plotting method: The third method is to plot figures by **pause(n)** command. And the result is printing on one page and we can see the circles add one by one and the circles will occur in a 'n' seconds interval.

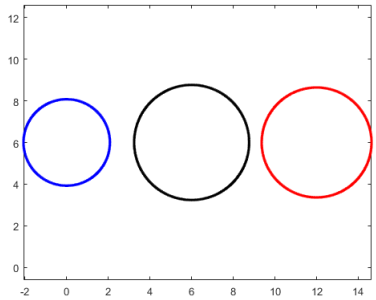


(a) Solution3's figure1

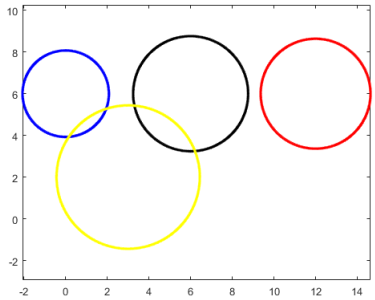


(b) Solution3's figure2

Figure 6: Solution3 's figure and figure2



(a) Solution3's figure3



(b) Solution3's figure4

Figure 7: Solution3 's figure3 and figure4

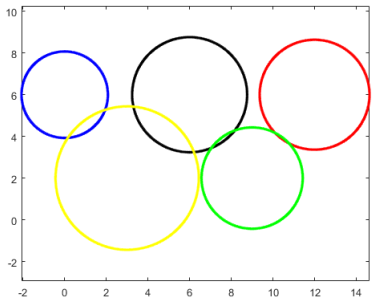


Figure 8: Solution3's figure5

B.The result of Liang Mi's code :

In my code, the circles are appear one by one. Because the radius be created by random. So, a suitable Olympic be built after thousands of times try. A figure on the below (Figure 10) is one of right result. Each circles intersect last circle and the circles in same line not intersect.

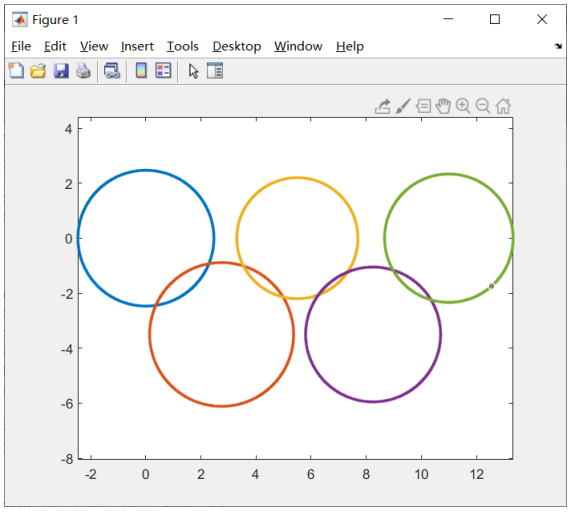
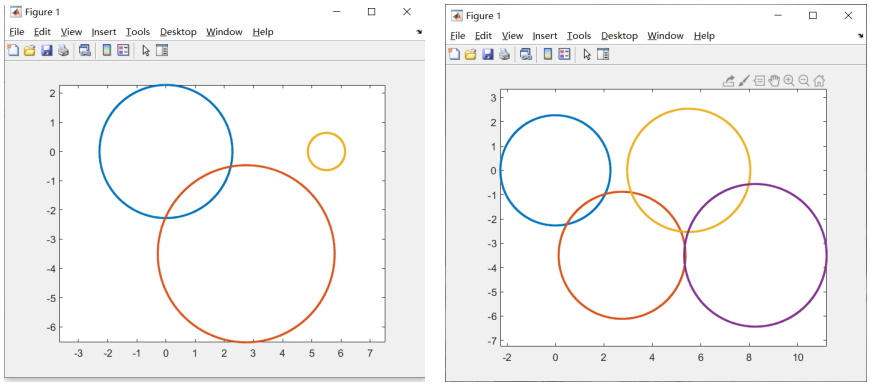


Figure 9: Final result

There are two fail results on the below(Figure 11 and Figure 12). In the first figure on the below(Figure 11), the third circle are too small to intersect the second circle. In second figure(Figure 12), the two circles at the bottom are intersecting. So those two result not meet the requirement. In the program, the fail result will be clear immediately and restart the "for" loop to get a new result until getting a right consequence.



(a) fail result in running (b) fail result in running

Figure 10: Solution3 's figure11 and figure12

5 Discussions and Conclusion

- We have achieve all the goals and the result are also same as what we predicted before.

  And we think the most difficult part of the task is:

  1. judging the scales' elements all in 0 to 1.
  2. plotting the circles one by one.

  The problem we met is :

  There is one problem which Jingya Wang ever have appeared is that :

  Because the run speed is too fast to let the viewer to see the circles adding one by one,so we change the **hold on**. Finally, we found the command **pause(n)** is working.

  Mi Liang's problem is can not show the circle one by one. The way i deal with the problem is finding information on the internet. After look through some essays and the code on the web, i know that, by default, the MATLAB only show the final result. It means that you can not see the process that the circle be drawn one by one. But, we find the code "drawnow". Adding this code after "plot", the figure will showing immediately and the problem be solved.

References

[1] Math open source. http://www.mathopenref.com/coordparamcircle.html.

A Appendix

A.1 List of .m files

- MATLAB file Wang Jingya's code:

  The first version:

  In the main.m file, the code is:

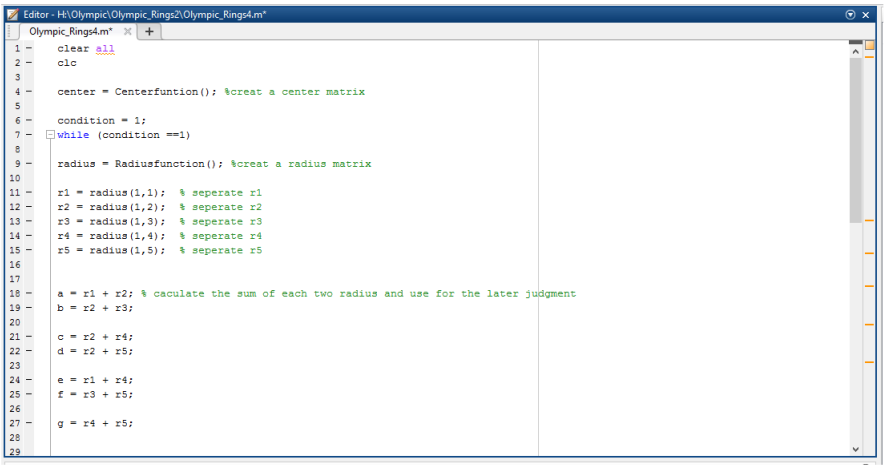


Figure 11: Solution1's code first part

Figure 12: Solution1's code second part

Figure 13: Solution1's code third part

In the centerfunction:

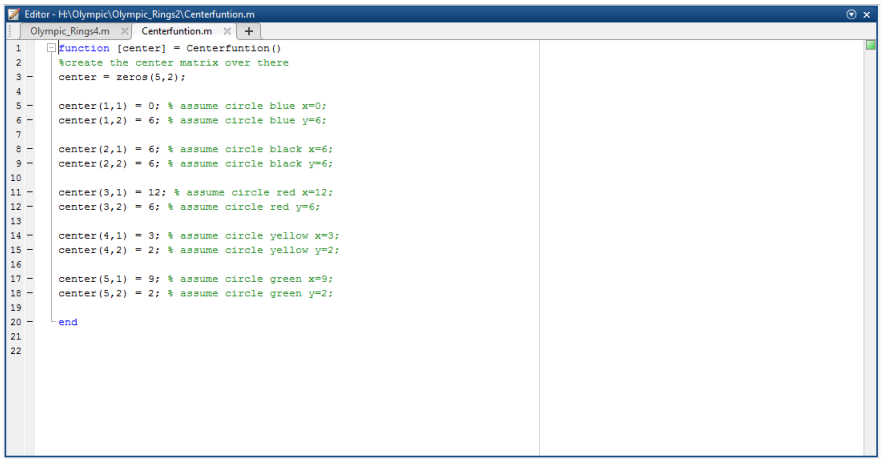


Figure 14: Solution1's center function part

In the radiusfunction:

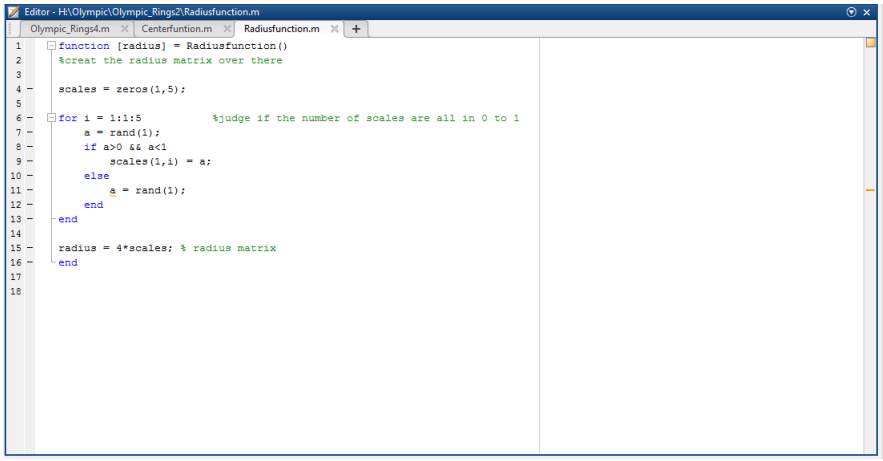


Figure 15: Solution1's radius function part

In the Calculate function:

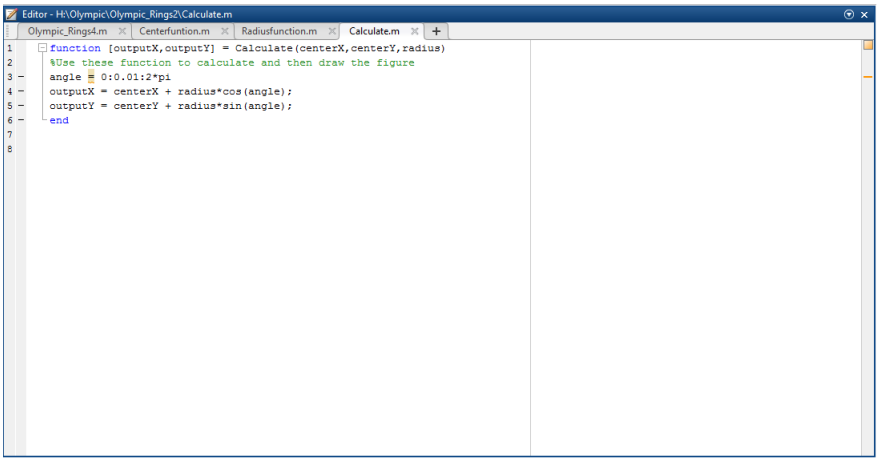


Figure 16: Solution1's calculate function part

The second version: the second version's function is same as the first version, and the differences are in the main.m file parts. To be more specific is the plotting parts use other two different ways.

In the second version's main.m file, the plotting part code is:

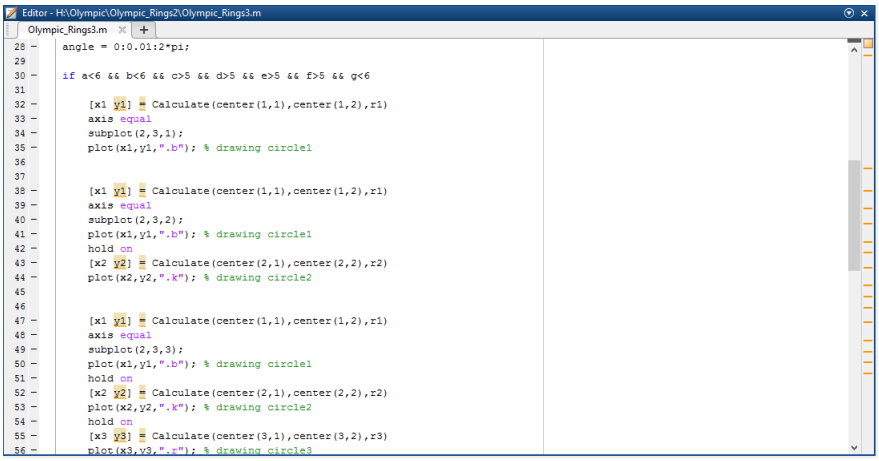


Figure 17: Solution2's plotting code first part

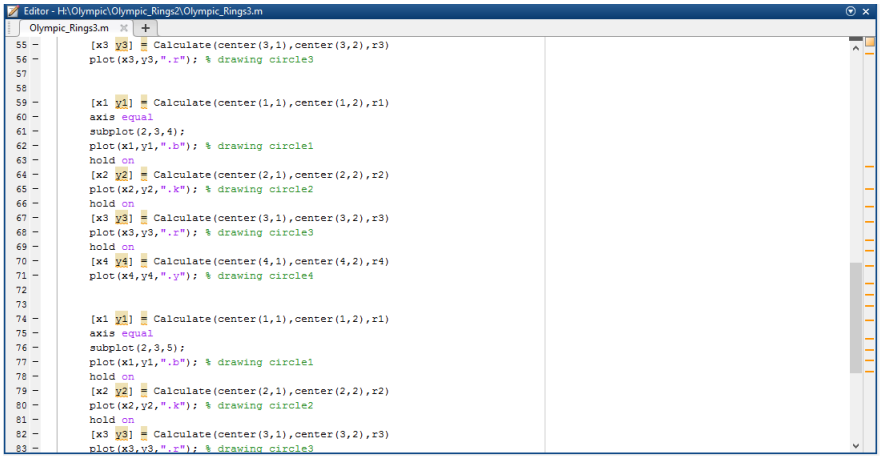


Figure 18: Solution2's plotting code second part

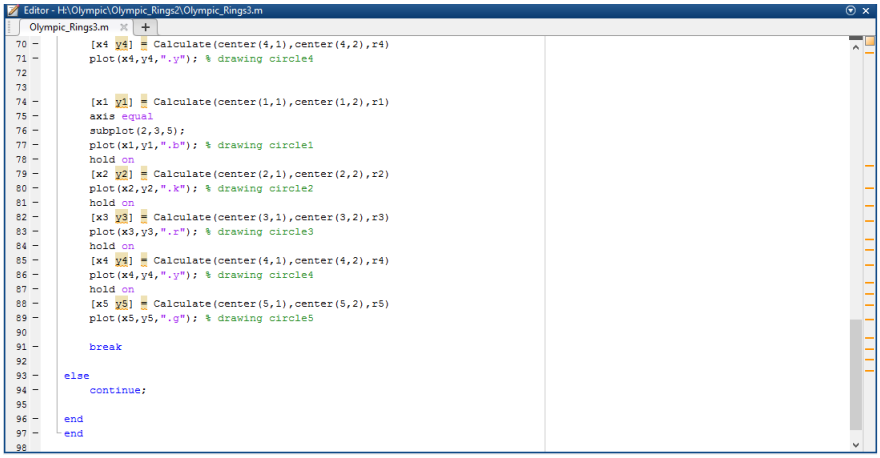


Figure 19: Solution2's plotting code third part

The Third version:

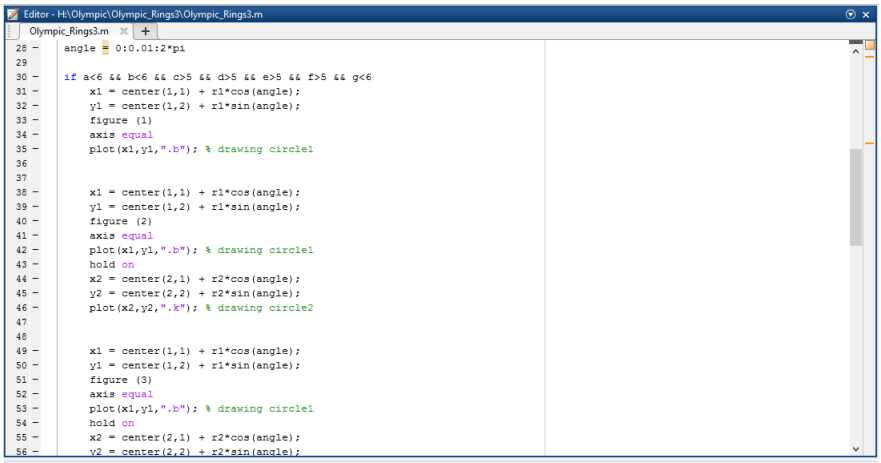


Figure 20: Solution3's plotting code first part

Figure 21: Solution3's plotting code second part

```
angle = 0:0.01:2*pi

if a<6 && b<6 && c>5 && d>5 && e>5 && f>5 && g<6
    x1 = center(1,1) + r1*cos(angle);
    y1 = center(1,2) + r1*sin(angle);
    figure (1)
    axis equal
    plot(x1,y1,".b"); % drawing circle1


    x1 = center(1,1) + r1*cos(angle);
    y1 = center(1,2) + r1*sin(angle);
    figure (2)
    axis equal
    plot(x1,y1,".b"); % drawing circle1
    hold on
    x2 = center(2,1) + r2*cos(angle);
    y2 = center(2,2) + r2*sin(angle);
    plot(x2,y2,".k"); % drawing circle2


    x1 = center(1,1) + r1*cos(angle);
    y1 = center(1,2) + r1*sin(angle);
    figure (3)
    axis equal
    plot(x1,y1,".b"); % drawing circle1
    hold on
    x2 = center(2,1) + r2*cos(angle);
    y2 = center(2,2) + r2*sin(angle);
```

Figure 22: Solution3's plotting code third part

- MATLAB file Mi Liang's code

```
center=zeros(5,2); %%create matrix for 5 center
radius=zeros(5,1); %%create vector for 5 radius
m=1;%%uesed to control the "while"loop
f=0;%%used "f++" in loop to show the times has run
while m
for i=1:5
    [center radius]=function_center_radius(center,radius,i);%this function is used to create center and radius matrix
    function_plot(center,radius,i);%%this function is uesd to draw the circle
    n=function_text_circle(radius,i);%%function ues to text plot(return a data n to contral the procedure)
    if n==1 %%build Olympic rings fail(get the wrong result)
        hold off %%clean the unqualified figure
        break%%restart "for" loop
    end
    if n==2%%get a suitable figure
        m=0;%%stop "while" loop and finish this procedure
    end
end
f=f+1;fprintf("The times fail:%d\n",f)%%show the times has run
end
axis equal%%adjust plot
```

Figure 23: The main.m

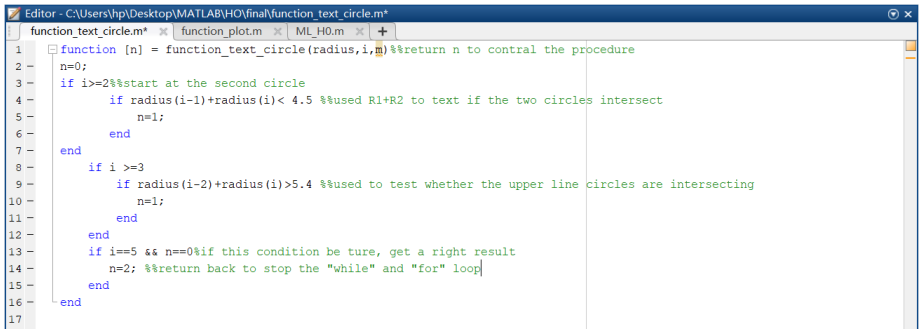


Figure 24: function text circle

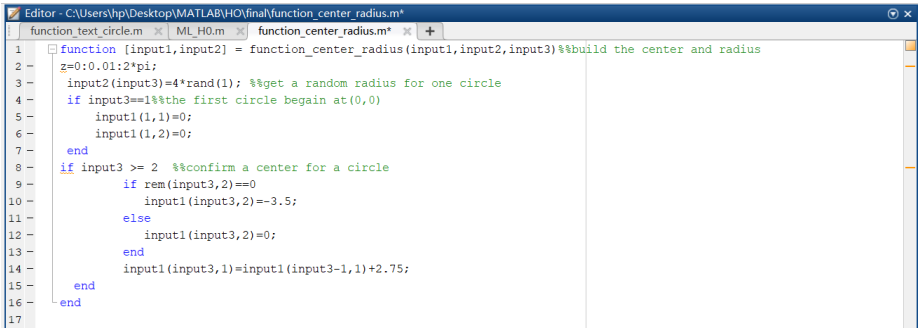


Figure 25: function center radius

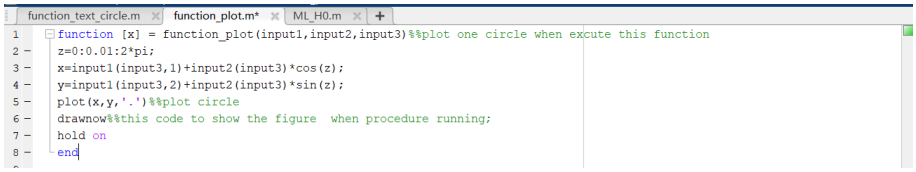


Figure 26: function plot

A.2 Mind map files

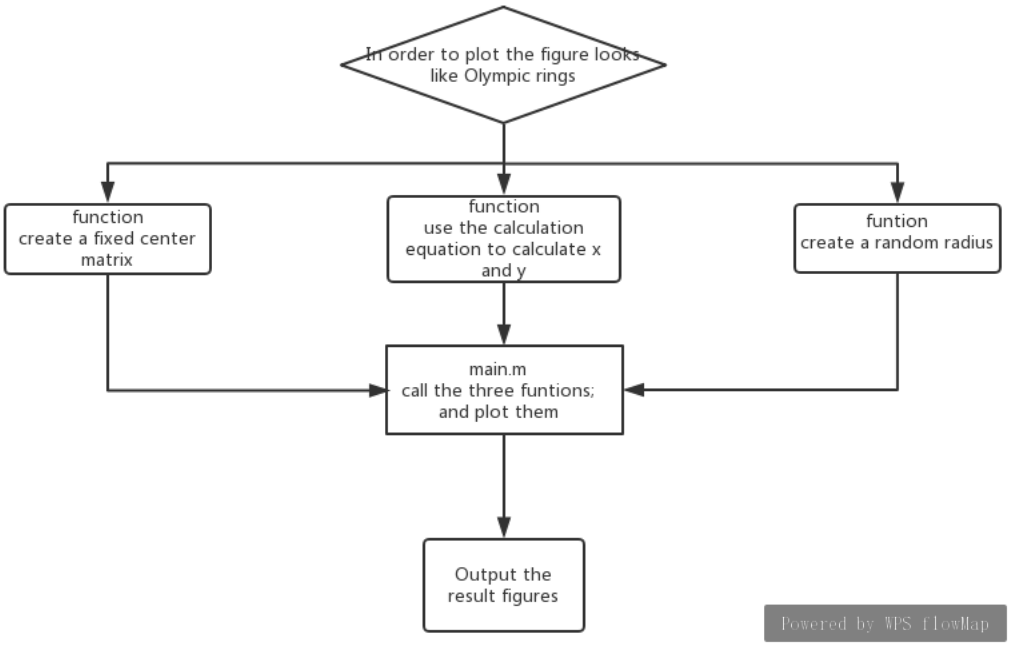


Figure 27: Wang Jingya mind map

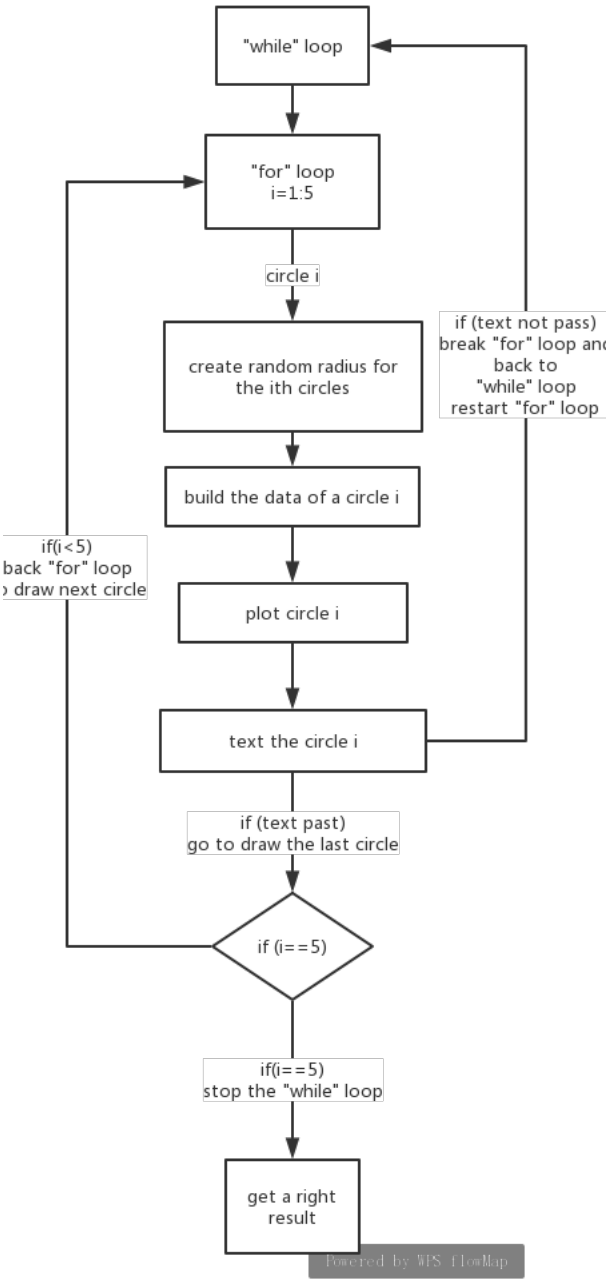


Figure 28: Mi liang Mind map