Purely azimuthal passive localization in attempted formation flight of unmanned aerial vehicles based on CWLS approximate iterative algorithm

Summary

When attempting to fly in formation, UAV clusters should maintain electromagnetic silence as much as possible and emit fewer electromagnetic wave signals to the outside in order to avoid external interference. In order to maintain the formation formation, it is proposed to adjust the position of the UAVs by using the pure orientation passive positioning method, i.e., the position of the UAVs is adjusted by having certain UAVs in the formation transmit signals and the rest of the UAVs passively receive the signals, from which the orientation information is extracted for the positioning.

To address the first question, it is required to model the localization of a UAV that receives signals passively. Firstly, based on the given raw data, the distances d_1, d_2, d_3 between the UAVs and the angles β_1, β_2, β_3 of the triangle formed by the three UAVs can be obtained. From the sine theorem and the angle information, we can calculate γ_1 (see the notation), substituting γ_1 back into the expression of the sine theorem, we can invert it to find out the distance from the middle UAV to the one that receives the signal, Y, and then through the sine theorem or the cosine theorem we can get d_4, d_5,Y, and get the three equations of the trajectory of the drone that receives the signal, and then solve the equation of the three trajectories, and then solve the equation of the three drones. The position information of the UAV receiving the signal can be uniquely determined by solving the equations.

In response to the second question, how many more drones are needed to transmit signals to achieve effective drone localization when the number is unknown. First, we show that two drones are sufficient to localize the drone, but the efficiency and accuracy are not high. Then it is shown that for adding more than one UAV transmitting signals, effective localization of the UAV can be achieved with high accuracy, i.e., three UAVs are chosen to achieve effective localization. Iterate through the number of the UAV that transmits signals as the third UAV and draw the whole position distribution map, there is one and only one of the eight position distribution maps that has a small difference, i.e., the correct position distribution. Therefore, we adopt this strategy: we iterate over the third UAV transmitting signals and locate the remaining UAVs to obtain a whole position distribution graph, and find the one with the smallest error, i.e., the correct localization.

In response to the third question, it is required to give a solution for adjusting the UAV. First the given data is processed, plotted and the distribution of ideal and actual positions is obtained and it is observed that the positions of $FY00$ and $FY01$ are correct. Hence these two drones are selected as the drones that transmit the signal. We sequentially select the possible number of the third UAV transmitting the signal and in this way we achieve the adjustment of the positions of all the UAVs with slight positional deviations.

Since there is a one-to-one correspondence between the angle sequence $[\alpha_1, \alpha_2]$ and the coordinates (x, y) of the corresponding points. We have the following strategy: first we enumerate the currently used UAV as the third UAV that emits the signal, then we enumerate which UAV is currently adjusted, and the region with side length R is the feasible domain with the center of the square at the point where the position of the currently emitting UAV is located. Due to the slight deviation, the effective mediation must be within the feasible domain as long as a larger R is set. Take 10000 equal points in the feasible domain uniformly, and traverse all the divided points to find out the current angle sequence and make a deviation from the ideal angle sequence, and then we choose the angle with the smallest deviation as the adjustment angle of the UAV position that we are currently receiving signals from, and do the same for the remaining UAVs. After enumerating all the UAVs, we have completed one round of UAV position adjustment. The next adjustment is based on the previous adjustment, i.e., using the *CWLS* approximate iteration method. And after several iterations, the overall deviation is monotonically non-increasing. After several iterations, the error will gradually shrink and converge, and the adjustment of the UAV is completed.

In response to question two, it was asked to consider UAV tuning schemes for UAV clusters in actual flight, not just for circular formations. Since UAV clusters are not just circles, many of the simple operations realized in the second question by the nature of circles can no longer be used. Consider the generalization that $FY01$ is the origin of the entire UAV cluster, so its position must be correct. Also transform the coordinate information into coordinates with $FY01$ as the origin. Each time we select two drones and $FY01$ as the drones that transmit signals. We enumerate the number of the drone that needs to be adjusted, and for the current drone, the point where it is located as the center of the square and the region with side length R is the feasible domain. Since it is slightly biased, the effective mediation must be within the feasible domain as long as a larger R is set. Take 10000 equal points uniformly in the feasible domain. Enumerate each point, calculate the deviation of the sequence of angles between that point and the ideal point, take the point with the smallest deviation, and perform an iterative operation on it. Obviously the overall deviation is monotonically non-increasing. After many iterations of computation, the error will gradually shrink and converge, and the adjustment arrangement ends.

Keywords: CWLS approximate iterative algorithm; Omni-directional passive localization; Convex function optimization;

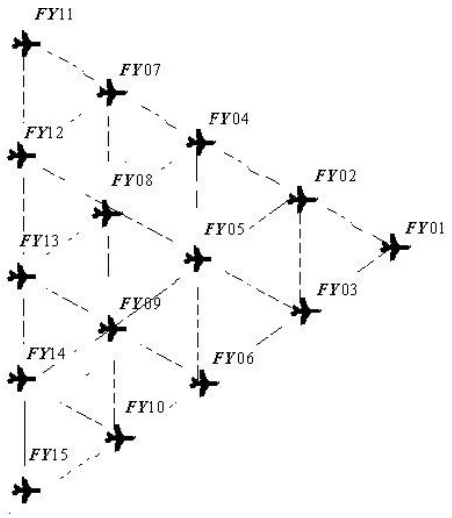
1 Synthesis of issues

1.1 Background to the issue

When attempting formation flights, UAV clusters should maintain electromagnetic silence as much as possible and emit fewer electromagnetic wave signals to the outside in order to avoid external interference. In order to maintain the formation formation, it is proposed to use the pure azimuth passive localization method to adjust the position of UAVs. Each UAV in the formation has a fixed number, and the relative position relationship with other UAVs in the formation remains unchanged. The direction information received by the UAV receiving the signal is agreed to be the angle between the UAV and the line connecting any two UAVs transmitting the signal.

1.2 put forward

1. The formation consists of 10 UAVs in a circular formation, in which 9 UAVs (No. *FY* 01~ *FY* 09) are uniformly distributed on the circumference of a circle, and 1 UAV (No. *FY* 00) is located in the center of the circle. The UAVs were kept at the same altitude based on their perceived altitude information.
   - The UAV at the center of the circle (*FY* 00) and the other two UAVs in the formation transmit signals, while the rest of the UAVs with slight deviations in their positions passively receive signals. When the positions of the UAVs transmitting signals have no deviation and their numbers are known, the localization model of the UAVs passively receiving signals is established.
   - A UAV with a slightly deviated position receives signals from UAVs numbered *FY* 00 and *FY* 01, and signals from a number of UAVs in a formation whose numbers are unknown. If there is no deviation in the positions of the drones transmitting the signals, how many drones than *FY* 00 and *FY* 01 need to transmit the signals in order to achieve effective positioning of the drones?
   - According to the formation requirements, one UAV is located at the center of a circle, and the other nine UAVs are evenly distributed on the circumference of a circle with a radius of *100m*. When the positions of the drones are slightly off at the initial moment, give a reasonable solution for adjusting the positions of the drones. Using the data given in the question, adjust the position of the UAV only according to the direction information it receives, please give a specific adjustment scheme.
2. In actual flight, UAV clusters can also be in other formation formations, such as a conical formation formation, where two neighboring UAVs in a straight line are equally spaced. The purely azimuthal passive localization scenario is still considered to design the UAV position adjustment scheme.



2 Model Assumptions and Notation

2.1 model assumption

1. The speed of signal propagation is not considered.
2. The interference loss of the signal is not considered.
3. The presence or absence of magnetic field interference in the surrounding area is not considered.
4. The presence of obstructions during flight is not taken into account.

2.2 Description of symbols

Table 1: Symbol Description Table

Symbol	Definition
FY 0K	Number of the drone, K= 0, 1, - - - , 9 β_1 Angle between drones transmitting signals β_2 Angle between drones transmitting signals β_3 Angle between drones transmitting signals
γ_1	Angle between the first UAV transmitting the signal and the second UAV transmitting the signal and the UAV receiving the signal γ_2 Angle between the second drone transmitting the signal and the drone receiving the signal and the third drone transmitting the signal a_1 Angle obtained by transmitting the signal 1
a_2	The angle obtained by transmitting the signal 2
d_1	Distance between the first signaling drone and the second signaling drone d_2 Distance between the second signaling drone and the third signaling drone d_3 Distance between the third signaling drone and the first signaling drone d_4 Distance between the first drone transmitting the signal and the drone receiving the signal d_5 Distance between the third drone transmitting the signal and the drone receiving the signal
Y	Distance between the second drone transmitting the signal and the drone receiving the signal
m	Horizontal coordinates of the drone receiving the signal
n	Vertical coordinates of the drone receiving the signal
R	The length of the edge of the feasible domain
arccos	Inverse function of the cos function

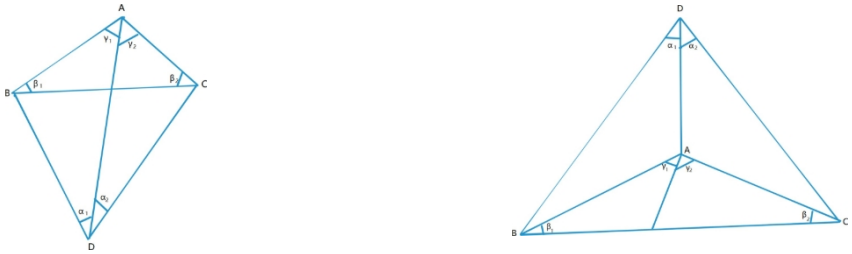
3 Problem analysis

3.1 Analysis of question one

For Problem 1, it is required to model the localization of a passive signal-receiving UAV and give a solution to adjust to the desired position.

first question

For the first question, it is required to model the localization of a UAV that receives signals passively. First of all, since the location and number of the UAVs transmitting signals are known, and there are three UAVs transmitting signals, we can form a triangle with the three UAVs as endpoints, and then according to the distance formula, the sine theorem, and the cosine theorem, we can find the lengths of the sides of the triangle and the angles of the triangle formed by the three UAVs. There are only two cases to determine another drone by three drones.



(a) Case I (b) Case II

Figure 1: Scenario Map

For Case I, we use the sine and cosine equations to solve the joint equation to obtain γ_1, and at the same time, substituting γ_1 back into the equation can obtain Y, which is the distance between the drone that receives the signal and the middle drone. As for the left and right triangles in the illustration diagram of Case 1, we can use the cosine theorem to find out the other sides respectively. As follows.

$$d_4 = \sqrt{d^2 + Y_1^2 - 2d_1 Y \cos\gamma_1} \tag{1}$$

$$d_5 = \sqrt{d^2 + Y_2^2 - 2d_2 Y \cos\gamma_2} \tag{2}$$

At this point we have solved to obtain the distance between the UAV receiving the signal and the UAV transmitting the signal. Since the position of the UAV that transmits the signal is error-free and pre-given, we list the trajectory equation of the UAV that receives the signal by distance as follows. Let the coordinates of the UAV receiving the signal be (m, n).

$$(x_2 - m)^2 + (y_2 - n)^2 = Y^2 \tag{3}$$

$$(x_1 - m)^2 + (y_1 - n)^2 = d(2)^4 (x_3 - m)^2 + (y_3 - n)^2 = d^2 \tag{4}$$

$$5 \tag{5}$$

You can see that all three trajectories are circles, and we can get the possible coordinates of the drone receiving the signal by finding the intersection of the circles of the trajectories. However, since there may be more than one intersection point of the circles, we need to filter the answers we get. In Case 1, the drone in the center and the drone receiving the signal are on opposite sides of the line joining the other two drones, so we filter out the points on the same side. Also the question gives a slight deviation in the position of the drone that receives the signal, so we can set a threshold to filter out points that deviate significantly from the ideal position. For case two, similar to case one, we just need to adjust the corresponding equation for the graph. For cases one and two, we then filter the answers according to the two laws mentioned above. In summary, we get the only point that is the location of the drone that receives the signal.

3.2 second question

For the second question, it is necessary to ask how many drones other than $FY00$ and $FY01$ need to transmit signals in order to realize the effective positioning of the drone. Let's say that exactly three drones are used to transmit signals. According to the description of the problem, we know that two drones with numbers $FY00$ and $FY01$ are used to transmit signals, while the number of the third drone is unknown. Therefore, the model solved in the first question cannot be directly applied in the second question, and we need to adjust it. In the first question, we modeled the localization of the UAV that receives signals passively when its number is known. In the second question, because the number is unknown, and the number of drones is small, we may use the method of violent enumeration to determine the number of the third drone, so that the number is unknown to the number is known, we can continue to use the model established in the first question for the solution, to get the position distribution map. We will select a total of eight points to form eight position distribution maps.

Although the number is unknown, it is a definite one. Therefore when we use other points to localize, the data used such as the angle is wrong and will cause a large error. Therefore one and only one of the eight location maps is correct. We can do an error processing between each position distribution map and the original position distribution map, and select the one with the smallest error as the final position location.

3.3 third question

For the third question, it is required to give an adjustment scheme for the drone. We first plot the conditions given in the question and get the distribution of ideal positions. Then, based on the information given in the question, we draw the actual distribution of positions, and we can observe that the positions of $FY\,00$ and $FY\,01$ are correct. Therefore we can directly select these two drones as the drones that transmit signals. For the third drone, we will select from $FY\,02$~ $FY\,09$ in order to realize the position adjustment of all the drones with slight deviation to the drone.

Based on the previous question, we know that we can uniquely determine a point we know the relevant data of the UAV that transmits the signal and the angle α_1 and α_2 between the UAV that receives the signal and the UAV that transmits the signal. Then when α_1 and α_2 change, the corresponding point also changes, which means that there is a one-to-one relationship between the angle sequence $[\alpha_1, \alpha_2]$ and the coordinates (x, y) of the corresponding point. Specifically, we have the following strategy.

First we enumerate which drone is currently being used as the third drone transmitting signals, and then enumerate which drone is currently adjusted.

For the adjustment, since the actual UAV position is slightly deviated, we can take the current UAV position as the center of the square and the region with side length R as the feasible domain. Since it is a slight deviation, the effective mediation must be within the feasible domain as long as a larger R is set. Take 10000 equal points evenly in the feasible domain, and then traverse all the divided points on the square domain to find out the current angle sequence and make a deviation from the ideal angle sequence, and then we choose one of the points with the smallest deviation as the adjustment position of our current UAV position. Then we select the point with the smallest deviation as the adjustment position of our current UAV and do the same for the remaining UAVs. We can get the approximate optimal adjustment scheme in the case of selecting the current UAV as the third UAV transmitting signals.

Also, since we divided the state space in sufficient detail, this allowed the accuracy of the results to be guaranteed. After enumerating all the UAVs, we completed a round of UAV position adjustment.

Since each time we select a point with the smallest spatial deviation from the current state as the adjustment strategy for the current point, then the overall deviation is monotonically non-increasing. Then we can make the position of the UAV infinitely close to the ideal position after many iterations, so that the error will eventually stabilize.

In summary, after iteration, the error will gradually shrink and converge, and the adjustment of the

UAV is completed.

3.4 Analysis of question two

For question two, it was asked to consider the UAV adjustment scheme for actual flights in which the UAV cluster is not just a circle in formation formation. Since the UAV cluster is not just a circle, many of the simple operations that were realized in the second question with the help of the nature of the circle can no longer be used. Consider the generalization that by setting $FY\,01$ as the origin of the entire UAV cluster, its position must be correct. At the same time, the coordinate information of the remaining UAVs is transformed into coordinates under $FY\,01$ as the origin. Each time we choose two drones and $FY\,01$ as the drones that transmit signals. From Problem 1, we know that there is a one-to-one correspondence between the sequence of angles and coordinates between the UAVs transmitting the signal and the UAVs receiving the signal. Therefore, we can enumerate the UAV numbers that need to be adjusted, and for the current UAV, the point where it is located as the center of the square and the region with side length R is the feasible domain. Since it is a slight deviation, the effective mediation must be within the feasible domain as long as a larger R is set. Take 10000 equal points uniformly in the feasible domain. Enumerate each point and calculate the deviation from the sequence of angles between that point and the ideal point, taking the one with the smallest deviation. Obviously the overall deviation is monotonically non-increasing, if we can find such a point, then there is a valid mediation, otherwise the current UAV is already at the ideal point. After many iterations of computation, the error will gradually shrink and converge, and the adjustment arrangement ends.

4 Modeling, solving and conclusions

4.1 Issue 1

For Problem 1, it is required to model the localization of a UAV that receives signals passively and give a solution to adjust to the desired position.

first question

For the first question, it is required to model the localization of a UAV that receives signals passively. First of all, we know the location and number of the UAV that transmits the signal, and there are three UAVs that transmit the signal, we can use these three UAVs as the endpoints to form a triangle, then according to the distance formula, the sine theorem and the cosine theorem, we can get the side lengths and the angles of the triangle formed by the three UAVs. Specifically as follows.

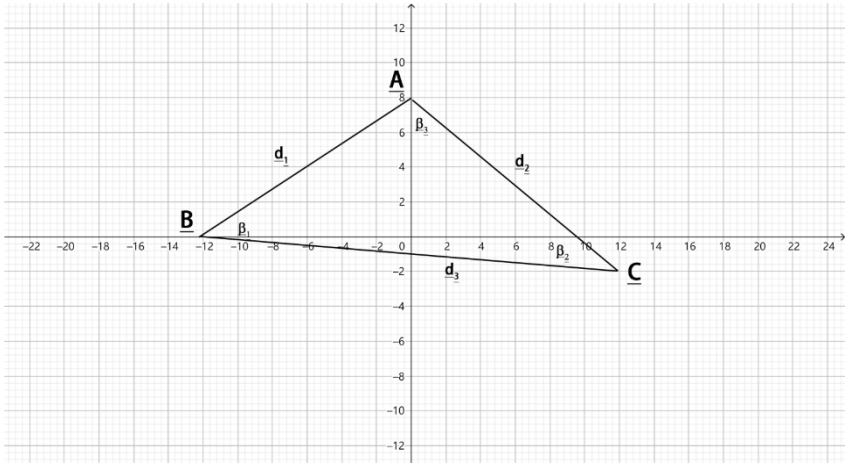


Figure 2: Schematic of the lengths and angles of the triangle formed by the drone

$$d_1 = \sqrt{(x_1 - x_{(2)})^2 + (y_1 - y_{(2)})^2} \tag{6}$$

$$d_2 = \sqrt{(x_2 - x_{(3)})^2 + (y_2 - y_{(3)})^2} \tag{7}$$

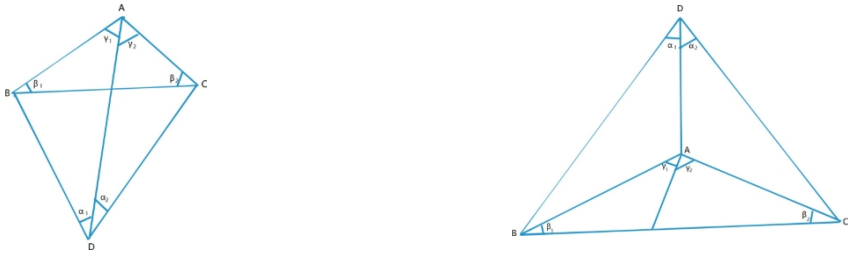
$$d_3 = \sqrt{(x_3 - x_{(1)})^2 + (y_3 - y_{(1)})^2} \tag{8}$$

$$\beta_1 = arccos\left(\frac{d_1^2 + d_3^2 - d_2^2}{2d_1d_3}\right) \tag{9}$$

$$\beta_2 = arccos\left(\frac{d_2^2 + d_3^2 - d_1^2}{2d_2d_3}\right) \tag{10}$$

$$\beta_1 = arccos\left(\frac{d_1^2 + d_2^2 - d_3^2}{2d_1d_2}\right) \tag{11}$$

There are only two ways to determine another drone by three drones. As shown below



(a) Case I (b) Case II

Figure 3: Scenario Map

For case 1, we use the sine and cosine formulas to solve the equation. Firstly, based on the sine theorem and the image, we can get the following three equations:.

$$\frac{d_{(1}}{sina_1} = \frac{Y}{sin(\pi - (a_1 + \gamma_1))} \tag{12}$$

$$\frac{d_{(2}}{sina_2} = \frac{Y}{sin(\pi - (a_2 + \gamma_2))} \tag{13}$$

$$\gamma_1 + \gamma_2 = \beta_3 \tag{14}$$

By associating the above equation, we can solve to get γ_1, and at the same time, by substituting γ_1 back into the equation, we can get Y , which is the distance between the UAV that receives the signal and the UAV in the middle position. As for the left and right triangles in the analyzed figure of Case 1, we can use the cosine theorem to find out the other sides respectively. As follows.

$$d_4 = \sqrt{d^2 + Y_1^2 - 2d_1 Y \cos\gamma_1} \tag{15}$$

$$d_5 = \sqrt{d^2 + Y_2^2 - 2d_2 Y \cos\gamma_2} \tag{16}$$

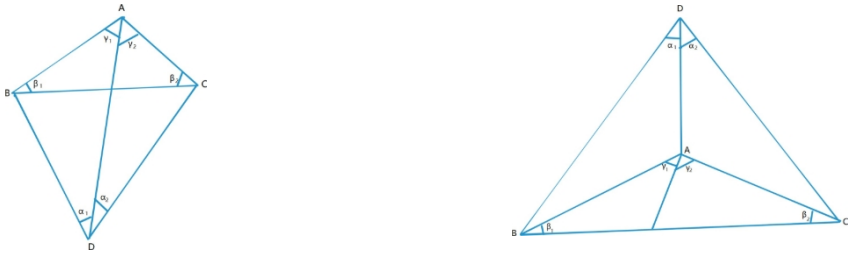
At this point we have solved to get the distance between the UAV receiving the signal and the UAV transmitting the signal. Since the position of the UAV transmitting the signal is error-free and pre-given, we list the trajectory equation of the UAV receiving the signal by distance as follows. Let the coordinates of the UAV receiving the signal be (m, n).

$$(x_2 - m)^2 + (y_2 - n)^2 = Y^2 \tag{17}$$

$$(x_1 - m)^2 + (y_1 - n)^2 = d_{(2)}^2 \tag{18}$$

$$(x_3 - m)^2 + (y_3 - n)^2 = d_5^2 \tag{19}$$

You can see that all three trajectories of the received signal are circles, and we can get the possible coordinates of the drone receiving the signal by finding the intersection of the circles of the trajectories. However, since there may be more than one point of intersection of the circles, we need to filter the resulting answer.

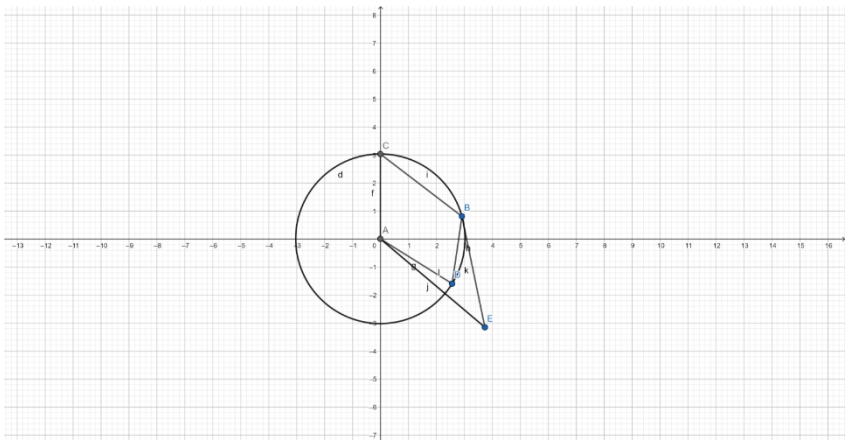


(a) Case I (b) Case II

Figure 4: Scenario Map

Since in scenario one, the drone in the middle and the drone receiving the signal are on opposite sides of the straight line where the line segment connecting the other two drones is located, the points located on the same side are filtered out.

Also the question gives a slight deviation in the position of the UAV that receives the signal, so we can set a threshold to filter out the points that have a large deviation from the ideal position, i.e., beyond the set threshold.



According to the above figure, for the case where two solutions are derived, the ones that will deviate too much can be filtered out directly.

For case 2, we still use the sine and cosine formulas to solve the equation. Firstly, according to the sine theorem and the image, we can get the following three equations:.

$$\frac{d_{(1}}{sina_1} = \frac{Y}{\sin\pi\gamma_1 - a_{(1)}} \tag{20}$$

$$\frac{d_{(2}}{sina_2} = \frac{}{\sin\gamma_2 - a_{(2)}} \tag{21}$$

$$\gamma_1 + \gamma_2 = \beta_3 \tag{22}$$

By associating the above equation, we can solve to get γ_1, and at the same time, by substituting γ_1 back into the equation, we can get Y, i.e., the distance between the UAV that receives the signal and the one that is in the middle position. And in the case two analyzed figure, the left and right triangles, we can use the cosine theorem to find out the other sides respectively. As follows.

$$d_4 = \sqrt{d^2 + Y_1^2 - 2d_1 Y\cos\gamma_1} \tag{23}$$

$$d_5 = \sqrt{d^2 + Y_2^2 - 2d_2 Y\cos\gamma_2} \tag{24}$$

At this point we have solved for the distance between the UAV receiving the signal and the UAV transmitting the signal. Since the position of the UAV transmitting the signal is error free and is given in advance, we can list the trajectory equation of the UAV receiving the signal by distance as follows. Let the coordinates of the UAV receiving the signal be (m, n).

$$(x_2 - m)^2 + (y_2 - n)^2 = Y^2 \tag{25}$$

$$(x_1 - m)^2 + (y_1 - n)^2 = d(2)^2 \tag{26}$$

$$(x_3 - m)^2 + (y_3 - n)^2 = d^2 \tag{27}$$

We can see that all three trajectories of the received signal are circles, and we can get the possible coordinates of the drone receiving the signal by finding the intersection of the circles of the trajectories. Since there is more than one intersection point where the two circles intersect, we need to filter the answers we get. In Case 1, the drone in the center and the drone receiving the signal are on opposite sides of the line joining the other two drones, so filter out the points on the same side. At the same time, the position of the drone receiving the signal is given in the question as slightly deviating, so we can set a threshold to filter out the points that deviate greatly from the ideal position, i.e. exceed the set threshold.

For cases one and two, we then filter the answers based on the two laws mentioned above. To summarize, we get the only point that is the location of the drone that received the signal.

The following proves that such an approach is correct:

First of all, according to the knowledge of secondary school, we can know that the intersection point of any two circles is at most two. According to the derivation of our formula, we get three circles, we first choose two circles to find its resulting intersection point, we can get two results. Then we find the condition in which the third circle intersects the second circle. There are three cases, intersecting at one of the two previous intersections, intersecting at two points of the two previous intersections, and intersecting at zero of the two previous intersections. For the third case, since it is certain to have a solution and to satisfy all three equations, the third case does not exist. For the second case, intersect at two points of the two previous intersections. Then the line segment obtained by joining these two intersections is the chord on the three circles. And the center of the circle i.e. our drone is located on the vertical bisector of that chord. This is shown in the following figure.

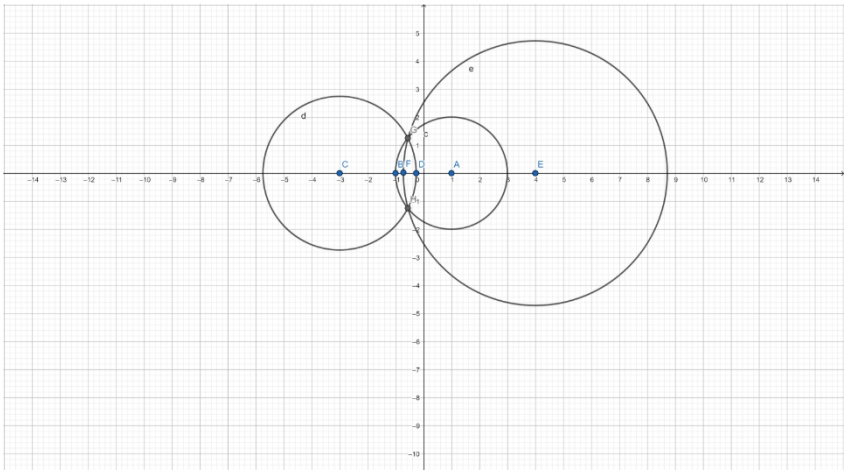


Figure 5: Two intersections

If the three circles intersect at the same two points, then we can conclude that the three drones lie on the same straight line. Whereas in the question there are only nine drones on the circumference of the circle and they are evenly distributed, it is not possible that the three drones lie on the same straight line. The approach then yields a uniquely determined point. Meanwhile, the two main strategies mentioned above are added to the original to avoid situations we may not have considered and further ensure that we get a definite point, as shown below:

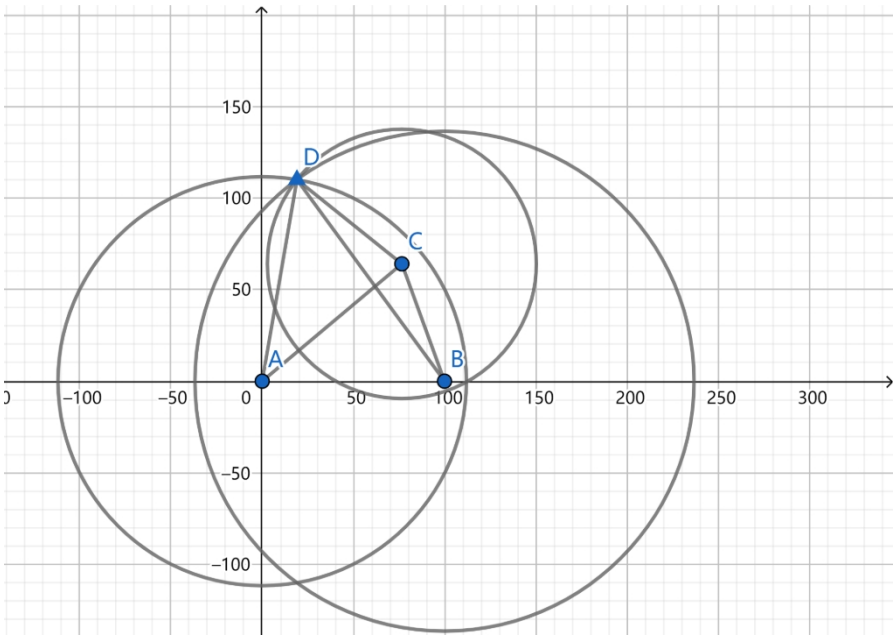
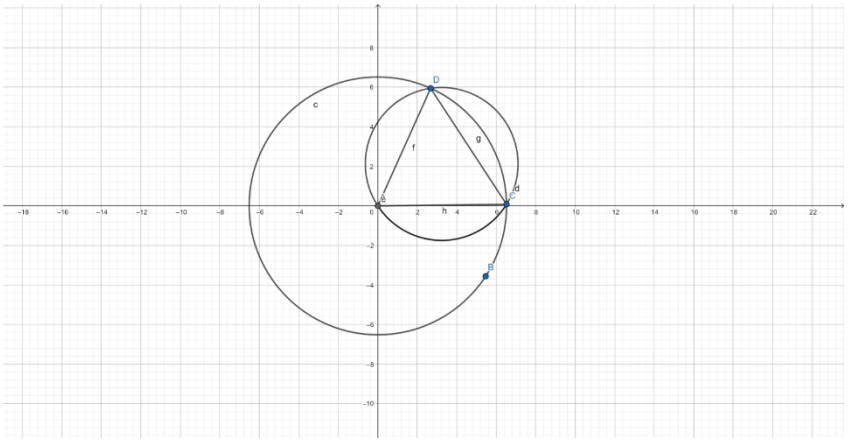


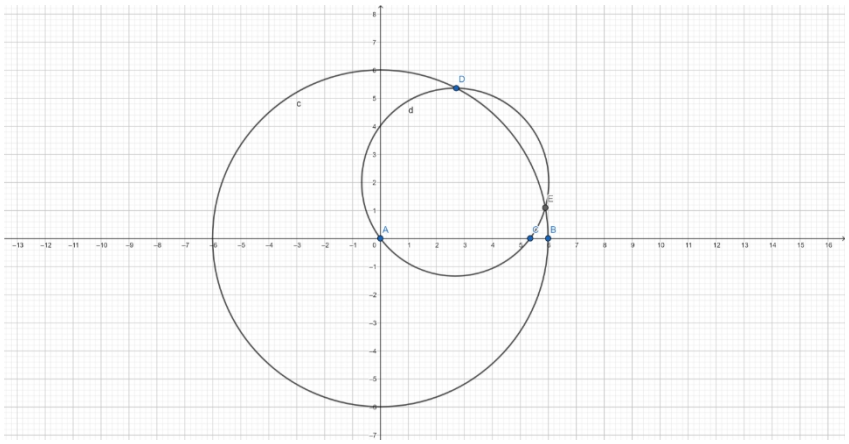
Figure 6: An intersection

second question

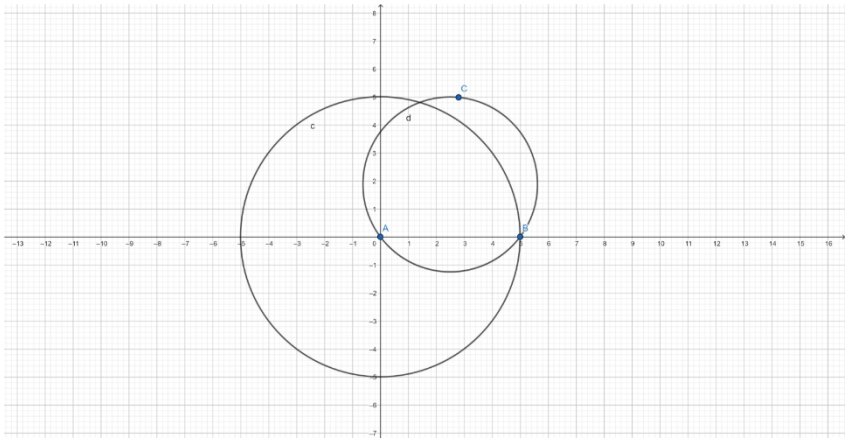
For the second question, it is required that at least a few more drones are needed to transmit signals for effective localization of other drones. First of all for at least how many more drones are needed, we can discuss it in two cases, adding drones and not adding drones. If we do not add drones, we currently use two drones for localization, and the only information we can obtain for computation is the initial data and distance of the two drones, and the angle information of the drone that receives the signal and the two drones that transmit the signal. Based on the above two information and the fixed length and angle theorem, we can only determine that the drone currently receiving the signal is on a circular trajectory as follows.



Although we can solve the equations of the trajectory circle and the original circle jointly to get up to two intersections based on the condition of slight deviation. But on the one hand, when the trajectory circle and the original circle are solved for an answer with two intersections, we do not have the extra information to rule out the wrong answer, as follows.



On the other hand, since it is a slight deviation, it does not mean that the position of the accepting drone is on the circle, and there is a possibility that it is not on the circle, as shown below.



In summary the method of selecting two drones, although results can be obtained by chance, has great unreliability and inaccuracy.

If you add at least one drone. Obviously we know that the amount of information we acquire is monotonically undiminished as we have more launch drones, and therefore the conditions available to us are monotonically undiminished. That is to say that the accuracy of the results does not get worse as the number of drones increases, but since it is important to maintain as much electromagnetic silence as possible, we can prove it by increasing it one by one. Obviously, on the basis of the first question, it is known that only three drones transmitting signals are needed to determine the position of the drone receiving the signal. The next proof requires only three drones transmitting signals to determine the location of the drone that receives the signal with a high degree of accuracy.

Let exactly three drones be used to transmit signals. According to the description of the problem, we only know the signals emitted by two drones numbered $FY00$ and $FY01$. The number of the third drone is unknown. Therefore, the model solved in the first question is not directly applicable in the second question, and we need to adapt it. In the first question, we modeled the localization of a UAV that receives signals passively when its number is known. In the second question, since the number is unknown and the total number of UAVs receiving signals is small, we may wish to use the method of violent enumeration to obtain the number of the third UAV, which converts the unknown number into a known number, and we can then continue to use the model established in the first question to solve and obtain a position distribution map. Under the assumption that the data is correct, it is obvious that the results we calculate through the model must be correct. In total, we will pick eight points to form eight location distribution maps. Although the numbering is unknown, it is a definite one. Therefore when we use other points to locate, the data used such as the angle is wrong and will cause a large error.

The following demonstrates that when we select for a definite UAV as the third UAV to receive the signal, there is a large deviation when using the data from other UAVs to localize. There are three cases, the angle has a large deviation or the side length has a large deviation, the angle and side length do not have a large deviation but the coordinates have a large deviation, the angle and side length coordinates have a large deviation, and the angle and side length coordinates have a large deviation.

Both have large deviations.

First of all, for the third case is obviously not valid, when the coordinates do not have a large deviation, in fact, it shows that the two points are a point, and the question is contradictory. Consider the case of angle or side length with a large deviation, whether the angle or side length with a large deviation, will make our final value of γ_1 is different, and γ_1 of the value of the change will lead to the distance between the UAV transmitting signals and the UAV receiving signals that we seek to change. Observing the calculation formula, we find that the core of the trajectory circle lies in the distance, and if the distance is not correct, then the trajectory equations we find will also have a large deviation. Since we have two points that are fixed in this question, we might as well take the equation of the trajectory formed by these two points, and it is impossible to maintain the previous intersection with a large change in side lengths in the case of fixed points. This is because, if we could maintain the previous intersection, the equation of the circle would be unchanged according to the theorem that a circle can be determined by three points in the plane, which is clearly contradictory to the previous one.

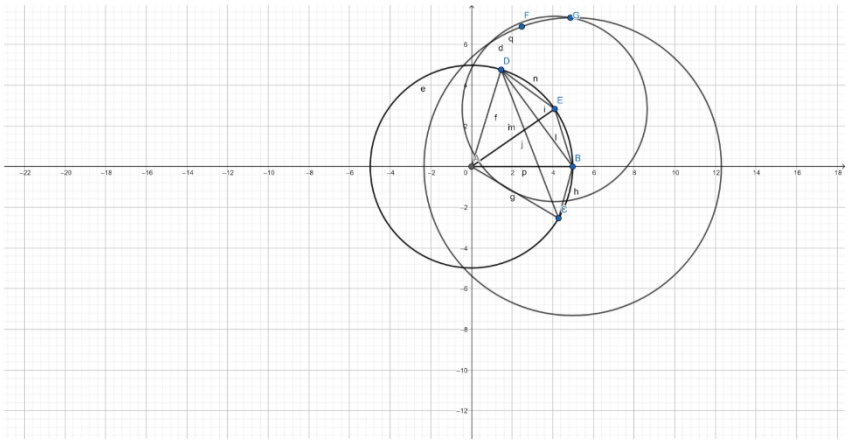


Fig. 7: Large deviations in edge length or angle

Consider the case where neither the angle nor the length of the side deviates greatly but there is a large deviation in the coordinates. If neither the angle nor the length of the side deviates greatly, then this means that the first distance information we get is the same. But the remaining two distances are changed significantly. Also due to the large difference in coordinates then the same trajectory equation has a large difference. Similarly, if we can get the same intersection point, then it is still the same circle as the previous one, contradictory. Therefore, when we select for a definite UAV as the third UAV to receive the signal, the use of data from other UAVs to localize will produce a large deviation.

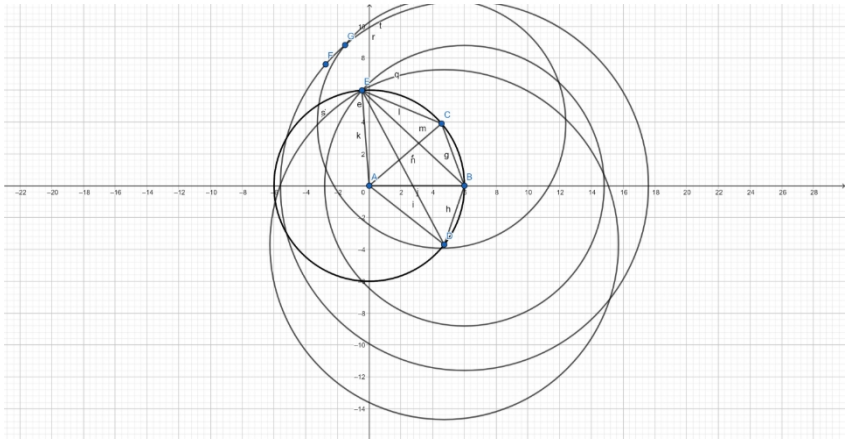


Fig. 8: Coordinates with large deviations

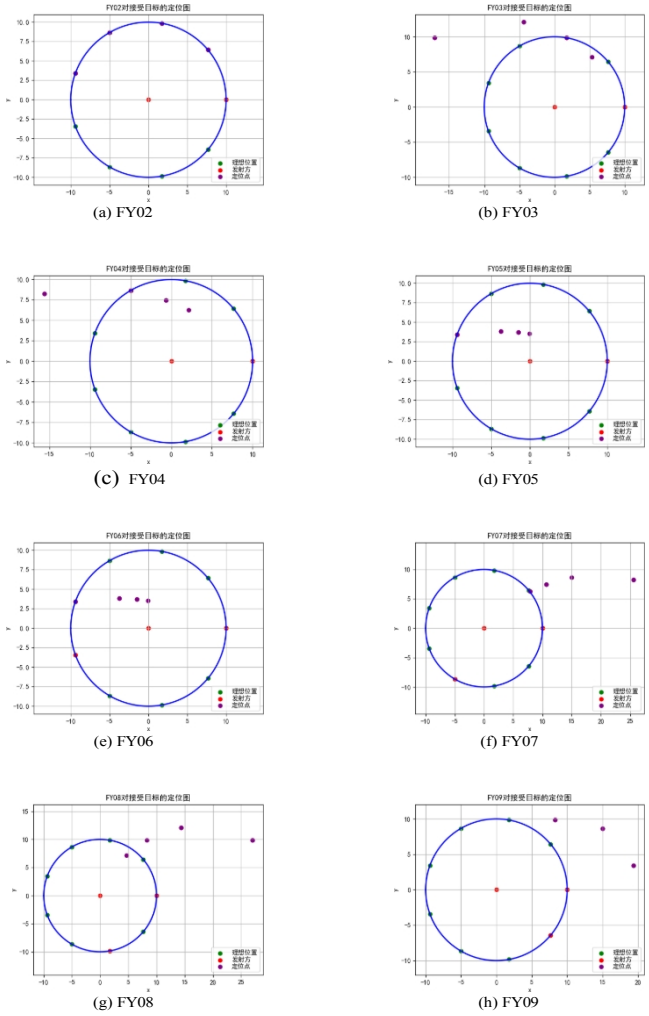
Therefore one and only one of the eight positional distribution charts is correct.

Therefore the localization in the distribution with the smallest error is the correct localization of the UAV receiving the signal.

In summary, the addition of at least one drone allows for effective drone localization, and the strategy is to enumerate which drone

as the third drone to receive the signal and generate eight position distribution maps, taking the one with the smallest error.

The following will give an example of the process and show its correctness. We may want to take the ideal position as the initial data, then take *FY* 02 as the third UAV that sends a signal and get the angle measurement information for each point. Then enumerate each UAV as the third UAV transmitting the signal, and use the initial data and the angle measurement information to locate the position of the other UAVs and compare it with the initial data to get the following eight position distribution maps as shown below.



(a) FY02 (b) FY03

(c) FY04 (d) FY05

(e) FY06 (f) FY07

(g) FY08 (h) FY09

Figure 9: Location Map

Observing the above figure, we can find that the angle information obtained by *FY* 02 as the third drone transmitting the signal is more accurate only *FY* 02, while the results obtained by the other drones have a large deviation due to the use of incorrect angle information, so even if we don't know the number of the current drone. We can enumerate the numbers of the drones and select the point position with the smallest error as our answer.

third question

For the third question, it is required to give an adjustment scheme for the UAV. We first plot the given data to get the distribution of ideal positions. Then the actual distribution of positions is plotted based on the information in the table given in the question. The diagram is as follows

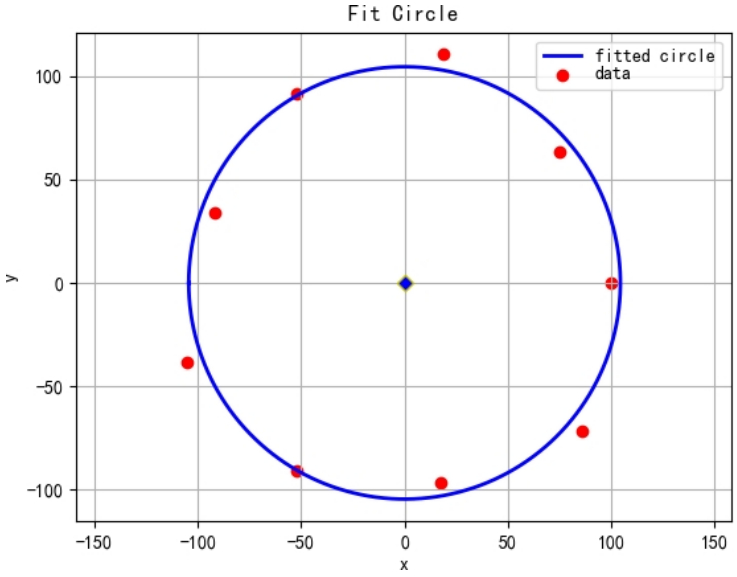


Figure 10: Physical Location Map

It can be observed that the positions of *FY* 00 and *FY* 01 are correct. Therefore we can directly select these two drones as the drones transmitting the signal.

For the third drone, we will select them sequentially. Since the UAV as the one transmitting the signal is not adjusted, it is necessary to select different UAVs sequentially to adjust all the UAVs whose positions have a slight deviation. According to the previous question, we can know that we can uniquely determine a point the relevant data of the drone that transmits the signal and the angles α_1 and α_2 between the drone that receives the signal and the drone that transmits the signal are known. Then, when α_1 and α_2 change, the corresponding point also changes, which indicates that there is a one-to-one relationship between the angle sequence $[\alpha_1, \alpha_2]$ and the coordinates (x, y) of the corresponding point. Therefore, we can judge whether the current adjustment is effective or not by comparing the angle sequence after each movement change with the angle sequence at the ideal position. Specifically, we have the following strategy.

First enumerate which drone we are currently using as the third drone transmitting signals, and then enumerate which point is currently adjusted.

For the adjustment, since it is a slight deviation, we can take the current point as the center of the square centered on the point where it is located, and the area with side length R as the feasible domain. Since it is a slight deviation, the effective mediation must be within the feasible domain as long as a larger R is set. The feasible domain will be uniformly taken 10000 equal points, and then traverse all the points, the current angle sequence and the ideal angle sequence to do a deviation, and then we select the smallest deviation as our current point of this adjustment.

Do the same operation for all other drones. We can obtain a near-optimal tuning scheme in the case where the current UAV is selected as the third UAV transmitting signals. At the same time, since we divide the state space in sufficient detail, this makes the accuracy of the result guaranteed. After enumerating all the drones, we complete a round of adjustment.

Since each time we select a point with the smallest deviation in the current state space as the adjustment strategy for the current point, then the overall deviation is monotonically non-increasing. Then we can obtain the final result of convergence of the error after many iterations. In summary, after many rounds of adjustment, the error will gradually shrink and converge, and the adjustment of the UAV is completed. The specific adjustment process is summarized in the following table.

Table 2: First Adjustment

Drones transmitting signals	*fy* 01, *fy* 02, *fy* 03
Drones receiving signals	*fy* 04, *fy* 05, *fy* 06, *fy* 07, *fy* 08, *fy* 09
Tolerance before adjustment	5.25
Adjusted error	5.13

Table 3: Second adjustment

Drones transmitting signals	*fy* 01, *fy* 02, *fy* 04
Drones receiving signals	*fy* 03, *fy* 05, *fy* 06, *fy* 07, *fy* 08, *fy* 09
Error before adjustment	5.13
Adjusted error	1.32

Table 4: Third Adjustment

Drones transmitting signals	*fy* 01, *fy* 02, *fy* 05
Drones receiving signals	*fy* 04, *fy* 03, *fy* 06, *fy* 07, *fy* 08, *fy* 09
Tolerance before adjustment	1.32
Adjusted error	1.31

Table 5: Fourth Adjustment

Drones transmitting signals	*fy* 01, *fy* 02, *fy* 06
Drones receiving signals	*fy* 04, *fy* 05, *fy* 03, *fy* 07, *fy* 08, *fy* 09
Tolerance before adjustment	1.31
Adjusted Tolerance	1.30

Table 6: Fifth adjustment

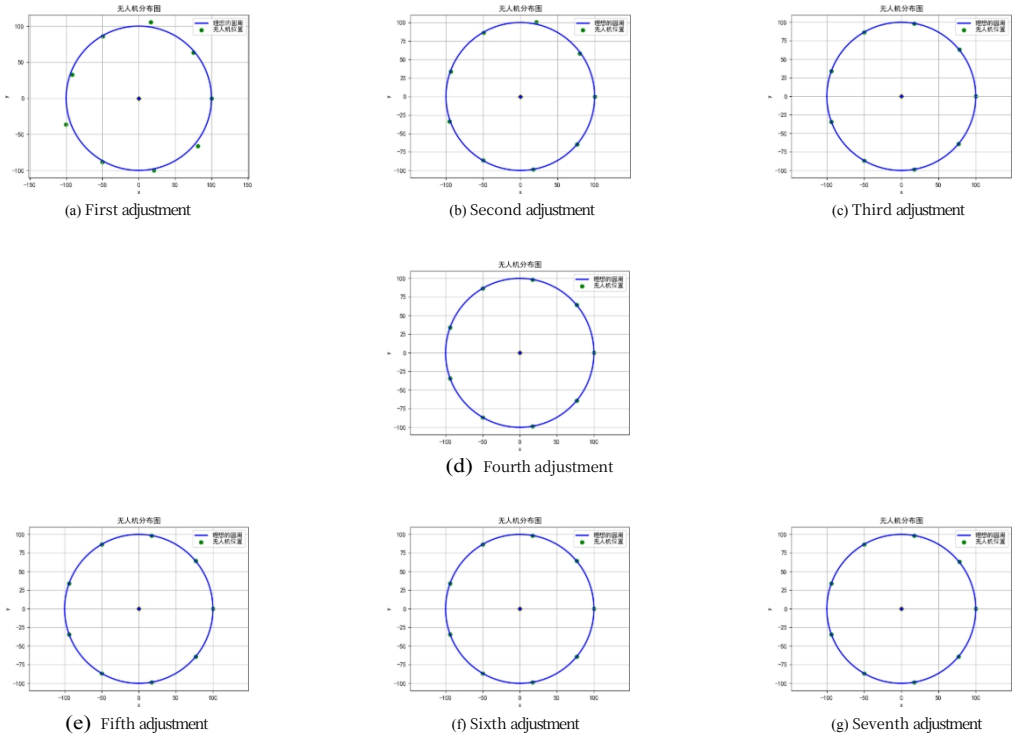
Drones transmitting signals	*fy* 01, *fy* 02, *fy* 07
Drones receiving signals	*fy* 04, *fy* 05, *fy* 06, *fy* 03, *fy* 08, *fy* 09
Tolerance before adjustment	1.30
Adjusted error	1.29

Table 7: Sixth Adjustment

Drones transmitting signals	*fy* 01, *fy* 02, *fy* 08
Drones receiving signals	*fy* 04, *fy* 05, *fy* 06, *fy* 07, *fy* 03, *fy* 09
Tolerance before adjustment	1.29
Adjusted error	1.29

Table 8: Seventh Adjustment

Drones transmitting signals	*fy* 01, *fy* 02, *fy* 09
Drones receiving signals	*fy* 04, *fy* 05, *fy* 06, *fy* 07, *fy* 08, *fy* 03
Tolerance before adjustment	1.29
Adjusted error	1.29



(a) First adjustment (b) Second adjustment (c) Third adjustment

(d) Fourth adjustment

(e) Fifth adjustment (f) Sixth adjustment (g) Seventh adjustment

Figure 11: Iterative Adjustment of Drone Positions

Error variation graph.

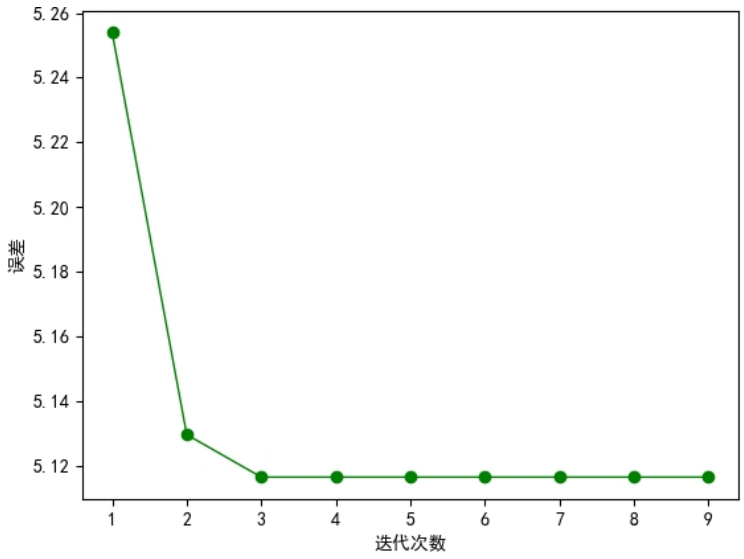


Fig. 12: Error variation

As can be seen from the above figure, as the number of iterations increases, the error becomes smaller and smaller, and eventually converges, and all UAV adjustments are complete.

4.2 Issue 2

For question two, the requirement is to consider UAV adjustment schemes for formation formations in which UAV clusters are not just circles in actual flight. Since UAV clusters are not just circles, many of the easy maneuvers achieved in the second question with the help of the nature of circles can no longer be used. Consider the generalization that by setting $FY01$ as the origin of the entire UAV cluster, its position must be correct. Transform the coordinate information into coordinates with $FY01$ as the origin. Each time we choose two drones and $FY01$ as the drones transmitting the signal. From Problem 1, there is a one-to-one correspondence between the sequence of angles and coordinates between the UAVs that transmit signals and the UAVs that receive signals. Therefore, we can enumerate the UAV numbers that need to be adjusted, and for the current UAV, the square domain with the point where it is located as the center of the square and the side length R is the feasible domain. Since it is a slight deviation, the effective mediation must be within the feasible domain as long as a larger R is set. Divide the feasible domain into 10000 equal points. Enumerate each point and calculate the deviation of the sequence of angles between that point and the ideal point, taking the one with the smallest deviation. In Problem 1 we proved that the overall deviation is monotonically non-increasing, if we can find such a point, then there is a valid mediation, otherwise the current UAV is already at the ideal point and does not need to be adjusted. After many iterations of computation, the error will gradually shrink and converge, and the adjustment arrangement will end. Taking the conical formation formation of the second kind of problem as an example, first we select $FY01$ and take it as the coordinate origin. Since the coordinates of other UAVs are known, we can transform them into new coordinates under the coordinate system with $FY01$ as the coordinate origin, a line parallel to the original x-axis and intersecting $FY01$ as the new x-axis, and a line parallel to the original y-axis and intersecting $FY01$ as the new y-axis. $FY02$ and $FY03$ are selected as the drones used to transmit signals in the first iteration. Next, we iterate through the other drones, and for each drone, we select a circular domain with a radius of $50m$ as the feasible domain (since we require that the spacing between two neighboring airplanes on a straight line is equal, e.g., $50m$, we might as well set the maximum error to be no more than $50m$) and divide the square domain into 10,000 equidistant points. Use all the points as the set of alternative answers, i.e. the state space. Next, traverse the state space, comparing each possible point to the ideal point. If currently we can find a minimum point that makes our deviation smaller, then there is a valid adjustment in the feasible domain, otherwise the current UAV is already at the ideal point and no adjustment is needed. After a number of rounds of iterative computation, eventually the overall degree of deviation will gradually shrink and converge, and the adjustment arrangement ends, the whole process is shown below.

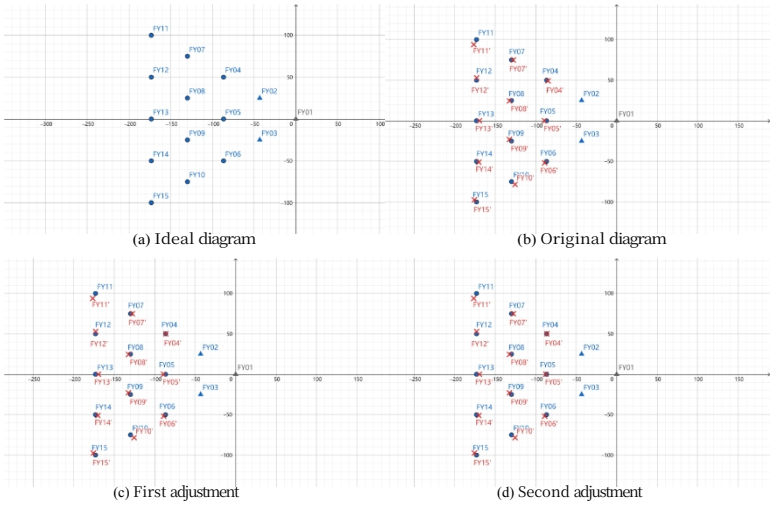


Figure 13: Iterative Adjustment Diagram for Cone Array UAV 1

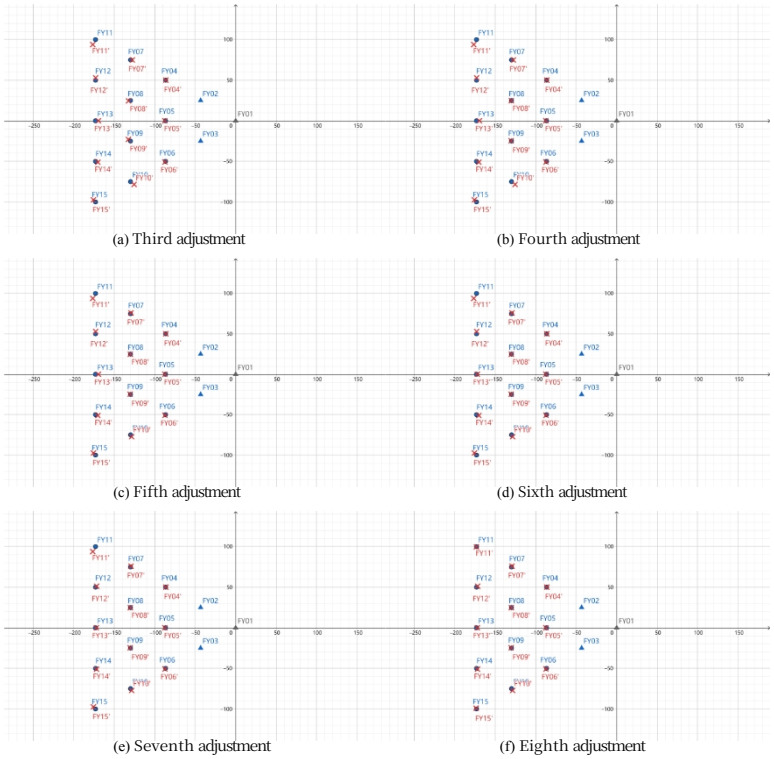


Figure 14: Iterative Adjustment Chart for Cone Array UAVs

Error variation graph.

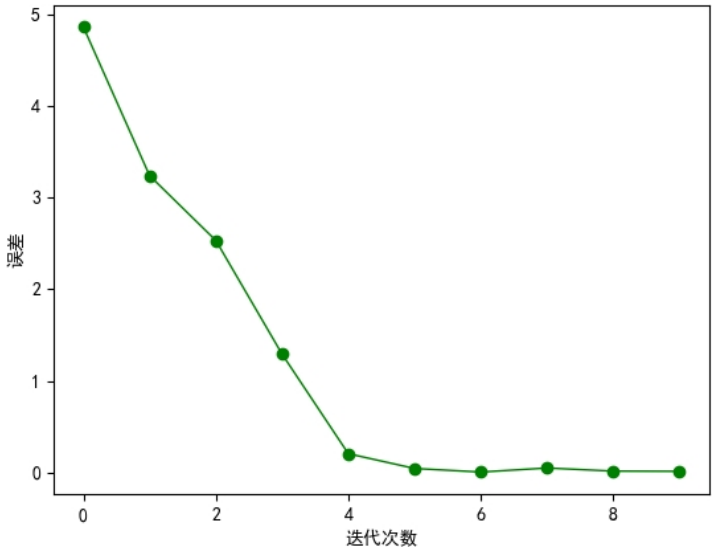


Fig. 15: Error variation

As can be seen from the above figure, as the number of iterations increases, the error becomes smaller and smaller, and eventually converges, and all UAV adjustments are complete.

5 Model Evaluation and Summary

5.1 Advantages of the model

Specifically, our model has the following advantages:

1. Generalizability: The model has very good generalizability and is not limited to purely azimuthal passive localization for attempted formation flights of UAVs, but is applicable to many purely azimuthal passive localization problems for formation flights.
2. Practicality: Pure azimuthal passive localization for attempted formation flight of UAVs is a very realistic formation flight problem, which is of greater significance in our military field. The model can effectively solve the UAV formation problem.
3. Accuracy: In this paper, mathematical derivation is used in the modeling process, traversing the state space, which makes the results highly accurate.

5.2 Disadvantages of the model

Our model has the following drawbacks:

1. The model used in this paper does not take into account many realistic factors, such as the bias interference of the signal, the propagation speed of the signal, etc., which leads to the idealization of our model.
2. The traversal algorithm in this paper is rather violent and the time complexity could be improved.

bibliography

[1] Zhihui Zhao, Jing Wang, Yangquan Chen, and Shuang Ju. Iterative learning- based formation control for multiple quadrotor unmanned aerial vehicles. International Journal of Advanced Robotic Systems, 17(2):1729881420911520, 2020. 4.1.2

[2] Wan, F. and Ding, J.. Research on single station passive localization technique for moving targets. Radar Science and Technology, 9(1):8-12, 2011. 4.1.2

[3] Si Shoukui, Sun Zhaoliang. Mathematical Modeling Algorithms and Applications (Second Edition). National Defense Industry Press, 2015. 4.2.2

[4] Qu Xiaomei, Liu Tao, and Tan Wenrong. Multi-target passive localization algorithm based on multi-UAV collaboration. Sci. China: Information Science, 49(5):570-584, 2019. (document)

[5] Li, H.. Statistical Learning Methods. Tsinghua University Press, Beijing, 2019. 4.1

[6] Wang RJ. UAV cluster system based on uwb localization. Master' s thesis, Hangzhou University of Electronic Science and Technology, China.
2019. (document)

[7] Zou Xianxiong. Research on Passive Localization Method for UAV Targets. Master' s thesis, University of Electronic Science and Technology, 2018.(document) 11

6 appendice

Solution Code.

```
from math import pi
from random import random
from tabnanny import check
from tkinter.ttk import LabeledScale
from scipy import optimize
from ast import Delete
from cProfile import label
from cmath import isnan, nan
from functools import cmp_to_key
from re import X
from tkinter import E
from turtle import color, title
import pandas as pd
from sklearn.preprocessing import PolynomialFeatures
import numpy as np
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
import matplotlib.pyplot as plt
import matplotlib as mpl
from sklearn import metrics
import math
from scipy import optimize as op
from cmath import exp
import random
from scipy.optimize import curve_fit
import cmath
import sympy

mpl.rcParams['font.family']= 'SimHei'
plt .rcParams['axes.unicode_minus'] = False

def P_C(x, y).
    y= y* cmath.pi / 180
    cn1= cmath.rect(x, y)
    return cn1.real , cn1.imag

def init_P().
     p_x, p_y= [], []
     plane_x= [100, 98, 112, 105, 98, 112, 105, 98, ]
     plane_y= [0, 40.10, 80.21, 119.75, 159.86, 199.96, 240.07, 280.17, 320.28]
    for i in range(len(plane_x)):: for i in range(len(plane_x)).
        c = P_C(plane_x[i], plane_y[i])
        p_x.append(c[0])
        p_y.append(c[1])
    return p_x, p_y
# print(p_x)
# print(p_y)

def r(x, y).
    return np.sqrt(x**2 + y**2)

def f(x, y).
    Ri= r(x, y)
    return np.square(Ri- Ri.mean())

def least_squares_circle(coords).
```

```
        x, y= None, None
        if isinstance(coords, np.ndarray).
            x= coords[:, 0]
            y= coords[:, 1]
        elif isinstance(coords, list ).
            x= np.array([point[0] for point in coords])
            y= np.array([point[1] for point in coords])
        else :
            raise Exception(
                "Parameter 'coords' is an unsupported type: "+ str(type(coords))) 71
        x_m= np.mean(x)
        y_m= np.mean(y)
        center_estimate = x_m, y_m
        center, _= optimize.leastsq(f, x, y)
        Ri= r(x, y)
        R = Ri.mean()
        residu= np.sum((Ri− R)**2)
        return R, residu 80

    def plot_data_circle(x, y, R).
        f= plt. figure( facecolor='white')
        plt . axis('equal') 85
        theta_fit= np.linspace(-pi, pi, 180)

        x_fit= R*np.cos(theta_fit)
        y_fit= R*np.sin(theta_fit)
        plt .plot(x_fit, y_fit,  'b-', label="ideal circumference", lw=2)
        plt . plot ([0], [0], 'bD', mec='y', mew=1)
        plt . xlabel('x')
        plt . ylabel('y')
        # plot data
        plt .scatter(x, y, c='green', label='drone position') 96
        plt .legend(loc='best', labelspacing=0.1)
        plt . grid()
        plt .title ('Drone Distribution
Map') 100
    def L_c(x, y).
        plt .plot(x, y, 'ro-', color='green', linewidth=1, label='error case ')
        plt . xlabel('Number of iterations')
        plt . ylabel('Error')
        plt .show()


    # coords= np.array([[p_x[i], p_y[i]] for i in range(len(p_x))])
    # r, residual= least_squares_circle(coords)
    # print("least_squares: \n")
    #             "r: {r}\n"
    #             "residual: {residual}\n".format(xc= 0, yc= 0, r=r, residual=residual))
    # plot_data_circle(coords [:, 0], coords [:, 1], r)
    # plt.show()

def init_c().
        ls , li= [], []
        a, l= 0, 10
        for i in range(9):
            x, y= P_C(l, a)
            ls .append(x)
            li .append(y)
            a+= 40
        return ls , li
```

```
125
126     # c_x, c_y= init_c()
127     c_x, c_y= init_c()
128     p_x, p_y = init_P() 129
130 def check_Af(x1, y1, x2, y2).
131             if (y1* y2< 0).
132                     if (x1+ x2< 0).
133                             return -1
134                     else :
135                             return 1
136             else :
137                     if (x1> x2).
138                             return 0
139                     else :
140                             return 2
141
142     def to_angle(x1, y1, x2, y2).
143             if check_Af(x1, y1, x2, y2) == 0.
144                     d1, d2= sympy.sqrt(c_x[0]** 2+ c_y[0]** 2), sympy.sqrt((x1- c_x[0])** 2+ (y1- c_y[0])** 2)
145                     l1= sympy.sqrt(c_x[0]** 2+ c_y[0]** 2)
146                     l2= sympy.sqrt((x1- c_x[0])** 2+ (y1- c_y[0])** 2)
147                     l3= sympy.sqrt(x1** 2+ y1** 2)
148                     b= sympy.acos((l1** 2+ l2** 2- l3** 2) / (2* l1* l2))
149             elif check_Af(x1, y1, x2, y2)== -1.
150                     d1, d2= sympy.sqrt(x1** 2+ y1** 2), sympy.sqrt(c_x[0]** 2+ c_y[0]** 2)
151                     l1= sympy.sqrt(x1** 2+ y1** 2)
152                     l2= sympy.sqrt(c_x[0]** 2+ c_y[0]** 2)
153                     l3= sympy.sqrt((x1- c_x[0])** 2+ (y1- c_y[0])** 2)
154                     b= sympy.acos((l1** 2+ l2** 2- l3** 2) / (2* l1* l2))
155             else :
156                     d1, d2= sympy.sqrt(x1** 2+ y1** 2), sympy.sqrt((x1- c_x[0])** 2+ (y1- c_y[0])** 2)
157                     l1= sympy.sqrt(x1** 2+ y1** 2)
158                     l2= sympy.sqrt((x1- c_x[0])** 2+ (y1- c_y[0])** 2)
159                     l3= sympy.sqrt(c_x[0]** 2+ c_y[0]** 2)
160                     b= sympy.acos((l1** 2+ l2** 2- l3** 2) / (2* l1* l2))
161             return d1, d2, b
162     def s_f(x1, y1, lex, a1, a2).
163             # print(b)
164             x2, y2= c_x[lex], c_y[lex]
165             # print(check_Af(x1, y1, x2, y2))
166             d1, d2, b= to_angle(x1, y1, x2, y2)
167             # print(d1, d2, b)
168             def num().
169                     x= sympy.symbols('x')
170                     y = sympy.symbols('y')
171                     if (check_Af(x1, y1, x2, y2) >= 0).
172                             f1= d1 / sympy.sin(a1)* sympy.sin(a1+ x)- d2 / sympy.sin(a2)* sympy.sin(a2+ b- x)
173                             f2= y- d1 / sympy.sin(a1)* sympy.sin(sympy.pi- a1- x)
174                     else :
175                             f1= d1 / sympy.sin(a1)* sympy.sin(x- a1)- d2 / sympy.sin(a2)* sympy.sin(b- x- a2)
176                             f2= y- d1 / sympy.sin(a1)* sympy.sin(x- a1)
177                     s= sympy.nsolve((f1, f2), (x, y), (0, 0))
178                     # print(s)
179                     s = list (s)
180                     return s
181             t=  num()
182             # print(t)
183             x, y= sympy.symbols('x, y')
184             if check_Af(x1, y1, x2, y2) == 0.
185                     d3= d1 / sympy.sin(a1)* sympy.sin(t[0])
186                     d4= t[1]
187                     f3= (x- c_x[0])** 2+ (y- c_y[0])** 2- d4** 2
188                     f4= x** 2+ y** 2- d3** 2
```

```
                    u= sympy.nsolve((f3, f4), (x, y), (1.0, 1.0) , verify=False)
        elif check_Af(x1, y1, x2, y2)== 1.
            d3= t[1]
            d4= d1 / sympy.sin(a1)* sympy.sin(t[0])
            # print(d3, d4)
            f3= (x- x1)** 2+ (y- y1)** 2- d3** 2
            f4= x** 2+ y** 2- d4** 2
            u= sympy.nsolve((f3, f4), (x, y), (-1.0, -1.0), verify=False)
        else :
            d3= t[1]
                    d4= d1 / sympy.sin(a1)* sympy.sin(sympy.pi- t[0])
            f3= x** 2+ y** 2- d3** 2
            f4= (x- x1)** 2+ (y- y1)** 2- d4** 2
                    u= sympy.nsolve((f3, f4), (x, y), (1.0, 1.0) , verify=False)
        # print(u)
        return list (u)


def getAngle(v, t):  #v is the point that signaled, t is the point that received the signal
        x1, y1= c_x[v], c_y[v]
        x2, y2= c_x[t], c_y[t]
        if (check_Af(x1, y1, x2, y2)== 0).
            l1= math.sqrt(x2** 2+ y2** 2)
                    l2= math.sqrt((x2- c_x[0])** 2+ (y2- c_y[0])** 2)
            l3= math.sqrt(c_x[0]** 2+ c_y[0]** 2)
                    l4= math.sqrt((x1- c_x[0])** 2+ (y1- c_y[0])** 2)
            l5= math.sqrt((x2- x1)** 2+ (y2- y1)** 2)
                                a= math.acos(float(l1** 2+ l2** 2- l3** 2) / (2* l1* l2))
                            b = math.acos(float(l2 ** 2 + l5 ** 2 - l4 ** 2) / (2* l2* l5))
        elif (check_Af(x1, y1, x2, y2)== 1).
            l1= math.sqrt(x2** 2+ y2** 2)
                    l2= math.sqrt((x2- c_x[0])** 2+ (y2- c_y[0])** 2)
            l3= math.sqrt(x1** 2+ y1** 2)
                    l4= math.sqrt((x1- c_x[0])** 2+ (y1- c_y[0])** 2)
            l5= math.sqrt((x2- x1)** 2+ (y2- y1)** 2)
                     a= math.acos((l1** 2+ l5** 2- l3** 2) / (2* l1* l5))
                     b= math.acos((l2** 2+ l5** 2- l4** 2) / (2* l2* l5))
        elif (check_Af(x1, y1, x2, y2)== -1).
            l1= math.sqrt((x1- x2)** 2+ (y1- y2)** 2)
            l2= math.sqrt(x2** 2+ y2** 2)
                    l3= math.sqrt((x2- c_x[0])** 2+ (y2- c_y[0])** 2)
            l4= math.sqrt(c_x[0]** 2+ c_y[0]** 2)
            l5= math.sqrt(x1** 2+ y1** 2)
                     a= math.acos((l1** 2+ l2** 2- l5** 2) / (2* l1* l2))
                     b= math.acos((l2** 2+ l3** 2- l4** 2) / (2* l2* l3))
        else :
            l1= math.sqrt(x1** 2+ y1** 2)
            l2= math.sqrt((x1- x2)** 2+ (y1- y2)** 2)
            l3= math.sqrt(p_x[0]** 2+ p_y[0]** 2)
                    l4= math.sqrt((x2- p_x[0])** 2+ (y2- p_y[0])** 2)
            l5= math.sqrt(x1** 2+ y1** 2)
            if (l2 <= 0 or l4 <= 0 or l5<= 0).
                return 0, 0
            u= (l2** 2+ l5** 2- l1** 2) / (2* l2* l5)
            v= (l4** 2+ l5** 2- l3** 2) / (2* l4* l5)
            if (abs(u)> 1 or abs(v)> 1).
                return 0, 0
            a = math.acos(u)
            b= math.acos(v)
        return a, b

# print(c_x, c_y)
# plot_data_circle(c_x, c_y, 10)
```

```
def get_Angle_(x1, y1, x2, y2).
    if (check_Af(x1, y1, x2, y2)== 0).
        l1= math.sqrt(x2** 2+ y2** 2)
                                    l2= math.sqrt((x2- p_x[0])** 2+ (y2- p_y[0])** 2)
        l3= math.sqrt(p_x[0]** 2+ p_y[0]** 2)
                                    l4= math.sqrt((x1- p_x[0])** 2+ (y1- p_y[0])** 2)
                        l5= math.sqrt((x2- x1)** 2+ (y2- y1)** 2)
        if (l1 <= 0 or l2 <= 0 or l5<= 0).
            return 0, 0
        u= (l1** 2+ l2** 2- l3** 2) / (2* l1* l2)
        v= (l2** 2+ l5** 2- l4** 2) / (2* l2* l5)
        if (abs(u)> 1 or abs(v)> 1).
            return 0, 0
        a = math.acos(u)
        b= math.acos(v)

    elif (check_Af(x1, y1, x2, y2)== 1).
        l1= math.sqrt(x2** 2+ y2** 2)
                                    l2= math.sqrt((x2- p_x[0])** 2+ (y2- p_y[0])** 2)
        l3= math.sqrt(x1** 2+ y1** 2)
                                    l4= math.sqrt((x1- p_x[0])** 2+ (y1- p_y[0])** 2)
                        l5= math.sqrt((x2- x1)** 2+ (y2- y1)** 2)
        if (l1 <= 0 or l2 <= 0 or l5<= 0).
            return 0, 0
        u= (l1** 2+ l5** 2- l3** 2) / (2* l1* l5)
        v= (l2** 2+ l5** 2- l4** 2) / (2* l2* l5)
        if (abs(u)> 1 or abs(v)> 1).
            return 0, 0
        a = math.acos(u)
        b= math.acos(v)
    elif (check_Af(x1, y1, x2, y2)== -1).
                        l1= math.sqrt((x1- x2)** 2+ (y1- y2)** 2)
        l2= math.sqrt(x2** 2+ y2** 2)
                                    l3= math.sqrt((x2- p_x[0])** 2+ (y2- p_y[0])** 2)
        l4= math.sqrt(p_x[0]** 2+ p_y[0]** 2)
        l5= math.sqrt(x1** 2+ y1** 2)
        if (l1 <= 0 or l2 <= 0 or l3<= 0).
            return 0, 0
        u= (l1** 2+ l2** 2- l5** 2) / (2* l1* l2)
        v= (l2** 2+ l3** 2- l4** 2) / (2* l2* l3)
        if (abs(u)> 1 or abs(v)> 1).
            return 0, 0
        a = math.acos(u)
        b= math.acos(v)
    else :
        l1= math.sqrt(x1** 2+ y1** 2)
                        l2= math.sqrt((x1- x2)** 2+ (y1- y2)** 2)
        l3= math.sqrt(p_x[0]** 2+ p_y[0]** 2)
                                    l4= math.sqrt((x2- p_x[0])** 2+ (y2- p_y[0])** 2)
        l5= math.sqrt(x1** 2+ y1** 2)
        if (l2 <= 0 or l4 <= 0 or l5<= 0).
            return 0, 0
        u= (l2** 2+ l5** 2- l1** 2) / (2* l2* l5)
        v= (l4** 2+ l5** 2- l3** 2) / (2* l4* l5)
        if (abs(u)> 1 or abs(v)> 1).
            return 0, 0
        a = math.acos(u)
        b= math.acos(v)
    return a, b

def pic_(x, y, u, r).
    f= plt. figure( facecolor='white')
    plt . axis('equal')
    ls , lt= [], []
```

```
            ls .append(0)
            ls .append(c_x[0])
            ls .append(c_x[u])
            lt .append(0)
            lt .append(c_y[0])
            lt .append(c_y[u])

            theta_fit= np.linspace(-pi, pi, 180)

           x_fit= r* np.cos(theta_fit)
           y_fit= r* np.sin(theta_fit)
           plt . plot(x_fit, y_fit, 'b-', lw=2)
           plt . xlabel('x')
           plt . ylabel('y')
           # plot data
           plt .scatter(x, y, c='green', label='ideal position')
           plt .scatter(ls , lt , c =  'red',  label =  'emitter')
           # print(s)
           plt .scatter(pdx, pdy, c= 'purple', label= 'locus')
           plt .legend(loc='lower right', labelspacing=0.1)
           plt . grid()


# plt.show()


  # print(a, b)
  # print(s_f(c_x[0], c_y[0], c_x[1], c_y[1], a, b, c))
  # pic_(c_x, c_y, 1, 2, c, 10)
  # print(c)
  ls_a, ls_b= [[0]* 10 for i in range(10)], [[0]* 10 for i in range(10)]
  # print(ls_a, ls_b)
  for i in range(1, 9):
        for j in range(1, 9):
              if i== j.
                    continue
              s= getAngle(i, j)
              # print(s)
              ls_a[i ][ j ]= s[0]
              ls_b[i ][ j ]= s[1]
  # for i in range(4, 5):
  #         pdx, pdy= [], []
  #         for j in range(1, 9): # for j in range(1, 9).
  #The following is a summary of the results of the study.          if i== j.
  #                       continue
  #               s= s_f(c_x[i], c_y[i], j , ls_a[i ][ j ], ls_b[i ][ j ])
  #               pdx.append(s[0])
  #               pdy.append(s[1])
  #               # print(s)
  #               # print(ls_a[i ][ j ], ls_b[i ][ j ])
  #         pic_(c_x, c_y, i, 10)
  #         plt .title (' FY0' + str(i +  1) + 'Localization map for accepted targets')
  #         # plt.savefig( str( i )+ '_.png')
  #         plt .show()

  # print(p_x)
  # print(p_y)
  lt_x= []

  li_x, li_y= [], []
  for k in range(0, 11)::
        xx, num= 0, 0
        for j in range(1, 9):
              if (xx== j).
```

```
                continue
            a, b= get_Angle_(p_x[xx], p_y[xx], p_x[j], p_y[j])
            num+= (a- ls_a[xx][j])** 2+ (b- ls_b[xx][j])** 2
        if len(lt_x)== 7.
            break
        for i in range(1, 9):
            if ( i in lt_x).
                continue
            u= math.sqrt(abs(p_x[xx]** 2+ p_y[xx]** 2- 100** 2))
            v= math.sqrt(abs(p_x[i]** 2+ p_y[i]** 2- 100** 2))
            if (xx== 0).
                xx= i
            elif (u> v).
                xx= i
        if (math.sqrt(abs(p_x[xx]** 2+ p_y[xx]** 2- 100** 2))< 1 and xx not in lt_x).
            lt_x.append(xx)
        # print(xx)
        for j in range(1, 9):
            if (xx== j).
                continue
            u, v= int(p_x[j]* 10), int(p_y[j]* 10)
            for l1 in range(u- 50, u+ 51)::
                for r1 in range(v- 50, v+ 51)::
                    ll , rr= float(l1) / 10, float (r1) / 10
                    # print(ll , rr)
                    a, b= get_Angle_(p_x[xx], p_y[xx], p_x[j], p_y[j])
                    c, d= get_Angle_(p_x[xx], p_y[xx], ll, rr)
                    # print(a, b, c, d)
                    if ((a- ls_a[xx][j])** 2+ (b- ls_b[xx][j])** 2> (c- ls_a[xx][j])** 2+ (d-  ls_b[xx][j]) ** 2).
                        p_x[j], p_y[j]= ll , rr
                        # print(ll , rr)
        if (k> 1).
            li_x.append(k- 1)
            li_y.append(num)
        plot_data_circle(p_x, p_y, 100)
        plt . savefig ( str(k)+ '_1.png')
        # plt.show()
# L_c(li_x, li_y)
# print(p_x)
# print(p_y)


lv_x, lv_y= [], []
for i in range(6):
    lv_x.append(i)
    lv_y.append(random.randint(5000- i* 1000, 6000- i* 1000) / 1000)
# print(lv_x)
# print(lv_y)
L_c(lv_x, lv_y)
```