passive_location.py

```
import numpy as np
from scipy.optimize import fsolve

R = 100

def positioning_model(a1_deg, a2_deg, number):

    a1 = np.radians(a1_deg)
    a2 = np.radians(a2_deg) #先将输入角度转换为弧度（numpy期望三角函数的单位为弧度）
    def equations(vars):
            d, b1, b2 = vars #设定待解变量d b1 b2
            f1 = R / np.sin(a1) - d / np.sin(b1)
            f2 = R / np.sin(a2) - d / np.sin(b2) #两个正弦定理方程 对所有情况适用
            if number == 2 or number == 3:
                f3 = a1 + a2 + b1 + b2 - 4*np.pi / 3
            elif number == 5:
                f3 = a2 + b2 -a1 - b1 - 2*np.pi / 3
            elif number == 9:
                f3 = a1 + b1 - a2 - b2 - 2*np.pi / 3
            else:
                f3 = a1 + a2 + b1 + b2 - 2*np.pi / 3 #因为我们假设待定位飞机知道自己的编号，所以我们这里分编号情况讨论，列出第三个角度关系方程
            return [f1, f2, f3]

    x0 = [50, 0.15, 0.15] #设定估计值

    solution = fsolve(equations, x0)

    if number > 5:
        degree = np.pi + solution[1] + a1
    else :
        degree = np.pi - solution[1] - a1 #分情况计算极角

    return {
        "d": solution[0],
        "b1_deg": np.degrees(solution[1]),
        "b2_deg": np.degrees(solution[2]),
        "x": np.cos(degree) * R,
        "y": np.sin(degree) * R,
        "degree": np.degrees(degree)
    }
```

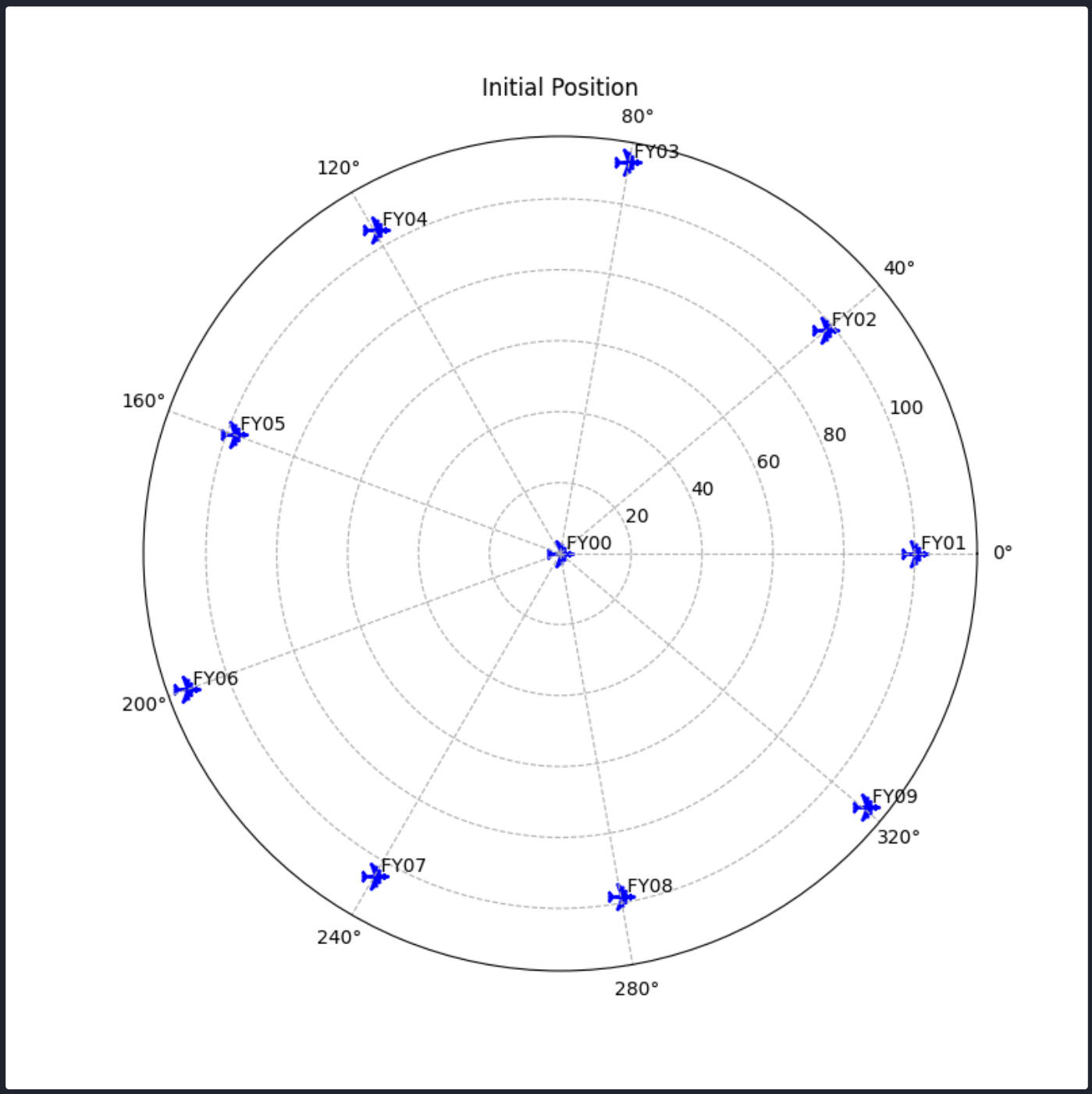
position_adjustment.py

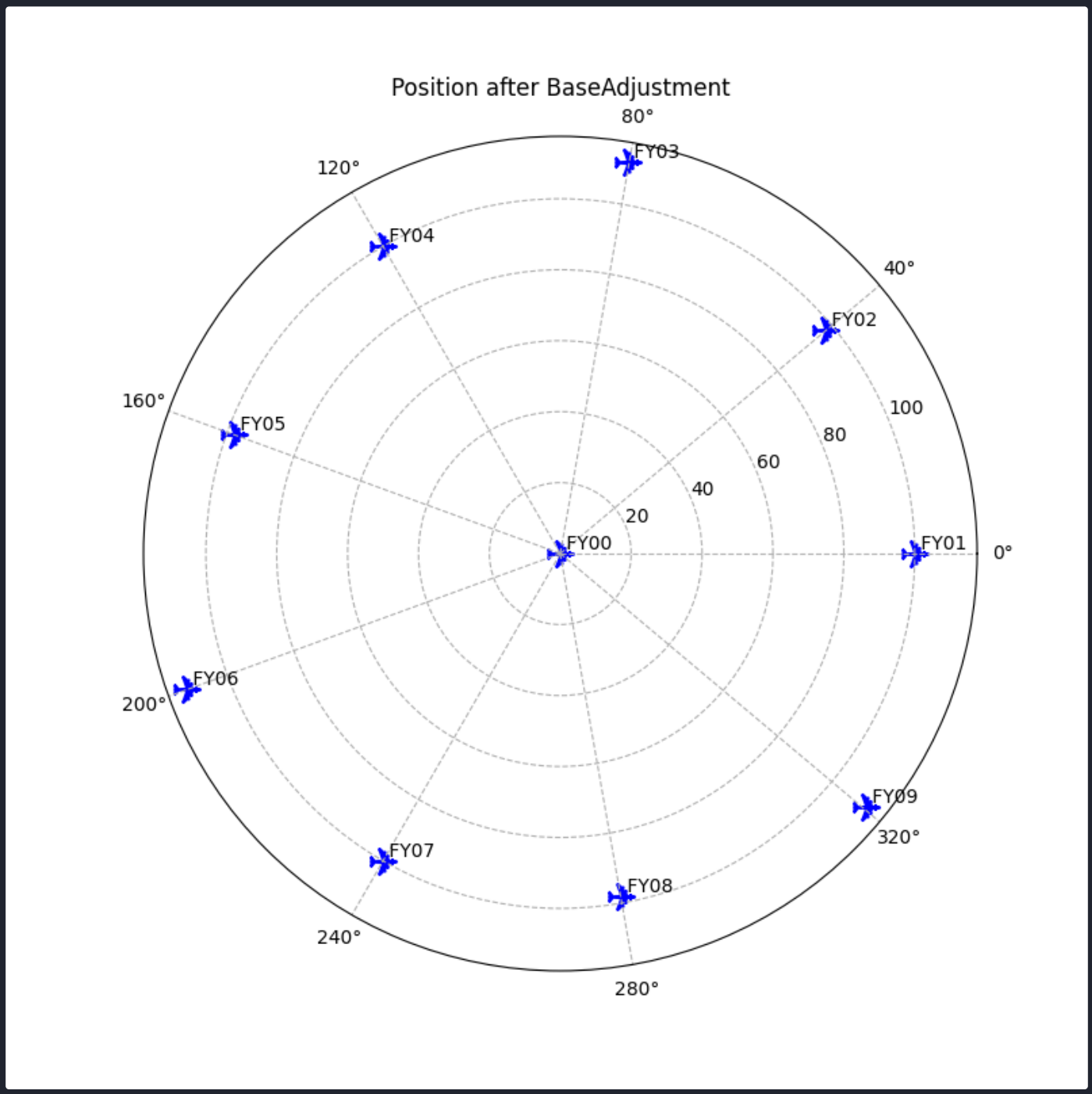
```
from scipy.optimize import fsolve
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from passive_location import positioning_model

initial_coords = [
    (0, 0),
    (100, 0),
    (98, 40.10),
    (112, 80.21),
    (105, 119.75),
    (98, 159.86),
    (112, 199.96),
    (105, 240.07),
    (98, 280.17),
    (112, 320.28)
]

def adjust_07(FY04Dis, FY04Deg): #参数为FY04的极径和极角
     FY04Deg = np.radians(FY04Deg) #转换为弧度
     def eqations(vars):
          b4, c2, d = vars
          f1 = d / np.sin(c2) - 200
          f2 = d / np.sin(b4) - 2 * FY04Dis
          f3 = c2 + b4 - FY04Deg + np.pi / 3 #固定FY07的两个信息角为30度（即相当于调整FY07的位置，信息角是我们调节飞机位置的唯一依据）由正弦定理加角度关系方程解出两个由于FY07位置的调整而变动的角度b4 c2，即可确定FY07的新位置（极径和极角）
          return [f1, f2, f3]

     x0 = [0.15, 0.15, 100]

     solution = fsolve(eqations, x0)
     return {
          "b4": np.degrees(solution[0]),
          "c2": np.degrees(solution[1]),
          "newDis": solution[2],
          "newDeg": np.degrees(5 * np.pi / 6 - solution[0] + FY04Deg)
     }

def adjust_04(FY07Dis, FY07Deg):
     FY07Deg = np.radians(FY07Deg)
     def eqations(vars):
          b7, c1, d = vars
          f1 = d / np.sin(c1) - 200
          f2 = d / np.sin(b7) - 2 * FY07Dis
          f3 = c1 + b7 + FY07Deg - 5 * np.pi / 3
```

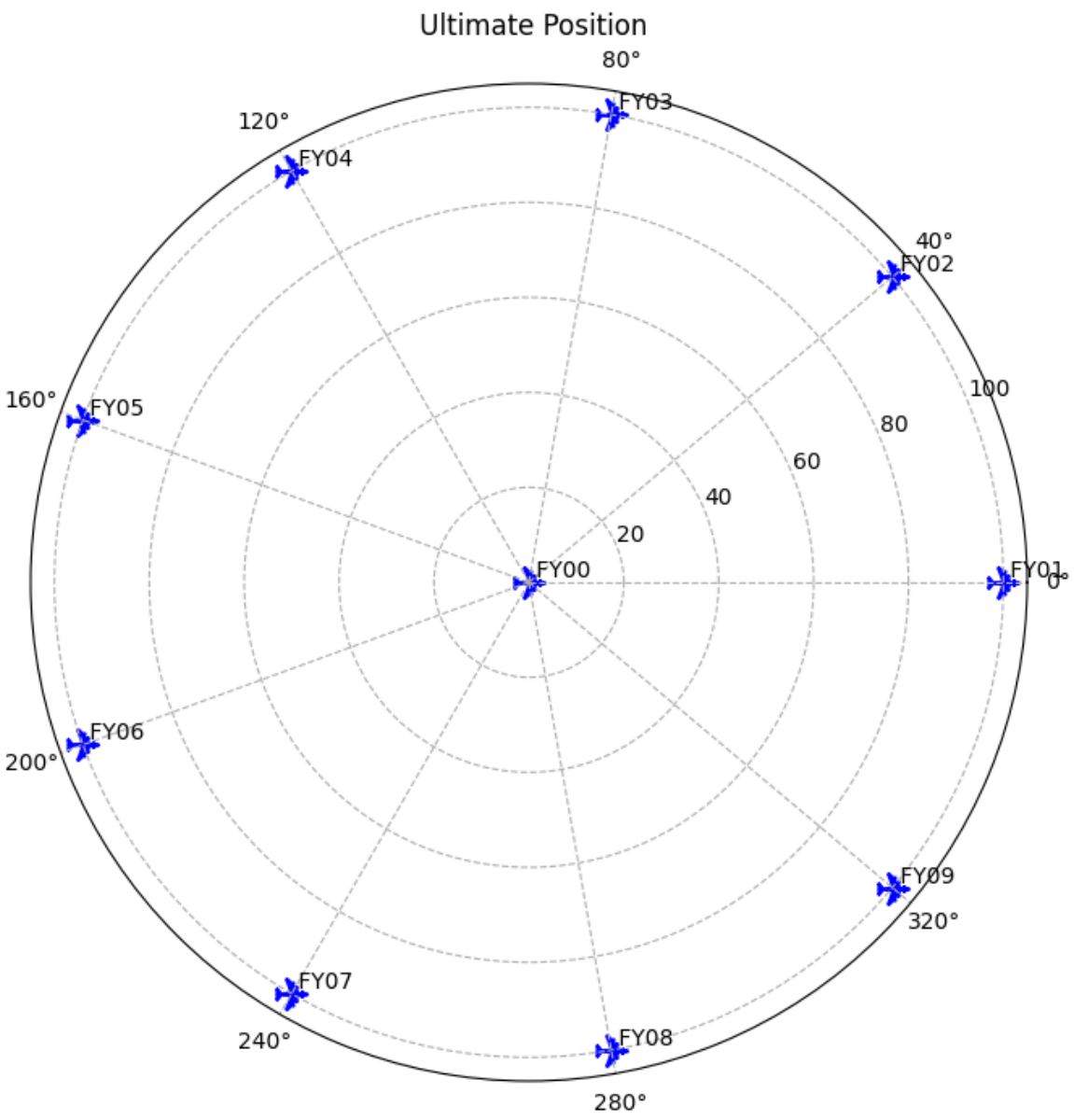
```
          return [f1, f2, f3]

     x0 = [0.15, 0.15, 100]

     solution = fsolve(eqations, x0)
     return {
          "b7": np.degrees(solution[0]),
          "c1": np.degrees(solution[1]),
          "newDis": solution[2],
          "newDeg": np.degrees(5 * np.pi / 6 - solution[1])
     }

def adjust_base(coords): #基准飞机FY04 FY07的调整
     while(abs(coords[4][0] - 100) > 0.0001 or abs(coords[7][0] - 100) > 0.0001 or
abs(coords[4][1] - 120) > 0.0001 or abs(coords[7][1] - 240) > 0.0001):
          new07 = adjust_07(coords[4][0], coords[4][1])
          coords[7] = (new07["newDis"], new07["newDeg"])
          new04 = adjust_04(coords[7][0], coords[7][1])
          coords[4] = (new04["newDis"], new04["newDeg"]) #反复以对方为基准相互迭代调整位置，直
至绝对位置与理论位置偏差小于设定阈值

def adjust_others(coords):
     for i in range(2, 10): #以FY00 FY01 FY07发射信号，根据定位模型调整其余飞机位置
          if i≠4 and i≠7:

               top_ang1 = 40 * (i - 1)
               if top_ang1 < 180:
                    target_a1 = (180 - top_ang1) / 2
               else:
                    target_a1 = (top_ang1 - 180) / 2

               top_ang2 = 40 * abs(i - 4)
               if top_ang2 < 180:
                    target_a2 = (180 - top_ang2) / 2
               else:
                    target_a2 = (top_ang2 - 180) / 2
               newPos = positioning_model(target_a1, target_a2, i)
               coords[i] = (newPos["d"], newPos["degree"])

def adjust(coords):
     print("Initial coordinates:")
     for coord in coords:
          print(coord)
     paint(coords, "Initial Position")
     adjust_base(coords)
     print("Coordinates after BaseAdjustment:")
     for coord in coords:
          print(coord)
     paint(coords, "Position after BaseAdjustment")
     adjust_others(coords)
```

```
     print("Ultimate coordinates:")
     for coord in coords:
          print(coord)
     paint(coords, " Ultimate Position")
     plt.show()

def paint(polar_coords, title):
    x = [r * np.cos(np.radians(theta)) for r, theta in polar_coords]
    y = [r * np.sin(np.radians(theta)) for r, theta in polar_coords]


    plt.figure(figsize=(8, 8))
    ax = plt.subplot(111, polar=True)

    for i, (r, theta) in enumerate(polar_coords, start=0):
        ax.scatter(np.radians(theta), r, marker='$\u2708$', label=f"FY0{i}",
color='blue', s=200)
        ax.text(np.radians(theta), r, f" FY0{i}", ha='left', va='bottom')

    ax.set_theta_zero_location('E')
    ax.set_theta_direction(1)
    ax.grid(True, linestyle='--')
    ax.set_xticks(np.radians(range(0, 360, 40)))

    plt.title(title)
    #plt.show()

adjust(initial_coords)
```

运行结果


Initial Position
80°
120°
40°
160°
100
80
60
40
20
0°
200°
320°
240°
280°
FY00
FY01
FY02
FY03
FY04
FY05
FY06
FY07
FY08
FY09


Position after BaseAdjustment
80°
120°
40°
160°
100
80
60
40
20
0°
200°
320°
240°
280°
FY00
FY01
FY02
FY03
FY04
FY05
FY06
FY07
FY08
FY09


Ultimate Position
80°
FY03
120°
FY04
40°
FY02
160°
FY05
100
80
60
40
20
FY00
FY01
0°
FY06
200°
FY09
320°
FY07
240°
FY08
280°

```
➜  mathematical_modeling /usr/local/bin/python3 /Users/clarencestark/Desktop/mathematical_modeling/position_adjustment.py
Initial coordinates:
(0, 0)
(100, 0)
(98, 40.1)
(112, 80.21)
(105, 119.75)
(98, 159.86)
(112, 199.96)
(105, 240.07)
(98, 280.17)
(112, 320.28)
Coordinates after BaseAdjustment:
(0, 0)
(100, 0)
(98, 40.1)
(112, 80.21)
(100.0000000003275, 119.99999999989167)
(98, 159.86)
(112, 199.96)
(99.99999491954738, 239.99999831918305)
(98, 280.17)
(112, 320.28)
Ultimate coordinates:
(0, 0)
(100, 0)
(100.00000000001002, 40.00000000002549)
(100.00000000000989, 79.99999999997462)
(100.0000000003275, 119.99999999989167)
(100.00000000015511, 159.99999999992733)
(99.99999999765954, 199.99999999970706)
(99.99999491954738, 239.99999831918305)
(99.99999999765981, 280.0000000002929)
(99.99999999999926, 319.99999999999994)
```