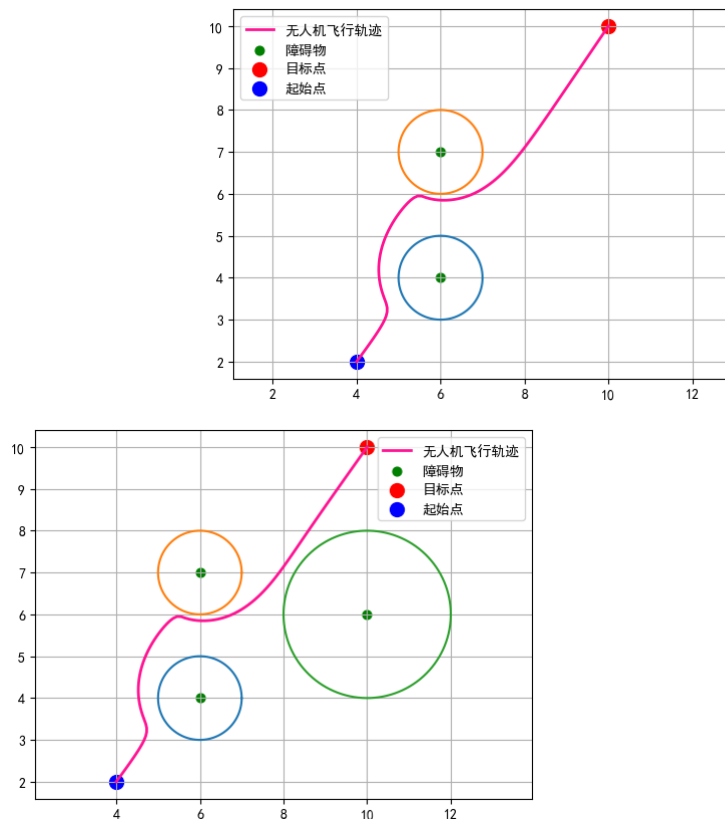


APF Python实现

二维

二维的APF比较简单，参见博客<https://blog.csdn.net/junshen1314/article/details/50472410>即可。



```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from pylab import * #解决matplotlib无法显示中文问题
4 mpl.rcParams['font.sans-serif'] = ['SimHei']
5
6 class APF:
7     def __init__(self):
8         self.obstacle = np.array([[6, 4],
9                                     [6, 7],
10                                    ])
11         self.Obstacle = np.array([1,1]) #仅做测试
12         self.qgoal = np.array([10,10])
13         self.x0 = np.array([4,2])
14         self.stepSize = 0.1
15         self.iter = 1000
16         self.epsilon = 0.8 #引力因子
17         self.eta = 0.2 #斥力因子
18         self.dgoal = 5
19         self.r0 = 4
20         self.path = self.x0.copy()
21         self.path = self.path[np.newaxis,:]
22         self.threshold = 0.5
23
```

```

24     def distanceCost(self, point1, point2):
25         return np.sqrt(np.sum((point1 - point2)**2))
26
27     def attraction(self, q, qgoal, dgoal, epsilon):
28         r = self.distanceCost(q, qgoal)
29         if r <= dgoal:
30             fx = epsilon * (qgoal[0] - q[0])
31             fy = epsilon * (qgoal[1] - q[1])
32         else:
33             fx = dgoal * epsilon * (qgoal[0] - q[0]) / r
34             fy = dgoal * epsilon * (qgoal[1] - q[1]) / r
35         return np.array([fx, fy])
36
37     def differential(self, q, other):
38         output1 = (q[0] - other[0]) / self.distanceCost(q, other)
39         output2 = (q[1] - other[1]) / self.distanceCost(q, other)
40         return np.array([output1, output2])
41
42     def repulsion(self, q, obstacle, r0, eta, qgoal):
43         f0 = np.array([0, 0])
44         Rq2qgoal = self.distanceCost(q, qgoal)
45         for i in range(obstacle.shape[0]):
46             r = self.distanceCost(q, obstacle[i, :])
47             if r <= r0:
48                 tempfvec = eta * (1 / r - 1 / r0) * Rq2qgoal ** 2 / r ** 2 *
self.differential(q, obstacle[i, :]) \
49                     + eta * (1/r - 1/r0) ** 2 * Rq2qgoal *
self.differential(q, qgoal)
50                 f0 = f0 + tempfvec
51             else:
52                 tempfvec = np.array([0, 0])
53                 f0 = f0 + tempfvec
54         return f0
55
56     def loop(self):
57         q = self.x0.copy()          #初始化位置
58         for i in range(self.iter):
59             Attraction =
self.attraction(q, self.qgoal, self.dgoal, self.epsilon)
60             Repulsion =
self.repulsion(q, self.obstacle, self.r0, self.eta, self.qgoal)
61             compositeForce = Attraction + Repulsion
62             unitCompositeForce = compositeForce /
np.sqrt(np.sum((compositeForce) ** 2))
63             q = q + self.stepSize * unitCompositeForce
64             self.path = np.vstack((self.path, q))
65             if self.distanceCost(q, self.qgoal) < self.threshold:
66                 self.path = np.vstack((self.path, self.qgoal))
67                 break
68     def draw(self):
69         plt.scatter(self.obstacle[:, 0], self.obstacle[:, 1], marker='o',
color='green', s=40, label='障碍物')
70         plt.scatter(self.qgoal[0], self.qgoal[1], marker='o', color='red',
s=100, label='目标点')
71         plt.scatter(self.x0[0], self.x0[1], marker='o', color='blue', s=100,
label='起始点')

```

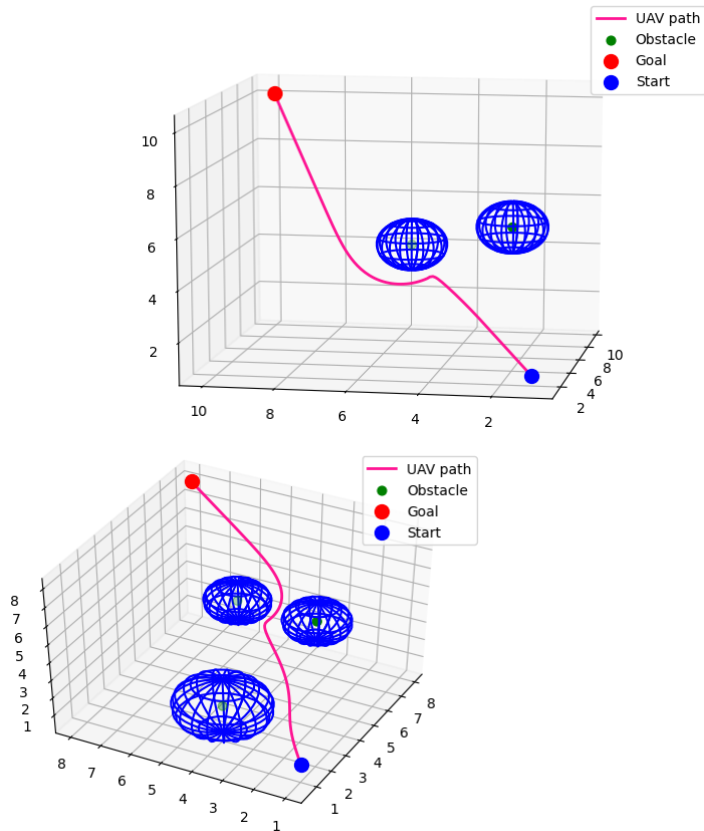
```

72 plt.plot(self.path[:,0],self.path[:,1],color="deeppink",linewidth=2,label =
    '无人机飞行轨迹')
73 plt.legend(loc='best') # 设置 图例所在的位置 使用推荐位置
74 plt.grid()
75 for i in range(self.Robstacle.shape[0]):
76     self.drawCircle(self.obstacle[i,:],self.Robstacle[i])
77 plt.axis('equal')
78 plt.show()
79
80 def drawCircle(self,pos,r): #仅做测试
81     theta = np.arange(0, 2 * np.pi, 0.01)
82     a = pos[0]
83     b = pos[1]
84     x = a + r * np.cos(theta)
85     y = b + r * np.sin(theta)
86     plt.plot(x, y)
87
88 def calculateTotalDistance(self):
89     sum = 0
90     for i in range(self.path.shape[0]-1):
91         sum += self.distanceCost(self.path[i,:],self.path[i+1,:])
92     return sum
93
94 if __name__ == "__main__":
95     apf = APF()
96     apf.loop()
97     apf.draw()
98     print("轨迹距离为:",apf.calculateTotalDistance())

```

三维

刚刚将二维APF升级为了三维版本，其实基本就是把所有数据格式增加了一列而已，至于矢量求导那也是很有规律性的，唯一变化大点的就是绘制部分了，绘制球的时候没有采用封闭式，而采用了框架式，这样方便看到轨迹如果与球相交后内部的情况。



看起来轨迹规划得还可以，但是还是会出现局部最优点以及轨迹穿过障碍物的情况。

```

1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  class APF:
5      def __init__(self):
6          self.obstacle = np.array([[5,5,5],
7                                     [4,2,6],
8                                     [2,4,2]]) #障碍物坐标
9          self.RObstacle = np.array([1,1,1.5]) #在apf中这个障碍物半径不会影响轨迹
10         self.qgoal = [8,8,8] #目标点
11         self.x0 = np.array([1,1,1]) #轨迹起始点
12         self.stepSize = 0.1 #物体移动的固定步长
13         self.iter = 1000 #迭代次数
14         self.epsilon = 0.8 #引力因子
15         self.eta = 0.2 #斥力因子
16         self.dgoal = 5 #当q与qgoal距离超过它时将衰减一部分引力
17         self.r0 = 4 #斥力超过这个范围后将不复存在
18         self.path = self.x0.copy()
19         self.path = self.path[np.newaxis,:] #增加一个维度
20         self.threshold = 0.5 #q与qgoal距离小于它时终止训练或者仿真
21
22     def distanceCost(self,point1,point2): #求两点之间的距离函数
23         return np.sqrt(np.sum((point1 - point2) ** 2))
24
25     def attraction(self,q,qgoal,dgoal,epsilon): #计算引力的函数
26         r = self.distanceCost(q,qgoal)
27         if r <= dgoal:
28             fx = epsilon * (qgoal[0] - q[0])
29             fy = epsilon * (qgoal[1] - q[1])
30             fz = epsilon * (qgoal[2] - q[2])
31         else:

```

```

32         fx = dgoal * epsilon * (qgoal[0] - q[0]) / r
33         fy = dgoal * epsilon * (qgoal[1] - q[1]) / r
34         fz = dgoal * epsilon * (qgoal[2] - q[2]) / r
35         return np.array([fx, fy, fz])
36
37     def differential(self, q, other):
38         output1 = (q[0] - other[0]) / self.distanceCost(q, other)
39         output2 = (q[1] - other[1]) / self.distanceCost(q, other)
40         output3 = (q[2] - other[2]) / self.distanceCost(q, other)
41         return np.array([output1, output2, output3])
42
43     def repulsion(self, q, obstacle, r0, eta, qgoal): #计算斥力的函数
44         f0 = np.array([0, 0, 0]) #初始化斥力的合力
45         Rq2qgoal = self.distanceCost(q, qgoal)
46         for i in range(obstacle.shape[0]):
47             r = self.distanceCost(q, obstacle[i, :])
48             if r <= r0:
49                 tempfvec = eta * (1 / r - 1 / r0) * Rq2qgoal ** 2 / r ** 2
50                 * self.differential(q, obstacle[i, :]) \
51                 + eta * (1/r - 1/r0) ** 2 * Rq2qgoal *
52                 self.differential(q, qgoal)
53                 f0 = f0 + tempfvec
54             else:
55                 tempfvec = np.array([0, 0, 0])
56                 f0 = f0 + tempfvec
57         return f0
58
59     def loop(self): #循环仿真
60         q = self.x0.copy()
61         for i in range(self.iter):
62             Attraction =
63             self.attraction(q, self.qgoal, self.dgoal, self.epsilon) #计算引力
64             Repulsion =
65             self.repulsion(q, self.obstacle, self.r0, self.eta, self.qgoal) #计算斥力
66             compositeForce = Attraction + Repulsion
67             #合力 = 引力 + 斥力
68             unitCompositeForce = compositeForce /
69             np.sqrt(np.sum((compositeForce) ** 2)) #力单位化, apf中力只用来指示移动方向
70             q = q + self.stepSize * unitCompositeForce #计算下一位置
71             self.path = np.vstack((self.path, q)) #记录轨迹
72             if self.distanceCost(q, self.qgoal) < self.threshold: #当与
73                 goal之间距离小于threshold时结束仿真, 并将goal的坐标放入path
74                 self.path = np.vstack((self.path, self.qgoal))
75                 break
76
77     def draw(self):
78         self.ax = plt.axes(projection='3d')
79         self.ax.scatter3D(self.obstacle[:, 0],
80 self.obstacle[:, 1], self.obstacle[:, 2], marker='o', color='green', s=40,
81 label='Obstacle')
82         self.ax.scatter3D(self.qgoal[0], self.qgoal[1], self.qgoal[2],
83 marker='o', color='red', s=100, label='Goal')
84         self.ax.scatter3D(self.x0[0], self.x0[1], self.x0[2], marker='o',
85 color='blue', s=100, label='Start')
86
87         self.ax.plot3D(self.path[:, 0], self.path[:, 1], self.path[:, 2], color="deeppin
88 k", linewidth=2, label = 'UAV path')
89         plt.legend(loc='best') # 设置 图例所在的位置 使用推荐位置

```

```

77     plt.grid()
78     for i in range(self.Robstacle.shape[0]):
79         self.drawSphere(self.obstacle[i,:],self.Robstacle[i])
80     plt.show()
81
82     def drawSphere(self,center,radius):
83         u = np.linspace(0, 2 * np.pi, 20)
84         v = np.linspace(0, np.pi, 20)
85         x = radius * np.outer(np.cos(u), np.sin(v)) + center[0]
86         y = radius * np.outer(np.sin(u), np.sin(v)) + center[1]
87         z = radius * np.outer(np.ones(np.size(u)), np.cos(v)) + center[2]
88         self.ax.plot_wireframe(x,y,z, cstride = 4, color = 'b')
89
90     def calculateTotalDistance(self):
91         sum = 0
92         for i in range(self.path.shape[0]-1):
93             sum += self.distanceCost(self.path[i,:],self.path[i+1,:])
94         return sum
95
96     if __name__ == "__main__":
97         apf = APF()
98         apf.loop()
99         apf.draw()
100        print("轨迹距离为:",apf.calculateTotalDistance())

```