题目

摘要

第一段：针对自己选择的题目，说明自己用了什么方法来解决的（这类题属于哪种典型的问题），其中利用了哪些关键的算法，再说出自己的所建模型的创新点。没有创新点，也可以说自己所建的模型相比较于其它的是一个很好的方案。

第二段：针对问题一中的具体问题，进行分析和求解，几句话介绍自己是怎么解决的，有数字结果的也可以直接贴结果。

。

。

第三段：问题二中，类比于第二段。

。

。

。

第四段：问题三中，类比于第三段。

。

。

。

第五段：问题四中，类比于第四段。

。

。

。

第六段：如果有问题五，类比于第五段，没有就结束，也可以写一下团队的想法。

。

。

。

关键词：

1 问题重述

1.1 问题背景

。。。

。

。

。

。

1.2 问题提出

。。。

。

。

问题一:

。

。

。

问题二:

。

。

。

问题三:

。

。

。

2 问题分析

2.1 问题一的分析

问题一需要。。。

。

。

。

。

2.2 问题二的分析

在问题一的基础上。。。

。

。

。

。

2.3 问题三的分析

在问题二的基础上。。。

。

。

。

。

2.4 总思路图（可选）

图 1. 总思路图（随便找的网图）

3 模型假设

1. 。。。

   。

2. 。。。

   。

3. 。。。

   。

4. 。。。

   。

4 符号说明

符号	说明	单位
d	两点间的距离	m
t	时间变量	s
v	速度	m/s
l	物体长度或路径长度	m
(x_i, y_i)	第i个点的平面坐标	-
θ_i	第i个角度变量	rad
A_i	第i个区域的面积	m^2
B_i	第i个模型的系数矩阵	-
C_i	第i类对象的成本或代价	元
α, β	模型参数（如权重系数）	-
ρ	密度	$\mathrm{kg/m}^3$
λ	到达率或衰减系数	1/s
$T_{\max}$	最大时间阈值	s
N	样本总数或迭代次数	-
R^2	拟合优度或决定系数	-
ε	误差项或极小量	-
∇f	函数f的梯度	-
$\sum_{i=1}^{n}$	从1到n的求和运算	-

5 模型的建立与求解

ps：这部分因人而异

5.1 问题一的求解

5.1.1 建模思路/解题步骤

。

。

。

5.1.2 运算方程/运算方法

。

。

（示例）

等距螺线的极坐标方程为

$$r(\theta)=a+b\theta \tag{1}$$

其中，r为极径；θ为极角；a和b均为实数，由题意可知$a=0$。

螺距p的大小可表示为

$$p=r(\theta+2\pi)-r(\theta)=b\cdot 2\pi$$

结合以上分析，得到

$$r(\theta)=\frac{p}{2\pi}\theta \tag{2}$$

将极坐标转换为直角坐标

$$\begin{cases} x=r(\theta)\cdot\cos\theta \\ y=r(\theta)\cdot\sin\theta \end{cases} \tag{3}$$

这样，就可以计算出舞龙队在平面上任一角度θ下的具体位置。

5.1.3 继续求解步骤（数据表、图）

。

。

。

表格 1. ***变化情况

	0s	60s	120s	180s	240s	300s
龙头 (m/s)	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
第一节龙身 (m/s)						
第二节龙身 (m/s)						
第三节龙身 (m/s)						
第四节龙身 (m/s)						
龙尾 (m/s)						

5.1.4 继续求解步骤（数据表、图）

- ◦

◦

◦

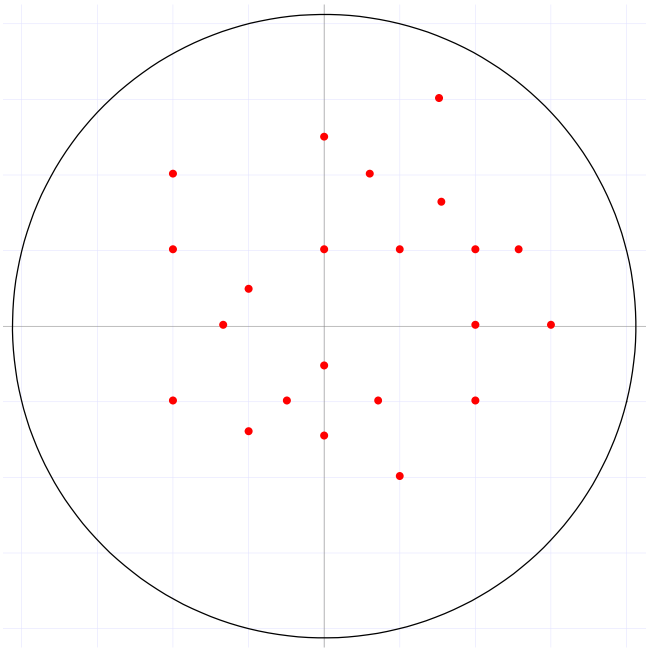

图 2. ***位置变化图

5.2 问题二的求解

5.2.1 建模思路/解题步骤

。

。

。

5.2.2 运算方程/运算方法

。

。

。

5.2.3 继续求解步骤（数据表、图）

。

。

。

5.2.4 继续求解步骤（数据表、图）

。

。

。

5.3 问题三的求解

5.3.1 建模思路/解题步骤

◦

◦

◦

5.3.2 运算方程/运算方法

◦

◦

◦

5.3.3 继续求解步骤（数据表、图）

◦

◦

◦

5.3.4 继续求解步骤（数据表、图）

◦

◦

◦

5.4 问题四的求解

5.4.1 建模思路/解题步骤

。

。

。

5.4.2 运算方程/运算方法

。

。

。

5.4.3 继续求解步骤（数据表、图）

。

。

。

5.4.4 继续求解步骤（数据表、图）

。

。

。

6 模型的评价

6.1 模型的优点

1.

2.

3.

6.2 模型的缺点

1.

2.

3.

6.3 模型的改进

1.

2.

3.

7 参考文献

[1] Alex Krizhevsky, Ilya Sutskever, 与 Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, 与 K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, 卷 25, 页面 0. Curran Associates, Inc., 2012.

附录A　支撑材料文件列表

文件名	说明
***-1.py	问题一到问题三的***
***-2.py	…
***-3.py	…
***-4.py	…
***1-1.py	…
***1-2.py	…
***1-3.py	…
***1-4.py	…
***2-1.py	…
***2-2py	…
***2-3.py	…

附录B　支撑材料的所有Python代码

```
"""
文件名: data_processing-1.py
用途: 2023全国大学生数学建模竞赛C题-蔬菜运输优化
      数据预处理模块（数据清洗+特征计算）
"""

import numpy as np
import pandas as pd
from scipy.optimize import linprog
import matplotlib.pyplot as plt

# ==================== 数据预处理函数 ====================
def load_and_clean_data(file_path):
    """
    数据加载与清洗
    参数:
        file_path : str - CSV文件路径
    返回:
        df : DataFrame - 处理后的干净数据
    """
    try:
        df = pd.read_csv(file_path, encoding='gbk')  # 处理中文编码
        df.dropna(inplace=True)  # 删除缺失值
        df = df[df['产量'] > 0]  # 过滤无效产量记录
        return df
    except Exception as e:
        print(f"数据加载失败: {str(e)}")
        return None

# ==================== 优化模型求解 ====================
def transport_optimization(cost_matrix, supply, demand):
    """
```

```
    运输问题线性规划求解
    参数:
        cost_matrix : ndarray - 运输成本矩阵 (m×n)
        supply : ndarray - 供应量数组 (m,)
        demand : ndarray - 需求量数组 (n,)
    返回:
        result : dict - 包含优化结果的字典
    """
    # 线性规划求解 (使用单纯形法)
    res = linprog(cost_matrix.flatten(),
                  A_eq=_build_constraints(supply, demand),
                  b_eq=_build_boundary(supply, demand),
                  method='highs')

    return {
        'status': res.status,
        'total_cost': res.fun,
        'schedule': res.x.reshape(cost_matrix.shape)
    }

# ==================== 可视化模块 ====================
def plot_solution(routes, nodes):
    """绘制运输路线图"""
    plt.figure(figsize=(10, 8))
    for (i,j), val in np.ndenumerate(routes):
        if val > 0:
            plt.plot([nodes[i][0], nodes[j][0]],
                     [nodes[i][1], nodes[j][1]],
                     'b-', alpha=0.5, linewidth=val*2)
    plt.scatter(nodes[:,0], nodes[:,1], c='r', s=50)
    plt.title("Optimal Transport Routes")
    plt.xlabel("X Coordinate")
    plt.ylabel("Y Coordinate")
    plt.grid(True)
    plt.savefig('routes.png', dpi=300)

if __name__ == '__main__':
    # 示例数据
    demo_cost = np.random.rand(5,3) * 100
    demo_supply = np.array([20, 30, 15, 25, 10])
    demo_demand = np.array([40, 30, 20])

    # 执行优化
    solution = transport_optimization(demo_cost, demo_supply, demo_demand)
    print(f"最优总成本: {solution['total_cost']:.2f} 元")
```