基于"XXX"的低碳建筑研究

摘要

关键词

1 问题重述

1.1 问题背景

“双碳”即碳达峰与碳中和的简称，我国力争 2030 年前实现碳达峰，2060 年前实现碳中和。“双碳”战略倡导绿色、环保、低碳的生活方式。我国加快降低碳排放步伐，大力推进绿色低碳科技创新，以提高产业和经济的全球竞争力。低碳建筑是指在建筑材料与设备制造、施工建造和建筑物使用的整个生命周期内，减少化石能源的使用，提高能效，降低二氧化碳排放量。

1.2 目标任务

问题一：计算给定建筑通过空调调节温度的年碳排放量。

问题二：建立综合评价模型，找出易于量化的指标，评估居住建筑整个生命周期的碳排放。

问题三：基于问题二，考虑建筑生命周期三个阶段的碳排放问题，对江苏省 13 个地级市的居住建筑进行评价，验证模型的有效性。

问题四：建立碳排放预测模型，基于江苏省建筑全过程碳排放的历史数据，对 2023 年江苏省建筑全过程的碳排放量进行预测。

问题五：结合前面的讨论给出江苏省建筑碳减排的政策建议。

2 问题分析

2.1 问题一

问题一要求计算通过空题调节温度产生的年碳排放量。我们需先求出空调制热和制冷的热量，借此通过 COP 和 EER 求出空调消耗的电量，最后转换成碳排放。其中 COP 和 EER 的定义分别为

$$COP = \frac{Q_{heat}}{W}, \qquad EER = \frac{Q_{cold}}{W} \tag{2.1}$$

Q_{heat}/Q_{cold} 指的是单位时间内的制热/制冷量，单位为 J，公式中 W 指的是单位为时间内空调消耗的功率，单位为 W

首先计算出建筑物各个月的能量需求量。设室内温度要维持的温度为 t_{in}，室外温度为 t_{out}, 当月该地区平均温度为 t_{ave} ，方便起见，我们规定

$$t_{out} = t_{ave}$$

$$t_{in} = \begin{cases} 18°C & t_{out} < 18°C \\ t_{out} & t_{out} \in [18°C, 26°C] \\ 26°C & t_{out} > 26°C \end{cases}$$

我们使用热传导方程计算用来需要制热/制冷的热量，其形式为

$$\Phi = \frac{\lambda \cdot A \cdot |\Delta T|}{\delta} \tag{2.2}$$

其中 Φ 表示传热速率，λ 为导热系数，A 为传热面积，Δ 是室内外温度差，即 $t_{in} - t_{out}$，L 表示材料厚度。

将建筑分成墙、门窗、房顶、地面四个部分，分别计算并累加即可得到需要制热/制冷的热量，设为 Q_{make}，由 COP 和 EER 的定义可得到需电量 Q_{elec} 和热量 Q_{make} 的转化关系

$$Q_{elec} = \begin{cases} \frac{Q_{make}}{EER} & \Delta t < 0 \\ 0 & \Delta t = 0 \\ \frac{Q_{make}}{COP} & \Delta t > 0 \end{cases} \tag{2.3}$$

最后根据需电量与碳排放的换算关系 $m = Q_{elec} \cdot 0.28$ 求出每月碳排放后累加，即得到年度碳排放量。

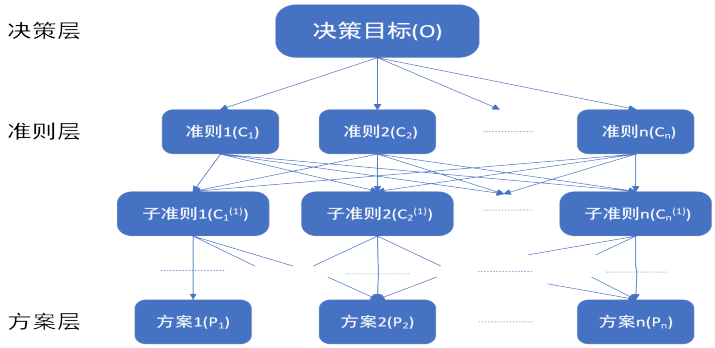

图 1: 层次分析法框架

3 模型假设

4 符号说明

5 模型的建立与求解

5.1 问题一的模型建立与求解

5.2 问题二的模型建立与求解

5.2.1 建立层次结构模型

6 结果检验与误差分析

7 模型评价

8 模型推广与改进

9 参考文献

10 附录

附录 A 问题一

Listing 1: **Question1**

```python
# -*- coding:utf-8 -*-
"""
Project: MathematicsModeling-2023.5
Written by: Evan Wong
DATE: 2023/5/9
TIME: 17:09
"""

import matplotlib.pyplot as plt

monthdays = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
temperatures = [-1, 2, 6, 12, 22, 28, 31, 32, 26, 23, 15, 2]

COP = 3.5
EER = 2.7

length = 4
width = 3
height = 3

roof_thickness = 0.3
roof_thermal = 0.2
roof_square = length * width
windowAndDoor_thickness = 0.1
windowAndDoor_thermal = 1.6
windowAndDoor_square = 5
wall_thickness = 0.3
wall_thermal = 0.3
wall_square = (length + width) * height * 2 - windowAndDoor_square
ground_thickness = 1.0
ground_thermal = 0.25
ground_square = length * width


def get_delta_t(tout):
    if tout < 18:
```

```
        return 18 - tout
    elif tout > 26:
        return 26 - tout
    else:
        return 0


def get_qmake(k, a, t, d):
    return (k * a * t) / d


def get_day_qmake(dt):
    if dt == 0:
        return 0.0

    day_qmake = \
        get_qmake(wall_thermal, wall_square, dt, wall_thickness) + \
        get_qmake(windowAndDoor_thermal, windowAndDoor_square, dt,
            windowAndDoor_thickness) + \
        get_qmake(roof_thermal, roof_square, dt, roof_thickness) + \
        get_qmake(ground_thermal, ground_square, dt, ground_thickness)
    return day_qmake * 86400


def get_month_qmake(month):
    return get_day_qmake( abs(get_delta_t(temperatures[month]))) * monthdays[month]


def get_qelec(qmake, dt):
    if dt < 0:
        return qmake / EER
    elif dt == 0:
        return 0
    else:
        return qmake / COP


def get_carbon_emission(qelec):
    return (qelec / 3600000) * 0.28


carbonEmissions = []
```

```
for i in  range(0, 12):
    qelec = get_qelec(get_month_qmake(i), get_delta_t(temperatures[i]))
    carbonEmissions.append(get_carbon_emission(qelec))

print("Sum of carbon emissions: " +  str( sum(carbonEmissions)) + "kg")

plt.figure(figsize=(10, 6))
plt.bar( range(1,  len(carbonEmissions) + 1), carbonEmissions, fc='g')
plt.title("Carbon Emissions")
plt.xlabel("Month")
plt.ylabel("kg")

plt.show()
```

附录 B　　问题二

Listing 2: **Question2**

```
import numpy as np
import matplotlib.pyplot as plt


cities = ["SuZhou", "NanJing", "NanTong", "WuXi", "ChangZhou"]

discriminantMatrix = np.array([[1, 1/5, 1/5], [5, 1, 1/2], [5, 2, 1]])
eigens = np.linalg.eig(discriminantMatrix)
maxEigenvalue = np. max(eigens[0])
rowAndColumn = np.argwhere(eigens[0] == maxEigenvalue)
maxEigenvector = eigens[1][::-1, rowAndColumn[0]]

RI_LIST = [0, 0.001, 0.58, 0.9, 1.12, 1.24, 1.32, 1.41, 1.45, 1.49, 1.52, 1.54, 1.56,
    1.58, 1.59]
dim = discriminantMatrix.shape[0]
RI = RI_LIST[dim]
CI = (maxEigenvalue - dim) / (dim - 1)
CR = CI / RI
print("CR: " +  str(CR))
print("Passed consistency test") if CR < 0.1 else print("Failed to passed consistency
    test")

Scores = np.array([[48.0, 126.3, 4.8],
```

```
                    [51.5, 134.3, 4.9],
                    [27, 118.7, 3.5],
                    [57.2, 97.8, 2.7],
                    [29, 71.6, 2.8]])

sums = np. sum(Scores, axis=0)
weightedScores = Scores / sums.reshape(1, -1)

weight = maxEigenvector /  sum(maxEigenvector)

finalScores = np.dot(weightedScores, weight)
for i in  range( len(finalScores)):
    print('City {:}, Scores {:}'. format(cities[i], finalScores[i, 0].real))

plt.figure(figsize=(10, 6))
x = [1, 2, 3, 4, 5]
x_label = cities
plt.bar(x, finalScores[:, 0].real, fc='g')
plt.title("Scores given by evaluation model")
plt.xticks(x, x_label)
plt.xlabel("City")
plt.ylabel("Evaluation score")

plt.show()
```

附录 C　　问题三

附录 D　　问题四

附录 E　　问题五