2023 年山东师范大学本科生数学建模竞赛
第一页

承诺书

我们仔细阅读了山东师范大学数学建模竞赛的规则。

我们知道，抄袭别人的成果是违反竞赛章程和参赛规则的，如果引用别人的成果或其他公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

我们郑重承诺，严格遵守竞赛章程和参赛规则，以保证竞赛的公正、公平性。如有违反竞赛章程和参赛规则的行为，我们将受到严肃处理。

我们授权山东师范大学数学建模竞赛指导中心，可将我们的论文以任何形式进行公开展示（包括进行网上公示，在书籍、期刊和其他媒体进行正式或非正式发表等）。

表 1: 队伍信息表

姓名	学院简写	学号	手机	QQ 号	获得建模相关奖项
贾庆志	信工	202211000109	19862178002	2120786474	无
刘权林	信工	202211000151	15523505879	3025684615	无
黄昱峰	信工	202211000139	15717087699	1280797395	无

基于”层次分析法”和”灰色预测模型”的低碳建筑研究

摘要

本文主要研究了在居住建筑的整个生命周期影响碳排放的主要因素和对建筑碳排放的预测，并建立了层次结构模型和灰色预测模型。对于问题一，我们使用热传导方程计算用来需要制热/制冷的热量总和，再根据热量算出空调制冷/供暖所需的电量总和，乘上系数 0.28 即可得出年度碳排放量。对于问题二，我们选择建立层次结构模型并在构建各因素的成对比较矩阵和将个城市的碳排放数据归一化的前提下进行三种方法（和积法，求根法，特征值法）的层次单排序和一致性检验，其中一致性检验主要考察的指标为一致性指标（CI），平均随机一致性指标（RI）和检验系数（CR），其中

$$CR = \frac{CI}{RI}$$

一般地，如果 CR 0.1，则认为该判断矩阵通过一致性检验，$A_{\max}$ 对应的特征向量 W 可以作为排序的权重向量，计算得到最大特征值 $\lambda_{\max}$ 对应的特征向量 T 和权重向量 W，最后将三个权重向量取算术平均作为最终的权重向量。设指标评分矩阵为 P，那么最后的得分矩阵 P 为

$$S = P \cdot W$$

对于问题三，我们基于第二问所建模型，通过建筑生命周期的三个阶段确定代表性强的指标加入模型评价，并将 2021 年江苏省 13 个地级市的居住建筑碳排放作为输入输入到所建立模型中进行综合评价，并将得到的结果排名与实际的排名进行比较。对于问题四我们选择了建立灰色预测模型，并对模型建立进行了级比检验，构造累加生成序列，构建一阶常微分方程工作。对于模型的求解，第一步我们先完成了方程的改写，然后通过最小二乘法求解最佳的参数的近似值，并借此求解微分方程得到拟合函数。最后通过后验差比值 C 的大小来判断模型的预测精度，$C = \frac{\sigma_1}{\sigma_2}$; 可达模型拟合结果与预测和历史记录对比图。对于问题五，我们基于对于前面的问题给出了江苏省筑碳减排的几条政策建议。

关键词：碳排放，热传导方程，层次结构模型，权重，一致性检验，矩阵，

灰色预测、生成序列，微分方程，最小二乘法，最佳参数，模型结果，筑碳减排

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

问题一要求计算空调调节温度产生的年碳排放量，又给出了碳排放量与用电量的关系，因此我们需要求出空调调节温度所用电量。为了进一步简化计算，我们假设室外温度始终为月平均温度，然后我们找出需要开空调

的月份，并且规定开空调的目的是让温度维持在目标温度上届或下届。然后计算出空调制冷或制热所制造的热量，通过 EER 或 COP 算出空调消耗的电量，即达成目标，最后换算即可。

2.2 问题二

问题二要求选取指标构建评价模型评估居住建筑整个生命周期的碳排放。我们选择了使用层次分析法来构建评价模型，将指标作为准则层的准则，构建起层次分析法的框架。

2.3 问题三

问题三要求在问题二的基础上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并验证模型的可行性。因此我们在建筑的生命周期三个阶段各自选取了一些指标增加进问题二所构建模型的准则层中，将江苏省 13 个地级市的居住建筑生命周期碳排放量作为输入输入到模型中，将模型给出的评价分进行排序，与实际排名进行对比，进而判断模型的有效性。

2.4 问题四

问题四要求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预测 2023 年江苏省建筑全过程碳排放。注意到数据和预测目标有数据量较少、预测目标较近期的特点，我们选择使用构建灰色预测 GM (1, 1) 模型作为预测模型。

2.5 问题五

经过以上分析，我们对江苏省建筑碳减排的政策的建议如下

1. 建筑设计标准：加强建筑能效等级和节能措施的规定，提高被动设计的实现率。鼓励使用绿色环保材料，减少建筑材料的使用量。

2. 促进可再生能源利用：鼓励建筑采用太阳能、地源热泵等可再生能源设施，减少对传统能源的依赖，降低运行阶段的碳排放量。

3. 改善建筑使用效率：推广智能化节能控制系统，优化空调、照明、通风等系统的运行方式，减少不必要的能源消耗。

4. 推广低碳生活方式：鼓励居民采用低碳出行方式，如步行、骑行、乘坐公共交通工具等，减少个人碳排放量。

5. 废弃物处理：鼓励采用回收利用的废弃物处理方式，减少拆除阶段的碳排放量。

6. 建立碳排放权交易市场：通过建立碳排放权交易市场，引导企业和个人减少碳排放，实现碳排放量的交易和管理。

7. 优化政策环境：加强碳排放标准的规定和执行力度，鼓励和支持低碳技术研发和应用。引导金融机构加大对低碳建筑项目的投资和支持力度。

8. 把节约能源资源放在首位，实行全面节约战略，倡导推广绿色低碳的生产生活方式，大幅提高投入产出效率，持续降低单位产出能源资源消耗和碳排放，从源头和入口形成有效的碳排放控制阀门。

9. 统筹省内外能源资源，推广先进绿色低碳技术和经验。推动省内应对气候变化对外合作有序开展，加强交流合作，不断提高参与度和影响力。

10. 加强低碳社会建设宣传教育，提高全社会对碳达峰、碳中和的认知度和认可度。引导和支持各类市场主体适应低碳发展要求，提升低碳创新水平。以示范创建为载体，推广绿色低碳生产生活方式，扩大绿色低碳产品供给和消费，倡导形成简约适度、绿色低碳的生活方式。

11. 政府和市场共同发挥作用，科技、产业和制度创新协同并进，增强原始创新支撑能力，加快全面数字化，深化能源等相关领域改革，形成有效激励约束机制，构建绿色低碳创新体系。

12. 进工业低碳工艺革新、数字化转型和绿色制造体系建设，加快重点领域对照标杆水平实施节能降碳技术改造，鼓励国有企业、骨干企业开展示范性改造。

13. 大力培育节能环保、资源循环利用、清洁能源等绿色低碳产业，聚焦集成电路、生物医药、人工智能等前沿领域，积极发展新一代信息技术、新材料、新能源汽车等战略性新兴产业。

3 模型假设

1. 假设建筑地面相当于厚度一米的导热层，假设门窗厚度为 0.1m；
2. 假设室外温度为月平均温度，假设空调只在室外温度不在目标温度区间时才开启；
3. 假设开空调的目的是使温度保持在目标温度区间上界或下届（取接近一方），不考虑刚开关空调时室内温度逐渐上升或逐渐下降的情况；
4. 假设各地区不同指标的碳排放为在该指标的得分，不考虑选定指标之外的碳排放；

4 符号说明

见下页表 2。

5 模型的建立与求解

5.1 问题一的模型建立与求解

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

$$COP = \frac{Q_{heat}}{W}, \qquad EER = \frac{Q_{cold}}{W} \tag{5.1}$$

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

$$\Phi = \frac{\lambda \cdot A \cdot |\Delta T|}{\delta} \tag{5.2}$$

其中 Φ 表示传热速率，λ 为导热系数，A 为传热面积，ΔT 是室内外温度差，即 $t_{in} - t_{out}$，δ 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

$$Q_{elec} = \begin{cases} \frac{Q_{make}}{EER} & \Delta t < 0 \\ 0 & \Delta t = 0 \\ \frac{Q_{make}}{COP} & \Delta t > 0 \end{cases} \tag{5.3}$$

表 2: 符号说明

变量	定义	单位
Q_{heat}	空调制热产生热量	J
Q_{cold}	空调制冷产生热量	J
Q_{make}	空调总共产生热量	J
Q_{elec}	空调消耗电能	J
t_{in}	室内温度	$°C$
t_{out}	室外温度	$°C$
t_{ave}	月平均温度	$°C$
ΔT	室内外温差	$°C$
Φ	传热速率	W
λ	导热系数	$W/m^2 \cdot K$
A	传热面积	m^2
δ	材料厚度	m
W	权重向量	/
$\lambda_{\max}$	最大特征值	/
T	最大特征值对应特征向量	/
CI	一致性指标	/
RI	平均随机一致性指标	/
CR	检验系数	/
P	评分矩阵	/
S	得分矩阵	/
$x^{(0)}$	年份-数据序列	/
$x^{(1)}$	累加生成序列	/
$\lambda(k)$	关联系数	/
a	发展系数	/
u	灰色作用量	/
$\hat{a}$	最佳近似发展系数	/
$\hat{u}$	最佳近似灰色作用量	/
C	后验差比值	/
σ_1	历史数据方差	/
σ_2	残差方差	/

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。

5.2 问题二的模型建立与求解

5.2.1 建立层次结构模型

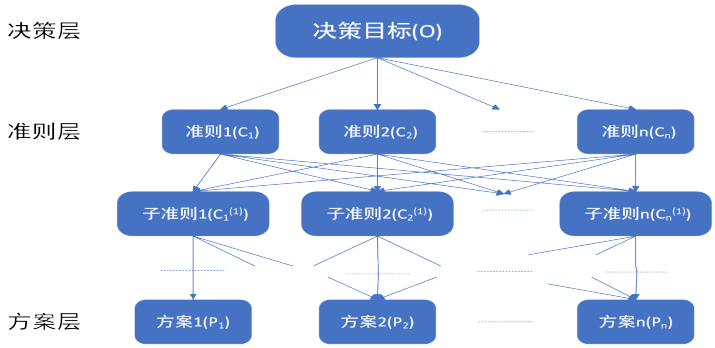

图 1: 层次分析法框架

准则层中准则因素之间相互独立。
我们选择的准则因素有：生活使用能耗、地区差异、周边产业、建造与拆除能耗、生产运输。

5.2.2 构建成对比较矩阵及归一化

1. 构建比较矩阵
   构造比较矩阵是通过比较同一层次上的各因素对上-层相关因素的影响作用. 而不是把所有因素放在一起比较，即将同一层的各因素进行两两对比。设某层有 n 个因素，$x = \{x_1, x_2 \dots x_n\}$ 要比较它们对上一层某一准则 (或目标) 的影响程度，确定在该层中相对于某一准则所占的比重。上述比较是两两因素之间进行的比较，比较时常取 1 9 尺度。

   用 a_{ij} 表示第 i 个因素相对于第 j 个因素的比较结果，则

$$a_{ij} = \frac{1}{a_{ji}}$$

尺度	含义
1	第 i 个因素与第 j 个因素影响相同
3	第 i 个因素与第 j 个因素影响稍强
5	第 i 个因素与第 j 个因素影响较强
7	第 i 个因素与第 j 个因素影响明显强
9	第 i 个因素与第 j 个因素影响极端强
2, 4, 6, 8	两相邻判断的中间值

$$A = (a_{ij})_{n\times n} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \tag{5.4}$$

A 则称为成对比较矩阵。

2. 归一化

对各城市的数据进行归一化处理

数据 \ 城市 / 指标	苏州	南京	南通	无锡	常州
直接	48	51.5	27	57.2	29
间接	126.3	134.3	118.7	97.8	71.6
运营	4.8	4.9	3.5	2.7	2.8
归一化比例	0.226	0.242	0.127	0.269	0.126
	0.230	0.245	0.216	0.178	0.130
	0.257	0.262	0.187	0.144	0.150

5.2.3 层次单排序及一致性检验

1. 层次单排序

和积法（算术平均法）：取判断矩阵 n 个列向量归一化后的算术平均

值，近似作为权重，即

$$W_i = \frac{1}{n}\sum_{j=1}^{n}\frac{a_{ij}}{\sum_{k=1}^{n}a_{kj}}(i=1,2,\cdots,n) \tag{5.5}$$

求根法（几何平均法）：将比较矩阵的各列（或行）向量求几何平均后归一化，可近似作权重，即

$$W_i = \sum_{j=1}^{n}\frac{(\prod_{j=1}^{n}a_{ij})^{\frac{1}{n}}}{\sum_{k=1}^{n}(\prod_{j=1}^{n})^{\frac{1}{n}}}(i=1,2,\cdots,n) \tag{5.6}$$

特征值法：求出矩阵的最大特征值 $\lambda_{\max}$ 以及其对应的特征向量 T，对求出的特征向量进行归一化即可得到我们的权重。

在我们的模型中综合采用了三种方法得到权重，降低了单一方法带来的不确定性，使数据结果更加可靠。

2. 一致性检验

通常情况下，由实际得到的判断矩阵不一定是一致的，即不一定满足传递性和一致性实际中，也不必要求一致性绝对成立，但要求大体上是一致的，即不一致的程度应在容许的范围内主要考查以下指标：

(a) 一致性指标 CI

$$CI = \frac{\lambda_{\max}-n}{n-1} \tag{5.7}$$

(b) 平均随机一致性指标 RI

为衡量 CI 的大小，引入随机一致性指标 RI：

$$RI = \frac{CI_1+CI_2+\cdots+CI_n}{n} \tag{5.8}$$

其中，随机一致性指标 RI 和判断矩阵的阶数有关，一般情况下，矩阵阶数越大，则出现一致性随机偏离的可能性也越大，对于阶数小于 9，其对应关系如图：

n	1	2	3	4	5	6	7	8	9
RI	0	0	0.58	0.90	1.12	1.24	1.32	1.41	1.45

(c) 检验系数 CR

考虑到一致性的偏离有可能是由于随机原因造成的，因此在检验判断矩阵是否具有满意的一致性时，还需将 CI 和 RI 进行比较，得出检验系数 CR，公式如下：

$$CR = \frac{CI}{RI} \tag{5.9}$$

一般地，如果 $CR \leq 0.1$，则认为该判断矩阵通过一致性检验，$A_{\max}$ 对应的特征向量 W 可以作为排序的权重向量，此时

$$\lambda_{\max} \approx \sum_{i=1}^{n} \frac{(A \cdot W)_i}{nw_i} = \frac{1}{n} \sum_{i=1}^{n} \frac{\sum_{j=1}^{n} a_{ij} w_j}{w_i} \tag{5.10}$$

否则就不具有满意一致性，需要调整对比较矩阵。

5.2.4 计算组合权重及得分

得到最大特征值对应的特征向量

$$T = [t_1 \quad t_2 \quad \cdots \quad t_n]$$

计算得到权重向量

$$W = [w_1 \quad w_2 \quad \cdots \quad w_n]$$
$$w_i = \frac{t_i}{\sum_{i=1}^{n} t_i}$$

随后分别通过公式 (5.2) 和公式 (5.3) 求得两个权重向量，最后将三个权重向量取算术平均作为最终的权重向量。设指标评分矩阵为 P，那么最后的得分矩阵 S 为：

$$S = P \cdot W$$

5.3 问题三的模型建立与求解

5.3.1 模型的建立

基于第二问所建模型，通过建筑生命周期的三个阶段确定代表性强的指标加入模型评价。我们选取的指标有：

建造阶段：建筑材料生产运输的碳排放系数、建造过程所产生的碳排放
运行阶段：生活使用能耗、气候原因
拆除阶段：拆除过程的能耗

5.3.2 模型的验证

为了验证模型的有效性，我们将 2021 年江苏省 13 个地级市的居住建筑碳排放作为输入输入到所建立模型中进行综合评价，并将得到的结果排名与实际的排名进行比较，结果如图：

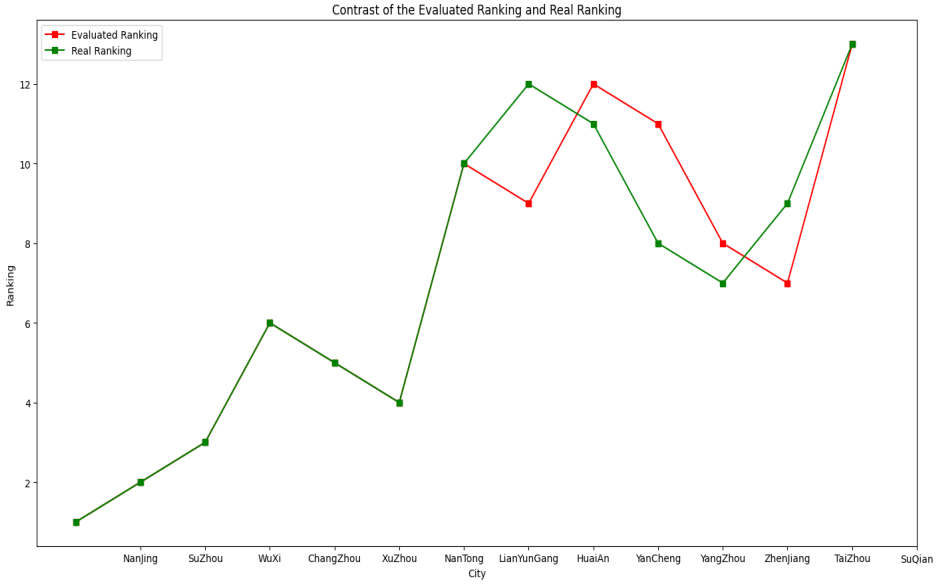

图 2: CONTRAST

通过比较来看，评价模型的评价结果与实际结果吻合得较好，很好的验证了所建立模型的有效性。

5.4 问题四的模型建立与求解

5.4.1 模型的分析

本题是分析预测 2023 年江苏省建筑全过程的碳排放量。由于要求基于江苏省建筑全过程碳排放的历史数据建立预测模型，用于分析江苏省未来建筑全过程的碳排放量，因此我们采用信息不完全、不充分的预测系统——灰色预测，建立灰色预测模型 GM (1,1) 模型，基于历史时期数据去预测未来时期数据。

5.4.2 模型的建立

1. 级比检验

级比检验用于判断数据序列进行模型构建的适用性, 如果通过了该检

验，则可以使用灰色预测。计算公式为：

$$\lambda(k) = \frac{x^{(0)}(k-1)}{x^{(0)}(k)}, k = 2, 3, \ldots, n \tag{5.11}$$

如果 $\lambda(k)$ 在区间

$$(e^{-\frac{2}{n+1}}, e^{\frac{2}{n+2}})$$

内，说明可用 GM (1,1) 模型，其中 $x^{(0)}(t)$ 是 年份-数据序列。

如果 $\lambda(k)$ 在该区间外，则可尝试通过平移变换，将每个数据都加上某一常数 c，再看 $\lambda(k)$ 是否在区间内，最后模型结果再减去 c 即可. 如果无论如何进行平移变换 $\lambda(k)$ 始终在该区间外，则说明该问题不适合用 GM (1, 1)。

而经过计算，我们的数据可以通过级比检验。

2. **构造累加生成序列**

由于历史数据量少且无明显规律，难以预测，我们使用累加生成法得到一个新的序列 $x^{(1)}(t)$，并尝试用数学方法拟合新序列进而预测。累加生成序列公式为：

$$x^{(1)}(k) = \sum_{i=1}^{k} x^{(0)}(i) \tag{5.12}$$

3. **构建一阶常微分方程**

通常情况下，累加生成序列的图像可用指数函数图像拟合，而一阶常微分方程的通解形式恰是指数函数，因此我们构建一阶常微分方程，通过求解这个方程得到预测函数。

该一阶常微分方程的形式为：

$$\frac{dx^{(1)}}{dt} + ax^{(1)} = u \tag{5.13}$$

其中 a, u 为发展系数和灰色作用量。

5.4.3 模型的求解

1. **方程式改写**

由于数据是离散而非连续的，首先将 $\frac{dx^{(1)}}{dt}$ 改写成 $\frac{\Delta x^{(1)}}{\Delta t}$。而由于 t 的单

位是年份，因此始终有 $\Delta t = (t+1)-t = 1$，因此我们得到新形式的方程

$$x^{(0)}(t) + ax^{(1)}(t) = u$$

移项得到：

$$x^{(0)}(t) = -ax^{(1)}(t) + u \tag{5.14}$$

得到该方程后，注意到可以通过最小二乘法求得最佳的参数 a 和 u。

2. **最小二乘法求解最佳参数**

方程 (5.14) 的矩阵形式为 $Y = BU$，具体形式为：

$$\begin{bmatrix} x^{(0)}(2) \\ x^{(0)}(3) \\ \vdots \\ x^{(0)}(N) \end{bmatrix} = \begin{bmatrix} -\frac{1}{2}[x^{(1)}(2) + x^{(1)}(1) & 1] \\ -\frac{1}{2}[x^{(1)}(3) + x^{(1)}(2) & 1] \\ \vdots & 1 \\ -\frac{1}{2}[x^{(1)}(N) + x^{(1)}(N-1) & 1] \end{bmatrix} \begin{bmatrix} a \\ u \end{bmatrix} \tag{5.15}$$

我们要最小二乘法要求解的目标就是 $(Y - BU)^T(Y - BU)$ 取最小时的 U。求解 U 的估计值的方法为

$$\hat{U} = [\hat{a}, \hat{u}] = (B^T B)^{-1} B^T Y$$

借此求得参数 $a,\ u$ 的最佳近似值 $\hat{a}, \hat{u}$ 后，即可求解微分方程得到拟合函数。

3. **求解微分方程**

通过方才求解到的 $\hat{a}, \hat{u}$，代入方程 (5.14)，求得方程的解：

$$\hat{x}^{(1)}(k+1) = (x^{(0)}(1) - \frac{\hat{u}}{\hat{a}})e^{-\hat{a}k} + \frac{\hat{u}}{\hat{a}}, \quad k = 0, 1, \cdots \tag{5.16}$$

当 k 小于已有数据量是，求得的为拟合值，大于时则是预测值。需要注意的是求得的为 $\hat{x}^{(1)}$ 是累加生成序列，真正要求的原始序列 $\hat{x}^{(0)}$ 需要再通过差分法求得，即：

$$\hat{x}^{(0)}(k) = \begin{cases} \hat{x}^{(1)}(1) & k = 1 \\ \hat{x}^{(1)}(k) - \hat{x}^{(1)}(k-1) & k > 1 \end{cases} \tag{5.17}$$

5.4.4 模型的检验

最后需要对我们的模型进行检验，判断拟合值是否能较好地逼近实际值。常用的检验方法有：级比偏差检验、残差检验、后验差比检验等。
我们选择的方法是后验差比检验，通过后验差比值 C 的大小来判断模型的预测精度，其计算公式为：

$$C = \frac{\sigma_1}{\sigma_2} \tag{5.18}$$

其中 σ_1, σ_2 分别表示历史数据方差和残差方差。
若 C 值小于 0.35，则可说明模型精度高；若小于 0.5，则说明模型精度合格；若小于 0.65，则说明模型精度基本合格；若大于 0.65，则说明模型精度不合格。

5.4.5 模型结果

模型拟合结果与预测和历史记录对比图如下：

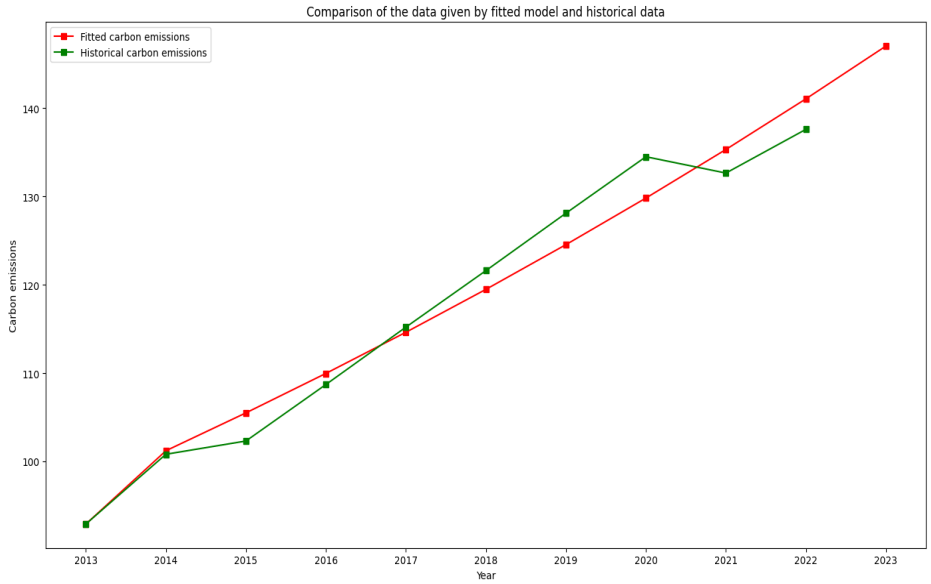

图 3: COMPARISON

6　结果检验与误差分析

6.1　问题三结果检验

«««< HEAD ======= 为了验证模型的有效性，我们将 2021 年江苏省 13 个地级市的居住建筑碳排放作为输入输入到所建立模型中进行综合评价，并将得到的结果排名与实际的排名进行比较，得到以下数据：

从数据中我们发现评价模型给出的排名大体上是符合实际的，但是针对部

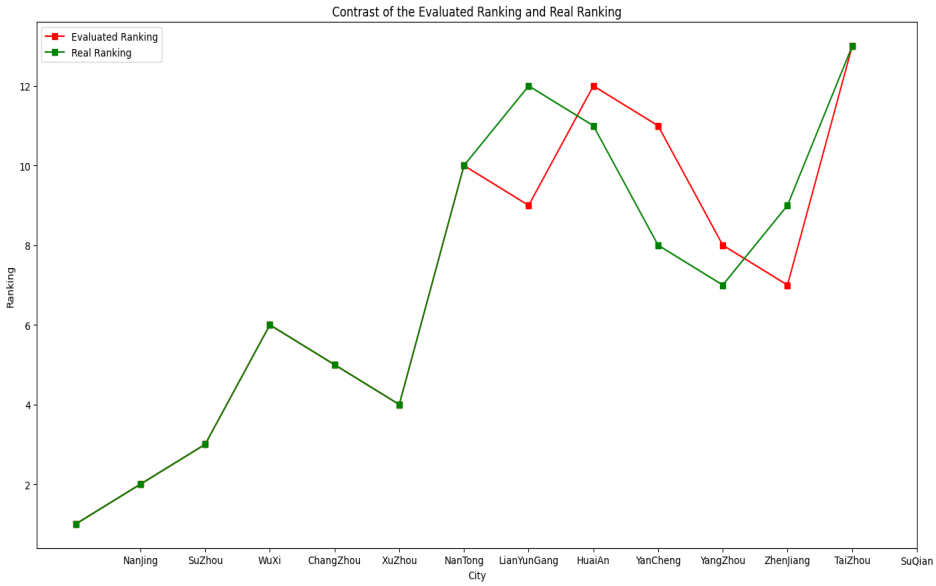

图 4: CONTRAST

偏差个数	最大偏差	平均偏差
5	3	0.77

分地区排名有较大差别，我们推测是由地区差异或周边产业导致的，而我们并未将其纳入指标范围内。

6.2 问题四结果检验

我们对模型的给出的拟合值进行了后验差比检验，得到的后验差比值 C 的值为 0.0020612375777761378，远小于 0.35，说明我们的模型拟合精度很高。»»»> 8d617c6f1c3184dc6152163c45efed9ce3dcf82a

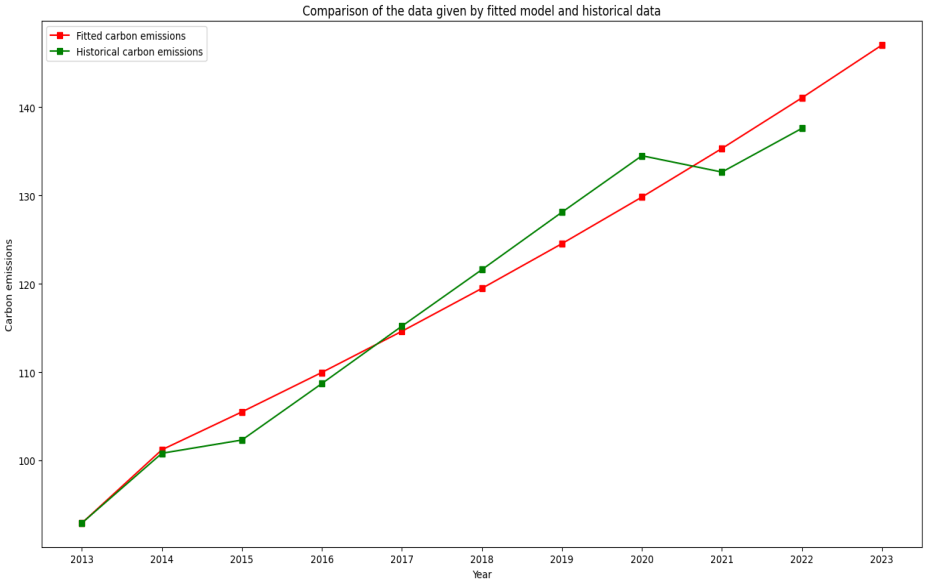

图 5: COMPARISON

7 模型评价

7.1 模型优点

7.1.1 建筑生命周期碳排放评价模型

使用层次分析法建立模型，定性与定量相结合，将对象视作系统进行了综合决策，能够考虑到诸多个因素的共同作用，有助于对建筑生命周期碳排放评价有更深入的了解。

7.1.2 江苏省建筑碳排放全过程短期预测模型

使用灰色预测模型 $GM(1,1)$，所需数据量少且运算方便，建模精度高，十分适合基于历史数据预测下一步的发展。

7.2 模型缺点

7.2.1 建筑生命周期碳排放评价模型

只能从原有的方案中选择，不能产生新的方案。

模型建立较为粗略，不适用于精度较高的问题。

判断矩阵的主观性太大，难以使人信服，除非采用公认的专家学者的判断。

7.2.2 江苏省建筑碳排放全过程短期预测模型

只能适用于样本较少的短期预测，如果样本过大或需要预测时间过长，都会产生较大误差，使预测目标失效。

8 模型推广与改进

8.1 评价模型的改进

8.1.1 判断矩阵的改进

原先的判断矩阵主观意志较强，不足以令人信服，可查阅更多相关文献，得到权威专家学者的判断来构成判断矩阵。

8.1.2 判断指标的优化

原先的判断指标考虑地不够充分，导致模型在部分城市上精度只差强人意，应选取更多具有代表性的指标加入到模型的准则层中。

8.2 预测模型的改进

8.2.1 方程式修正

考虑到原方程中含有 $\frac{\Delta x^{(1)}}{\Delta t}$，因此将 $x^{(1)}(t)$ 修正为均值生成序列 $z^{(1)}(t)$，计算公式为

$$z^{(1)}(t) = \frac{x^{(1)}(t)}{2} + \frac{x^{(1)}(t-1)}{2} + \cdots, \quad t = 2, \ldots, n$$

即方程改写为

$$x^{(0)}(t) = -az^{(1)}(t) + u$$

8.2.2 检验改进

为进一步检验模型精度，最后的模型检验可使用多重方法检验模型精度。

9 参考文献

[1] 胃口很大的一条小蛇仔，数学建模之综合评价模型（层次分析法+Topsis 法 + 熵权法），2023-5-9

[2] 果州做题家，【建模算法】层次分析法（Python 实现），2023-5-9

[3] 数学建模 BOOM ，层次分析法—评价类问题（北海数学建模：数模零基础入门国赛美赛必看），2023-5-10

[4] 汪江波，江苏省建筑业低碳发展的治理机制研究；2019(6);29-45.

[5] 李爽, 陶东, 夏青. 基于扩展 STIRPAT 模型的我国建筑业碳排放影响因素研究 [J]. 管理现代化, 2017 (3): 96-98.

[6] Lu Y, Cui P, Li D. Which activities contribute most to building energy consumption in China? A hybrid LMDI decomposition analysis from year 2007 to 2015[J]. Energy and Buildings, 2018, 165: 259-269.

[7] Ma M, Cai W. Do commercial building sector-derived carbon emissions decouple from the economic growth in tertiary industry? A case study of four municipalities in China[J]. Science of The Total Environment, 2019, 650: 822-834.

[8] code_king1, 数学建模——灰色预测模型 Python 代码,2023-5-12

[9] 欧阳罢笔，灰色预测模型，2023-5-12

[10] 邓聚龙. 灰色预测与灰决策 [M]. 武汉: 华中科技大学出版社,2002.

[11] 基于灰色关联度的组合预测模型的性质 [J]. 陈华友, 赵佳宝, 刘春林. 东南大学学报 (自然科学版),2004(01)

[12] 中共江苏省委办公厅,2023-5-13

[13] 黄向岚, 张训常, 刘晔. 我国碳交易政策实现环境红利了吗?[J]. 经济评论, 2018 (6): 86-99.

[14] 单彩杰, 冯大阔. 《建筑业 10 项新技术 (2017 版)》绿色施工技术综述 [J]. 建筑技术, 2018, (3): 270-275

[15] 郭广翠. 中国建筑节能政策对降低建筑能耗的影响研究 [D]. 北京: 北京交通大学, 2018.

[16] 欧晓星. 低碳建筑设计评估与优化研究 [D]: 南京: 东南大学, 2016.

[17] 荆克迪. 中国碳交易市场的机制设计与国际比较研究 [D]: 天津: 南开大学, 2014

10 附录

附录 A 问题一

Listing 1: **Question1**

```
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):
```

```
37      return (k * a * t) / d
38
39
40  def get_day_qmake(dt):
41      if dt == 0:
42          return 0.0
43
44      day_qmake = \
45          get_qmake(wall_thermal, wall_square, dt, wall_thickness) + \
46          get_qmake(windowAndDoor_thermal, windowAndDoor_square, dt,
                windowAndDoor_thickness) + \
47          get_qmake(roof_thermal, roof_square, dt, roof_thickness) + \
48          get_qmake(ground_thermal, ground_square, dt, ground_thickness)
49      return day_qmake * 86400
50
51
52  def get_month_qmake(month):
53      return get_day_qmake( abs(get_delta_t(temperatures[month]))) * monthdays[month]
54
55
56  def get_qelec(qmake, dt):
57      if dt < 0:
58          return qmake / EER
59      elif dt == 0:
60          return 0
61      else:
62          return qmake / COP
63
64
65  def get_carbon_emission(qelec):
66      return (qelec / 3600000) * 0.28
67
68
69  carbonEmissions = []
70  for i in  range(0, 12):
71      qelec = get_qelec(get_month_qmake(i), get_delta_t(temperatures[i]))
72      carbonEmissions.append(get_carbon_emission(qelec))
73
74  print("Sum of carbon emissions: " +  str( sum(carbonEmissions)) + "kg")
75
76  plt.figure(figsize=(10, 6))
77  plt.bar( range(1,  len(carbonEmissions) + 1), carbonEmissions, fc='g')
```

```
78 plt.title("Carbon Emissions")
79 plt.xlabel("Month")
80 plt.ylabel("kg")
81
82 plt.show()
```

附录 B　　问题二

Listing 2: **Question2**

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4
5  cities = ["SuZhou", "NanJing", "NanTong", "WuXi", "ChangZhou"]
6
7  discriminantMatrix = np.array([[1, 1/5, 1/5], [5, 1, 1/2], [5, 2, 1]])
8  eigens = np.linalg.eig(discriminantMatrix)
9  maxEigenvalue = np. max(eigens[0])
10 rowAndColumn = np.argwhere(eigens[0] == maxEigenvalue)
11 maxEigenvector = eigens[1][::-1, rowAndColumn[0]]
12
13 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]
14 dim = discriminantMatrix.shape[0]
15 RI = RI_LIST[dim]
16 CI = (maxEigenvalue - dim) / (dim - 1)
17 CR = CI / RI
18 print("CR: " +  str(CR))
19 print("Passed consistency test") if CR < 0.1 else print("Failed to passed consistency
       test")
20
21 Scores = np.array([[48.0, 126.3, 4.8],
22                    [51.5, 134.3, 4.9],
23                    [27, 118.7, 3.5],
24                    [57.2, 97.8, 2.7],
25                    [29, 71.6, 2.8]])
26
27 sums = np. sum(Scores, axis=0)
28 weightedScores = Scores / sums.reshape(1, -1)
29
```

```
# Eigenvalue average
eigenvectorWeight = maxEigenvector /  sum(maxEigenvector)

# Arithmatic average
arithmaticSums = np. sum(discriminantMatrix, axis=0)
normalizedMatrix = discriminantMatrix / arithmaticSums.reshape(1, -1)
arithmaticWeight = np. sum(normalizedMatrix, axis=1)
arithmaticWeight /= dim
arithmaticWeight = arithmaticWeight[-1::-1]

# Geometric average
prodVector = np.prod(discriminantMatrix, axis=1)
prodVector = np.power(prodVector, 1/dim)
prodSums = np. sum(prodVector, axis=0)
geometricWeight = prodVector / prodSums
geometricWeight = geometricWeight[-1::-1]

weight = (arithmaticWeight + geometricWeight) / 2

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

plt.figure(figsize=(10, 6))
x = [1, 2, 3, 4, 5]
x_label = cities
plt.bar(x, finalScores, 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

Listing 3: **Question3**

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

```

```
4
5  cities = ["NanJing", "SuZhou", "WuXi", "ChangZhou", "XuZhou", "NanTong", "LianYunGang"
       ,
6            "HuaiAn", "YanCheng", "YangZhou", "ZhenJiang", "TaiZhou", "SuQian"]
7
8  discriminantMatrix = np.array([[1, 1/5, 1/5], [5, 1, 1/2], [5, 2, 1]])
9  eigens = np.linalg.eig(discriminantMatrix)
10 maxEigenvalue = np. max(eigens[0])
11 rowAndColumn = np.argwhere(eigens[0] == maxEigenvalue)
12 maxEigenvector = eigens[1][::-1, rowAndColumn[0]]
13
14 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]
15 dim = discriminantMatrix.shape[0]
16 RI = RI_LIST[dim]
17 CI = (maxEigenvalue - dim) / (dim - 1)
18 CR = CI / RI
19 print("CR: " +  str(CR))
20 print("Passed consistency test") if CR < 0.1 else print("Failed to passed consistency
       test")
21
22 Scores = np.array([[51.5, 134.3, 4.9],
23                    [48.6, 126.3, 4.8],
24                    [57.2, 97.8, 2.7],
25                    [29, 71.6, 2.8],
26                    [30.1, 79.6, 2.91],
27                    [27, 118.7, 3.5],
28                    [12.4, 36.4, 1.32],
29                    [13.5, 32.5, 1.34],
30                    [16.7, 45.2, 1.72],
31                    [20.4, 52.3, 2.01],
32                    [14.01, 36.1, 1.401],
33                    [14.2, 37.3, 1.412],
34                    [9.27, 25.3, 0.965]])
35
36 sums = np. sum(Scores, axis=0)
37 weightedScores = Scores / sums.reshape(1, -1)
38
39 # Eigenvalue average
40 eigenvectorWeight = maxEigenvector /  sum(maxEigenvector)
41
42 # Arithmatic average
```

```
arithmaticSums = np. sum(discriminantMatrix, axis=0)
normalizedMatrix = discriminantMatrix / arithmaticSums.reshape(1, -1)
arithmaticWeight = np. sum(normalizedMatrix, axis=1)
arithmaticWeight /= dim
arithmaticWeight = arithmaticWeight[-1::-1]

# Geometric average
prodVector = np.prod(discriminantMatrix, axis=1)
prodVector = np.power(prodVector, 1/dim)
prodSums = np. sum(prodVector, axis=0)
geometricWeight = prodVector / prodSums
geometricWeight = geometricWeight[-1::-1]

weight = (arithmaticWeight + geometricWeight) / 2

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

sortedScores =  sorted(
    enumerate(finalScores), key=lambda finalScores:finalScores[1], reverse=True)
finalRanking = [finalScores[0] + 1 for finalScores in sortedScores]
realRanking = [1, 2, 3, 6, 5, 4, 10, 12, 11, 8, 7, 9, 13]

plt.figure(figsize=(20, 12))
x =  range(1, 14)
x_label = cities

plt.plot(x_label, finalRanking, 's-', color='r', label="Evaluated Ranking")
plt.plot(x_label, realRanking, 's-', color='g', label="Real Ranking")

plt.title("Contrast of the Evaluated Ranking and Real Ranking")
plt.xticks(x, x_label)
plt.xlabel("City")
plt.ylabel("Ranking")
plt.legend(loc="best")

plt.show()
```

附录 D　　问题四

Listing 4: **Question4**

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

historicalData = [92.9, 100.8, 102.3, 108.7, 115.2, 121.6, 128.1, 134.5, 132.65,
    137.6]
years =  len(historicalData)
Data0 = np.array(historicalData)

e = math.e


# 级比检验
def comparison_test():
    left_border = math.exp( float(-2) / (years + 1))
    right_border = math.exp( float(2) / (years + 2))
    for k in  range(2, years):
        test_value =  float(Data0[k - 1]) / Data0[k]
        if test_value < left_border or test_value > right_border:
            return False
    return True


def get_prefix_sum_sequence():
    history_data_prefix_sum = [ sum(historicalData[0: i + 1]) for i in  range(years)]
    return np.array(history_data_prefix_sum)


def get_y_and_b(data1):
    y = np.zeros([years - 1, 1])
    b = np.zeros([years - 1, 2])
    for i in  range(0, years - 1):
        y[i][0] = historicalData[i + 1]
        b[i][0] = -0.5 * (data1[i] + data1[i + 1])
        b[i][1] = 1
    return y, b


def calc_a_and_u(y, b):
    _u = np.linalg.inv(b.T.dot(b)).dot(b.T).dot(y)
    return _u[0][0], _u[1][0]
```

```
41
42
43  def get_fitted_data1(_a, _u):
44      fitted_data1 = np.zeros(years + 1)
45      fitted_data1[0] = Data0[0]
46      for i in  range(1, years + 1):
47          fitted_data1[i] = (Data0[0] - _u / _a) * math.exp(-_a * i) + _u / _a
48      return fitted_data1
49
50
51  def get_diff_ave(fitted_data1):
52      diff_ave = 0.0
53      for i in  range(0, years):
54          diff_ave += (Data0[i] - fitted_data1[i])
55      return diff_ave / years
56
57
58  def get_diff_variance(fitted_data1):
59      diff_ave = get_diff_ave(fitted_data1)
60      _diff_var = 0.0
61      for i in  range(0, years):
62          _diff_var += (Data0[i] - fitted_data1[i] - diff_ave) ** 2
63      return _diff_var / years
64
65
66  def get_histo_ave():
67      return  sum(Data0) / years
68
69
70  def get_histo_variance():
71      histo_ave = get_histo_ave()
72      _histo_var = 0.0
73      for i in  range(0, years):
74          _histo_var += (Data0[i] - histo_ave) ** 2
75      return _histo_var / years
76
77
78  def get_fitted_data0(fitted_data1):
79      fitted_data0 = np.zeros(years + 1)
80      fitted_data0[0] = fitted_data1[0]
81      for i in  range(1, years + 1):
82          fitted_data0[i] = fitted_data1[i] - fitted_data1[i - 1]
```

```
    return fitted_data0


def test_model_accuracy(_histo_var, _diff_var):
    _c = _histo_var / _diff_var
    print("C " +  str(_c))
    if _c < 0.35:
        print("High model accuracy!")
    elif _c < 0.5:
        print("Qualified model accuracy!")
    elif _c < 0.65:
        print("Basically qualified model accuracy!")
    else:
        print("Unqualified model accuracy!")
        return False
    return True


if __name__ == '__main__':
    if not comparison_test():
        print("Failed to pass comparison test!")
        exit(0)

    print("Passed comparison test!")

    Data1 = get_prefix_sum_sequence()
    Y, B = get_y_and_b(Data1)
    a, u = calc_a_and_u(Y, B)
    fittedData1 = get_fitted_data1(a, u)
    histo_var = get_histo_variance()
    diff_var = get_diff_variance(fittedData1)
    fittedData0 = get_fitted_data0(fittedData1)

    if not test_model_accuracy(histo_var, diff_var):
        print("Using Grey Prediction model is not suitable!")
        exit(0)

    print("Predict: " +  str(fittedData0[-1]))

    plt.figure(figsize=(20, 12))
    x0_label =  range(2013, 2024)
    x1_label =  range(2013, 2023)
```

```
    plt.plot(x0_label, fittedData0, 's-', color='r', label="Fitted carbon emissions")
    plt.plot(x1_label, Data0, 's-', color='g', label="Historical carbon emissions")

    plt.title("Comparison of the data given by fitted model and historical data")
    plt.xticks(x0_label)
    plt.xlabel("Year")
    plt.ylabel("Carbon emissions")
    plt.legend(loc="best")

    plt.show()
```