关于提高医院整体水平的研究的统计分析及建模

摘要

随着我国的经济快速发展，人民对美好生活的需求日益增长。其中，医疗服务也是影响人民生活的幸福程度的关键因素之一。本文基于题目所给信息和要求，从多方面、多层次、多角度，并结合近期人口老年化趋势和民营、公立医院间的关系对不同的问题建立数学模型进行分析，从而提高医院的整体水平，以此满足人民的需求。

对于问题一，第一问首先通过国家统计局收集并整理出所需数据，然后采用 ARIMA 模型对近 25 年的人口年龄数据进行拟合，预测出未来老年化的人口走向，结果显示，随着年份增长，高龄人口越来越多，老龄化越来越严重。第二问首先通过国家统计局收集并整理所需数据，然后采用多项式回归模型，根据已有的居民收入、人口年龄结构三个指标进行拟合，预测了未来几十年间的居民医疗需求，结果显示，未来的医疗支出将持续增长。

对于问题二，通过广东卫生年鉴收集并整理出所需数据，然后采用单指数平滑法进行拟合，预测出未来常见疾病发生率走向，结果显示些预测数据与常识性预测相匹配，并为医院提供了对应疾病的相关建议，即根据趋势做好医院资源规划。做好医院人力分配、做好宣传疾病预防。

对于问题三，首先根据题目将患者进行分类并建立一个基本的医院模型，然后使用 M/M/c 模型来表示多个检查点的服务系统，以最小化平均等待时间为目标并求解模型，最后得到了一种通用的排队理论及其相关的最优排队方法，即时间紧急程度大小为第一排队依据。

对于问题四，首先构建一个博弈模型，通过假设变量，给出民营医院与公立医院合作与不合作之间的四种竞争情况的公式，由于医院采取合作策略的概率会根据时间的推进而发生改变，然后，我们以这个为变化量，求出医院采取合作策略的概率图像，显示医院合作的概率为 1，即民营医院与公立医院必然会合作，最后根据以上信息，提出了多家医院之间的最佳合作与竞争策略，即提升自身医院各方面水平、合作互相取长补短、在市场有竞争优势。

关键词：预测 老龄化 医疗需求 病发率 排队理论 博弈论

一. 问题重述

1.1 背景介绍

习近平总书记曾在党的二十大报告中提出，推进健康中国建设，保障人民健康放在优先发展的战略位置，完善人民健康促进政策。为此，各地各医院需要不断坚持以人民为中心的发展思想，解决好人民群众看病就医难题，积极推动从以治病为中心向以人民健康为中心转变。同时，只有提高医院的整体水平、完善人性化服务，才能为人民看小病治大病，为社会做好关键传染病预防和对应措施，不断满足人民的需求。因此，提升医院的整体水平至关重要。除此之外，医院还需要根据社会需求对不同的疾病分配好医疗资源和人力，所以了解社会的实际情况，根据已有数据总结过去常见的疾病，预测未来的老龄化趋势和居民的医疗需求，能大大提升医院的服务水平。为了从多方面提升医院的人性化服务，满足不同年龄不同需求的患者的需求，医院还需要坚持深化改革，努力开创医疗保障事业新局面，深入推进门诊保障制度改革，不断优化服务系统，努力让群众“少跑腿”。

而根据不同的群众和社会的需求，民营医院也早早地随之产生。民营医院以独特的机制和创新能力，致力于为不同的消费人群提供多样化和高质量的医疗服务。同时，它也成为了公立医院的竞争对手和可合作对象。只有二者在竞争中合作，在合作中竞争，利用各自的优势，才能满足群众和社会的需求。

1.2 问题任务

问题一：基于居民收入水平、人口年龄结构和经济发展状况等数据，合理预测中国的老龄化趋势和居民对医疗服务的需求

问题二：分析某省份地区未来最常见的疾病，并为大型公立医院的总体发展提出建议。

问题三：不同类型的患者需要在医院进行不同的检查，而这些检查项目可能分布在不同地点，且各自的排队人数可能差异很大。请提出一种通用的排队理论和最优的排队方法，解决此类问题。

问题四：在民营医院与公立医院复杂的合作与竞争关系中，制定出多家医院之间的最佳合作与竞争策略。

二. 问题分析

2.1 问题一的分析

针对问题一，首先通过国家统计局，查找出三个所需指标的具体数据，分别是居民收入、人口年龄结构、经济发展水平，我们用 GDP 和人均 GDP 数据反应经济发展水平，并通过绘制条形图将近 25 年的数据详细表现出来。然后，我们假设 65 岁及以上的人口为老龄化，以近 25 年的人口年龄数据为指标，采用 ARIMA 模型进行拟合，预测出未来老年化的人口走向和大概数据。最后，采用多项式回归模型，由于 GDP 与个人收入之间存在秩亏，舍弃了根据已有的经济发展水平数据，用居民收入、人口年龄结构二个指标进行拟合，并用此模型去预测未来几十年间的居民医疗需求。

2.2 问题二的分析

针对问题二，具体思路和问题一比较类似。首先通过广东卫生年鉴查找出近年来最常见的疾病指标的具体数据，并通过绘制图表将近年的数据详细表现出来。然后，以这些数据为指标，采用单指数平滑法进行拟合，并使用一些误差概念用以评估模型的准确性保证模型的质量，预测出未来常见疾病发生率走

向和大概数据。最后，将这些数据绘制为图表，以此分析该省未来最常见的疾病，并为该省大型公立医院的总体发展提出建议。

2.3 问题三的分析

针对问题三，首先我们根据题意构建出医院的背景模型，由于疾病繁多，需要的检查项目大多各不相同，我们假设所有的疾病所需的检查疾病点分为 2 个，即 CT 和 X 光并根据实际情况将患者进行分类，分为急诊患者、紧急患者和普通门诊患者，以此给出排队问题的具体背景信息，然后根据背景信息我们使用 M/M/c 模型来表示多个检查点的服务系统，以最小化平均等待时间为求解排队方案的目标，并给出了平均队列长度、平均等待时间、平均系统时间等量的公式以衡量具体方案的优劣，最后通过对比两种方案，即以时间紧急程度大小为第一排队和以优先级为第一排队依据，并绘制出相关图表，得到了一种通用的排队理论及其相关的最优排队方法。

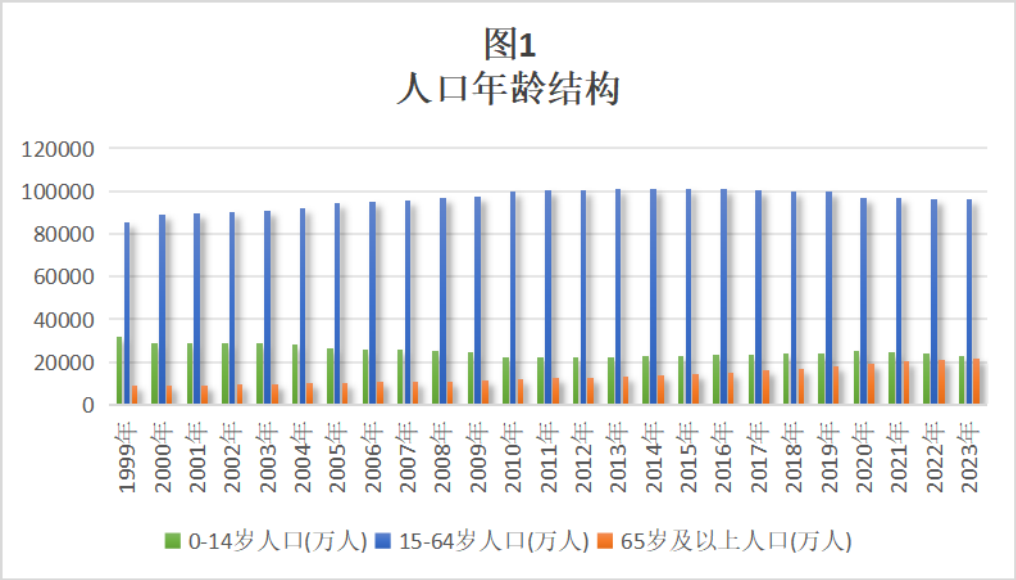
2.4 问题四的分析

针对问题四，我们使用博弈论来分析民营医院和公立医院之间的互动关系。构建一个博弈模型，研究四种情况下的收益，即民营医院和公立医院都合作，民营医院合作而公立医院不合作，民营医院不合作而公立医院合作和民营医院与公立医院都不合作。通过对比这四种情况下的收益，并以根据时间推进的合作概率来刻画合作的可取性，并结合公立医院与民营医院的特点，为改善多家医院间的合作与竞争策略提供借鉴与参考。

三. 数据预处理

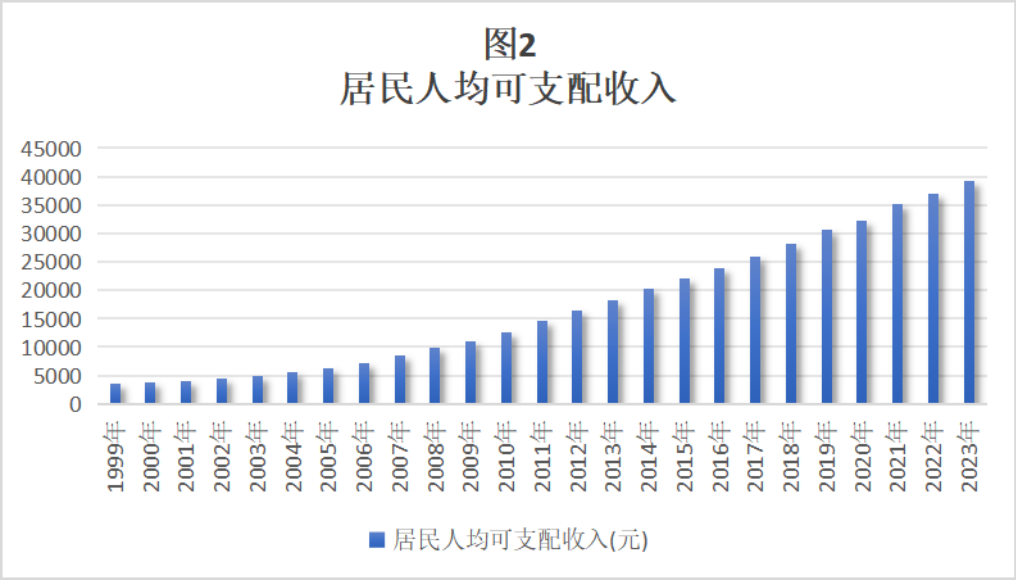
3.1 问题一的数据整合与分析

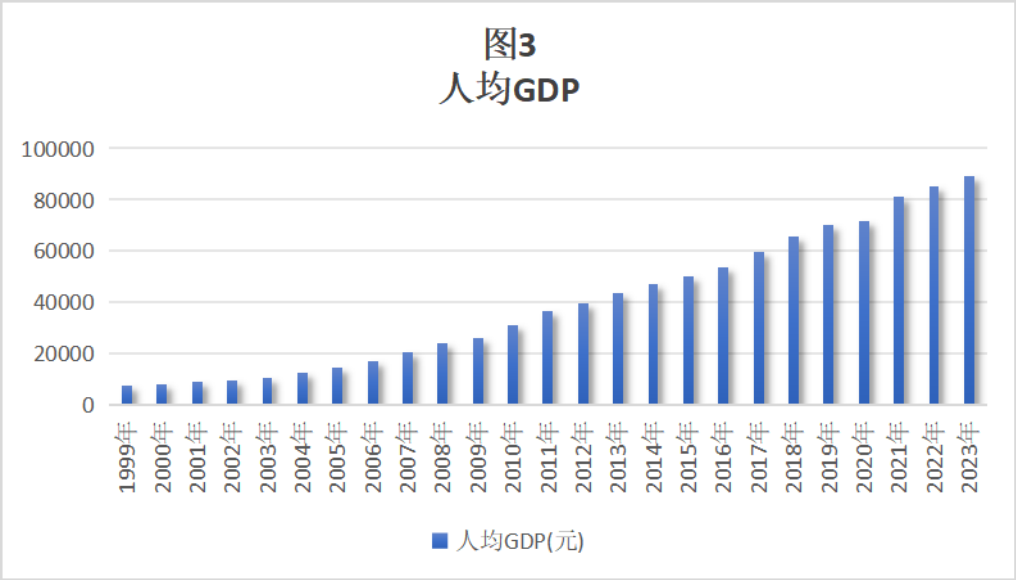
由国家统计局查找出近 20 年间的人口年龄结构，并以国家退休年龄为标准，将年龄分为三个层次，分别是 0 至 14 岁、15 至 64 岁、65 岁以上。具体数据如图 1 所示。

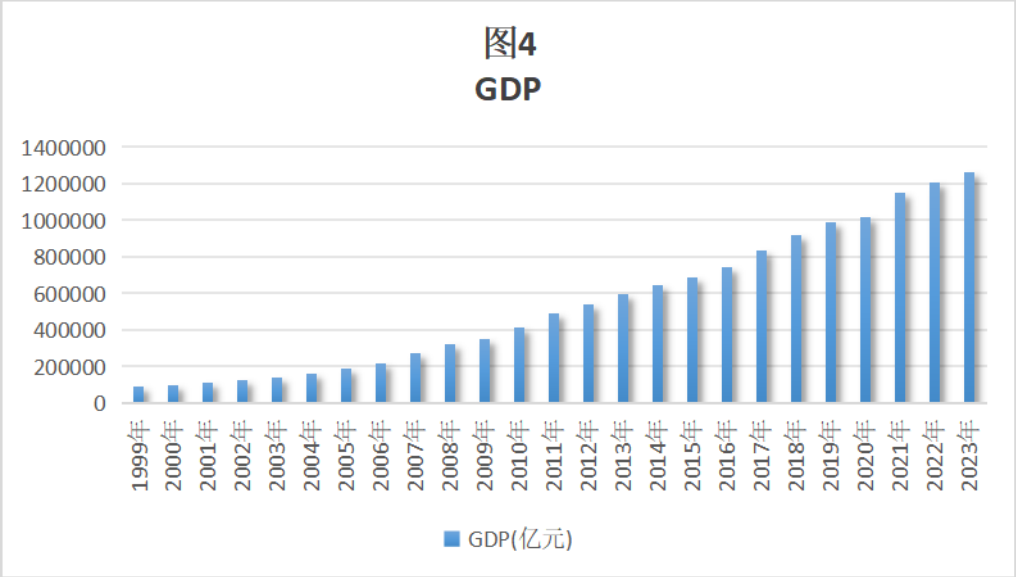


由图 1 可知，65 岁及以上人口呈现出逐年递增的趋势，意味着中国的老龄化程度正在持续加深。

同理，也查找出了近 20 年间的居民可支配收入、人均 GDP 和 GDP，后两者用于反应中国的经济发展水平。具体数据分别如图 2、图 3 和图 4 所示。







由图 2 可知，居民人均可支配收入呈现出逐年递增的趋势，这意味着居民的购买力和生活水平不断增加，居民会更愿意在各个方面消费更多的钱。

由图 3 和图 4 可知，GDP 和人均 GDP 也呈现出逐年递增的趋势，这意味着我国的经济发展呈现出稳中向好、长期向好的态势。

3.2 问题二的数据整合与分析

由广东省卫生统计年鉴查找出近年间的疾病发生率，为了贴合题目背景和符合生活信息，我们找出病发率前五的五大类疾病，分别是循环系统疾病、呼吸系统疾病、消化系统疾病、妊娠、分娩和产脊期类的疾病和其他接受的医疗服务类疾病，具体数据如表 1 所示。

表1 2012年到2023年广东省病发率前五疾病占比

指标	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023
循环系统疾病	11.3	12.1	12.1	11.1	11.3	11.5	13.5	12.1	13.5	13.4	13.2	12.4
呼吸系统疾病	14.8	13.1	12.7	12.8	12.6	12	13.2	13.6	10.2	10.7	12	14.1
消化系统疾病	8.8	9.5	9.3	8.6	7.8	7.8	7.7	7.6	7.8	8.1	8	7.9
妊娠、分娩和产脊期	15.7	13.5	14.1	16	17.1	16.6	14.2	13.7	13.2	11.1	10.5	9.2
其他接受医疗服务	6.5	7.7	7.6	7.8	6.8	7.1	6.8	8.2	9.1	9.9	10.5	10.3

注:数据来源于全省住院病人病案首页。

由表 1 可知，这发生率前五疾病中，循环系统疾病发生率整体呈现出一定的波动性，但在近几年有所上升；呼吸系统疾病发生率整体呈现出一定的波动性，但在最近两年有所上升；消化系统疾病发生率总体保持稳定，近几年略有下降；妊娠、分娩和产脊期整体呈现出一定的波动性，但在近几年下降的幅度比较大；其他接受医疗服务整体呈现出一定的波动性，但在近几年上升的幅度比较大。

四. 模型假设

1 假设未来年份内未发生任何紧急或不可预见的情况，以确保预测结果的稳定性和可靠性
2 假设所有医院的外部环境，如政策、经济形势和人口结构，在预测期内保持相对稳定，不会对模型结果产生显著干扰。
3 假设已收集数据是完全可获得且准确的，数据分析基于现有可用信息，无重大数据遗漏或错误。
4 假设所有医院在合作与竞争中的行为均遵循理性决策原则，且各方均以最大化自身利益为目标。

五. 模型的建立与求解

5.1 问题一的求解

问题一要求对中国的老龄化趋势和居民的医疗需求进行合理预测，预测老龄化趋势是根据以往数据直接进行拟合，因此采用高灵活性和贴切数据拟合的 ARIMA 模型。居民的医疗需求是在以往数据基础上根据居民收入、人口年龄结构和经济发展水平这几个自变量进行拟合，因此采用多项式回归模型建立函数

模型。

5.1.1 问题一模型介绍与设定

ARIMA 模型全称为自回归移动平均模型，是研究时间序列的重要方法。其在时序数据分析过程中既考虑了数据在时间序列上的依存性， 又考虑了随机波动的干扰性， 对对应时序发生得的变量运行短期趋势的预测准确率较高， 是近年应用比较广泛的方法之一。其拟合的一般公式如下所示。

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \cdots + \beta_p Y_{t-p} + \epsilon_t + \alpha_1 \epsilon_{t-1} + \alpha_2 \epsilon_{t-2} + \cdots + \alpha_q \epsilon_{t-q}$$

ARIMA(p，d，q)中，AR 是“自回归”，p 为自回归项数；MA 为滑动平均，q 为滑动平均项数，d 为使之成为平稳序列所做差分次数（阶数）。其流程图如下图所示。

图5 ARIMA模型建模步骤

多项式回归模型是一种通过将数据拟合到多项式函数来建立数学模型的方法，可以用于分析实验或观测数据中的关系，并用多项式函数来逼近数据。其拟合的一般公式如下所示。

$$y = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

5.1.2 问题一模型求解与结论

对于求解 ARIMA 模型，因为原老年化数据的非平稳性，首先需要对数据进行差分处理，以此消除趋势并实现平稳化。经过多次差分后，使用 ADF 检验确认数据的平稳性，最终确定使用二阶差分。然后，通过绘制差分后的数据的自相关函数（ACF）和偏自相关函数（PACF）图，结合 AIC（赤池信息准则）和 BIC（贝叶斯信息准则）选择模型的最佳阶数。根据模型的 AIC 和 BIC 值，我

们选择了最优的 AR（自回归）和 MA（移动平均）阶数。

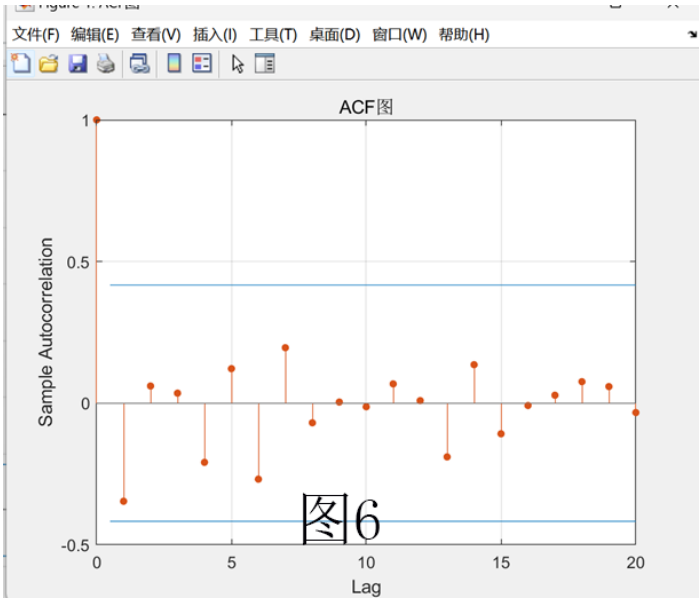


图6

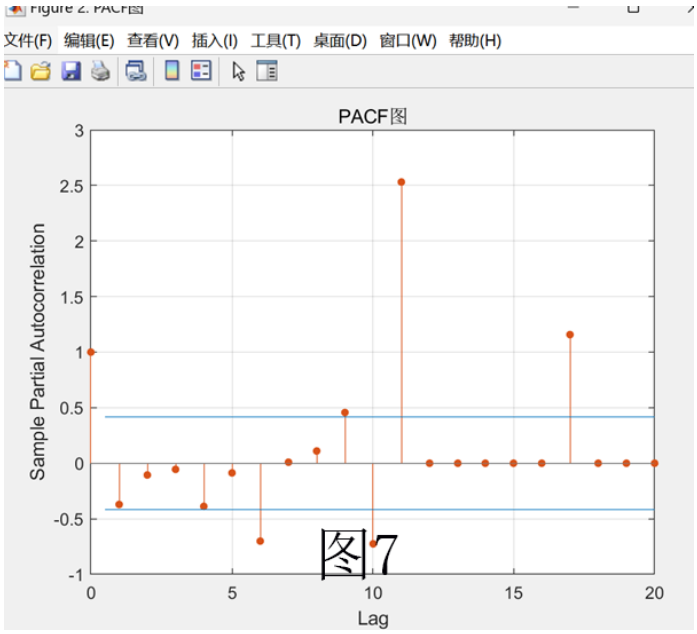
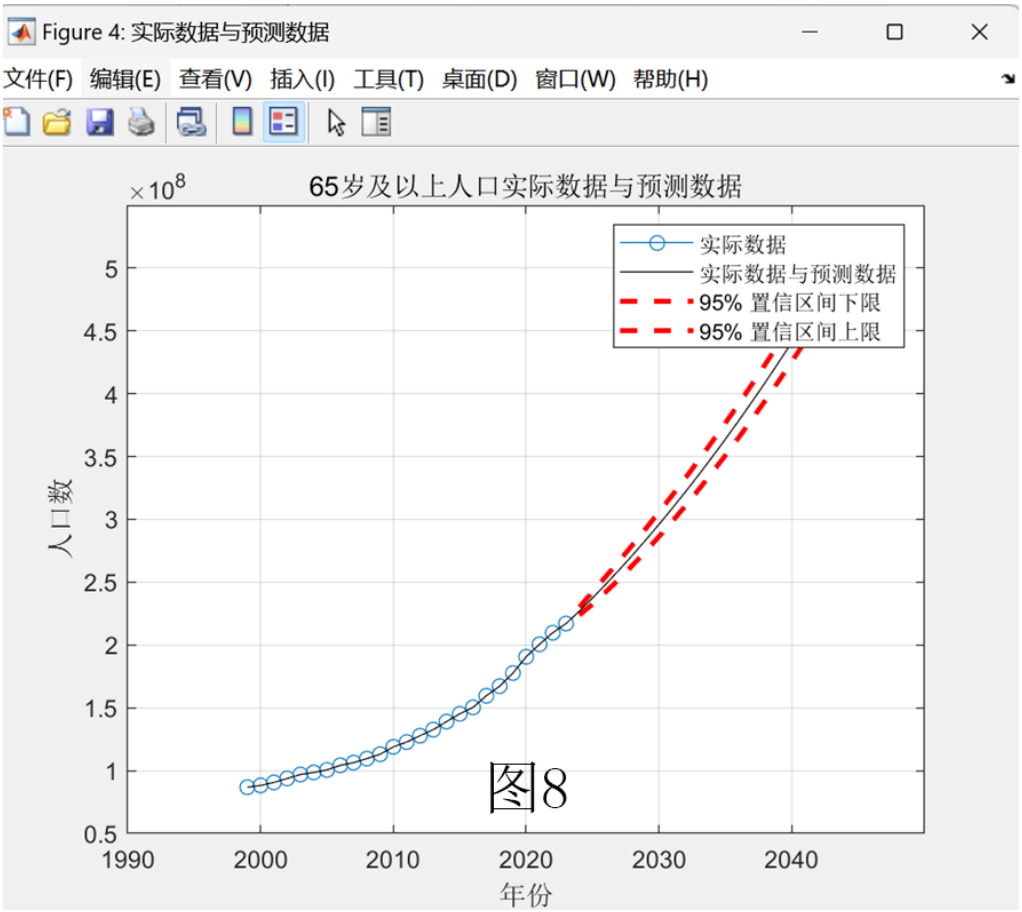


图7

接着，使用选择的阶数建立 ARIMA 模型，并对模型进行拟合（见下图）。通过对模型残差的分析，使用 Durbin-Watson 统计量检查残差的自相关性，以及标准化残差的 ACF 和 PACF 图验证模型的适用性。在拟合模型后，使用该模型对未来 20 年的 65 岁及以上人口进行预测。结果在 95%置信区间的上下限的基础上显示出老龄化趋势的持续增长和相关数据、如下图所示。



结果显示，随着年份增长，高龄人口越来越多，老龄化越来越严重。

在求解多项式回归模型时，我们首先从现有数据集中提取所需的变量，包括收入、不同年龄段（0-14 岁、15-64 岁、65 岁及以上）的比例以及医疗支出等，并对这些变量进行标准化处理，以消除不同量纲之间的影响。由于发现 GDP 与个人收入之间存在秩亏，因此需要检查数据中的多重共线性问题。为了解决此问题并确保模型的有效性，我们决定舍弃 GDP 数据，改用同时可以反映经济发展水平的居民收入来拟合模型。在模型拟合过程中，我们根据初步分析选择了合适的多项式阶数，以捕捉变量间的非线性关系。此外，为了校准数据，评估模型的性能，还计算训练数据上的预测误差，包括均方误差（MSE）和均方根误差（RMSE）。最后基于所拟合的模型，预测了 2024 年至 2043 年的医疗支出。结果显示，随着人口老龄化和收入水平的提高，未来的医疗支出将持续增长，这与常识性预测一致。

2024到2043年医疗支出的预测值：　　表2

年份：	2024，	预测医疗支出：	2605.21
年份：	2025，	预测医疗支出：	2769.51
年份：	2026，	预测医疗支出：	2934.69
年份：	2027，	预测医疗支出：	3102.51
年份：	2028，	预测医疗支出：	3273.84
年份：	2029，	预测医疗支出：	3449.16
年份：	2030，	预测医疗支出：	3629.15
年份：	2031，	预测医疗支出：	3813.47
年份：	2032，	预测医疗支出：	4002.41
年份：	2033，	预测医疗支出：	4195.67
年份：	2034，	预测医疗支出：	4393.38
年份：	2035，	预测医疗支出：	4596.20
年份：	2036，	预测医疗支出：	4802.85
年份：	2037，	预测医疗支出：	5014.17
年份：	2038，	预测医疗支出：	5229.60
年份：	2039，	预测医疗支出：	5449.53
年份：	2040，	预测医疗支出：	5673.69
年份：	2041，	预测医疗支出：	5902.39
年份：	2042，	预测医疗支出：	6134.98
年份：	2043，	预测医疗支出：	6371.80

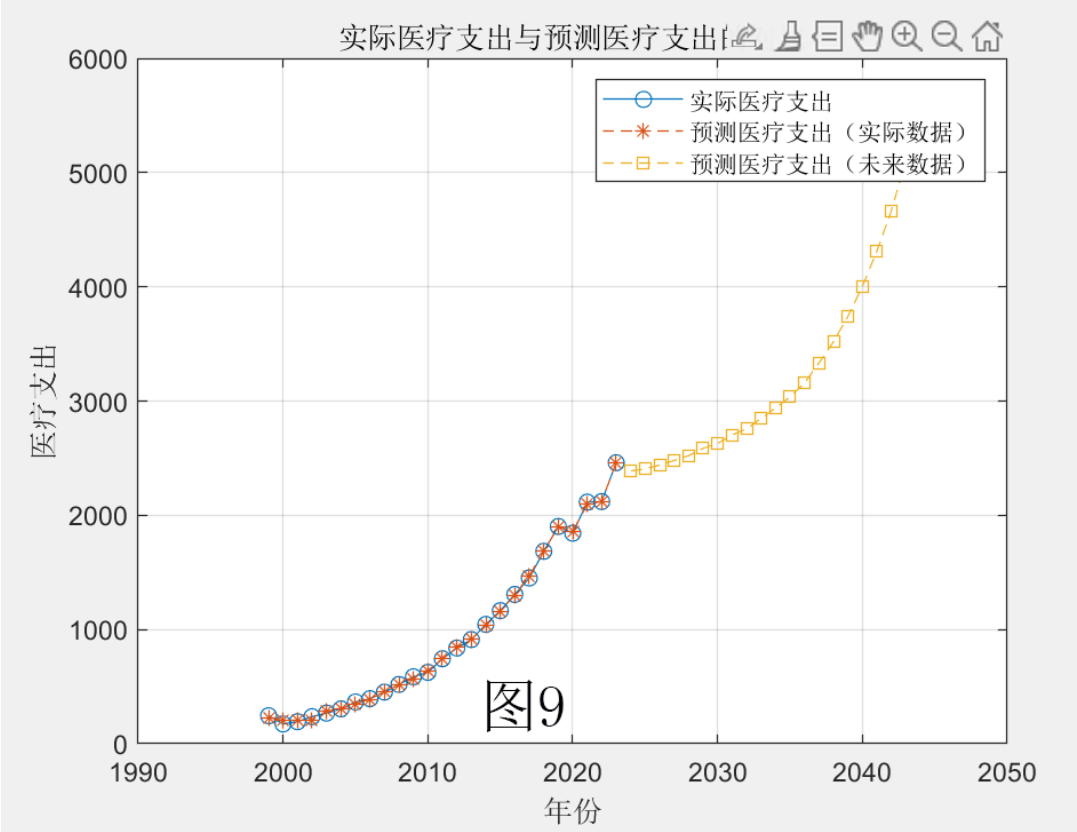


图9

5.2 问题二的求解

问题二要求分析某省未来最常见的疾病，我们根据要求选择了广东省并查找了相关数据。根据需求，采用了单指数平滑法（SES）进行拟合，并使用均方误差（MSE）、平均绝对误差（MAE）、平均绝对百分比误差（MAPE）用以评估模型。

5.2.1 问题二模型介绍与设定

单指数平滑（SES），是一种基本的时间序列预测方法，适用于没有明显趋势和季节性的数据。其核心思想是对历史数据进行加权平均，其中较近的观测值赋予更高的权重。权重随着观测值与预测点距离的增加而指数级减小。其公式见下。

$$L_t = \alpha \cdot y_t + (1 - \alpha) \cdot L_{t-1}$$

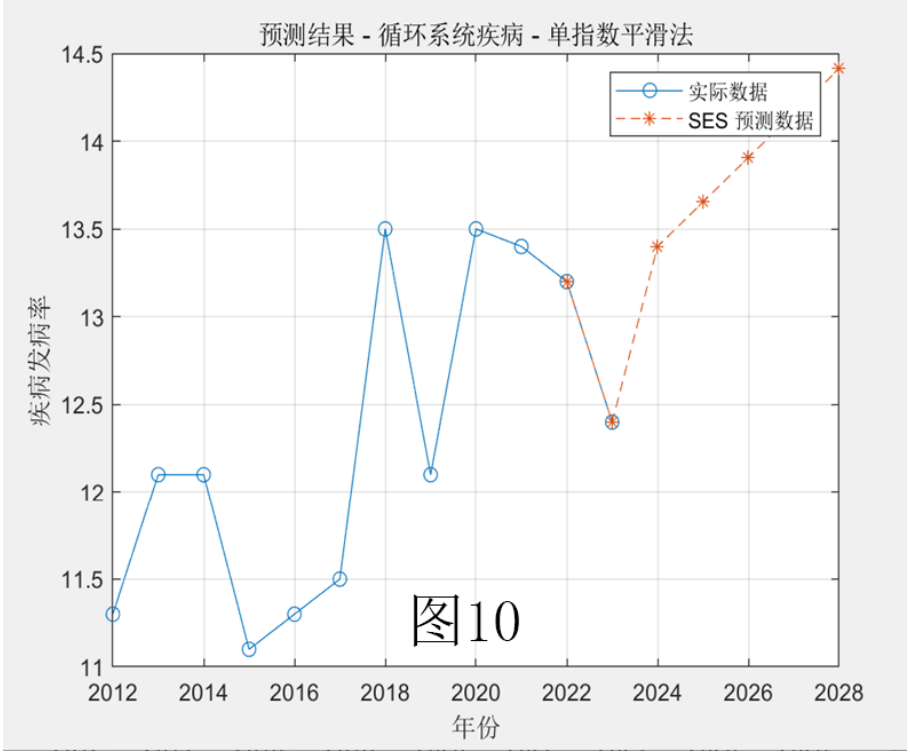
在这个模型中，选择合适的平滑系数 α 是 关键。通常通过最小化预测误差的方法来选择最佳的 α。不同的 α 值会影响平滑结果的灵敏度。α=1 时，SES 退化为简单的移动平均。较大的 α 值会使模型对最新数据的变化更加敏感，适合数据波动较大的情况。较小的 α 值则使模型对历史趋势更加稳重，适合数据变化较为平稳的情况。

SES 适用于那些没有明显趋势或季节性成分的时间序列数据。其优点是简单易行，计算量小，且对短期预测有较好的效果。

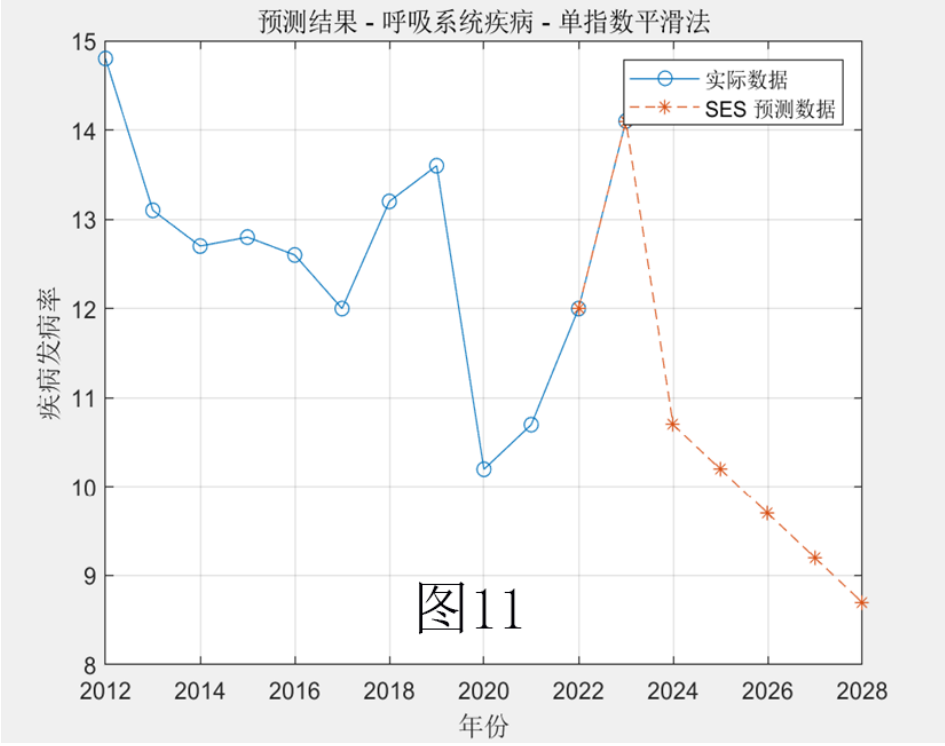
5.2.2 问题二模型求解与结论

对于求解问题二的模型，首先导入数据，对收集到的数据进行清洗和预处理，去除异常值，填补缺失值，并根据需要进行数据转换和标准化，以确保数据的质量和一致性，并划分训练集数据和测试集数据（假设使用前 10 年的数据训练，后 2 年的数据测试），保证求解模型的准确性。然后，根据题目和需求，我们拟定平滑系数 α 为 0.5，并设定为训练集的第一个值，然后根据每个时间按照公式进行更新平滑值，根据平滑值得到趋势值，利用趋势预测出测试

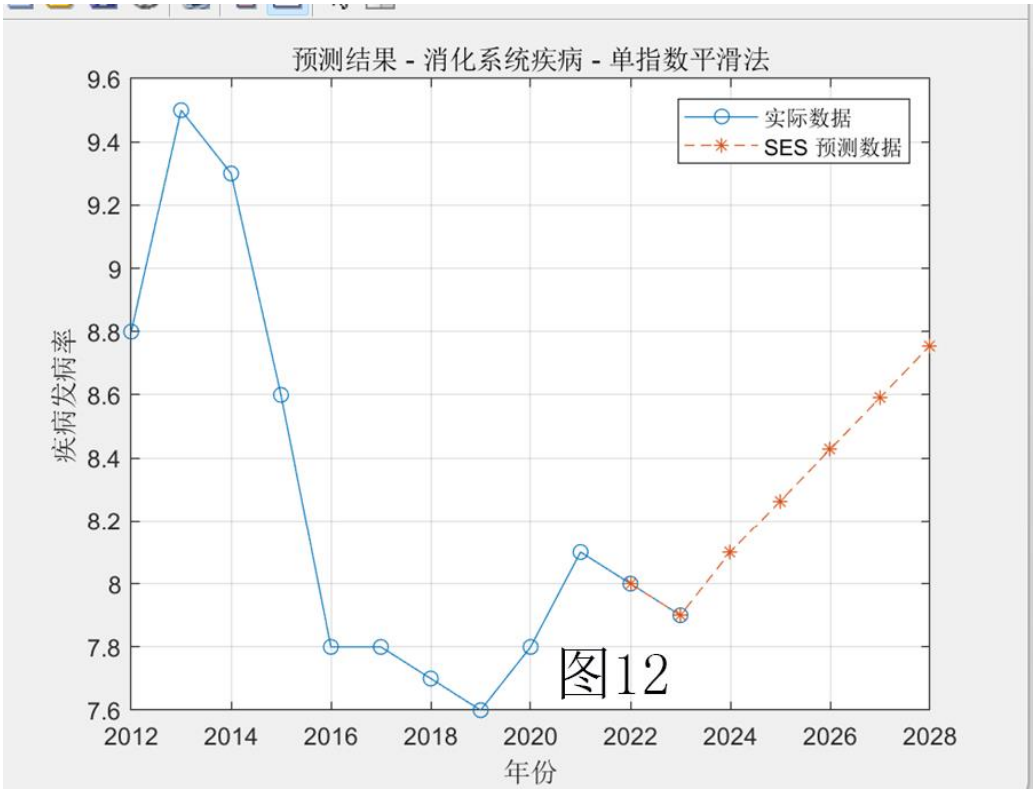
集和未来 5 年的发病率。最后，使用均方误差 (MSE)、平均绝对误差 (MAE)、平均绝对百分比误差 (MAPE) 来评价预测的准确性，并将预测结果以表格形式展示，包括每个未来年份的预测发病率。以下是预测结果。



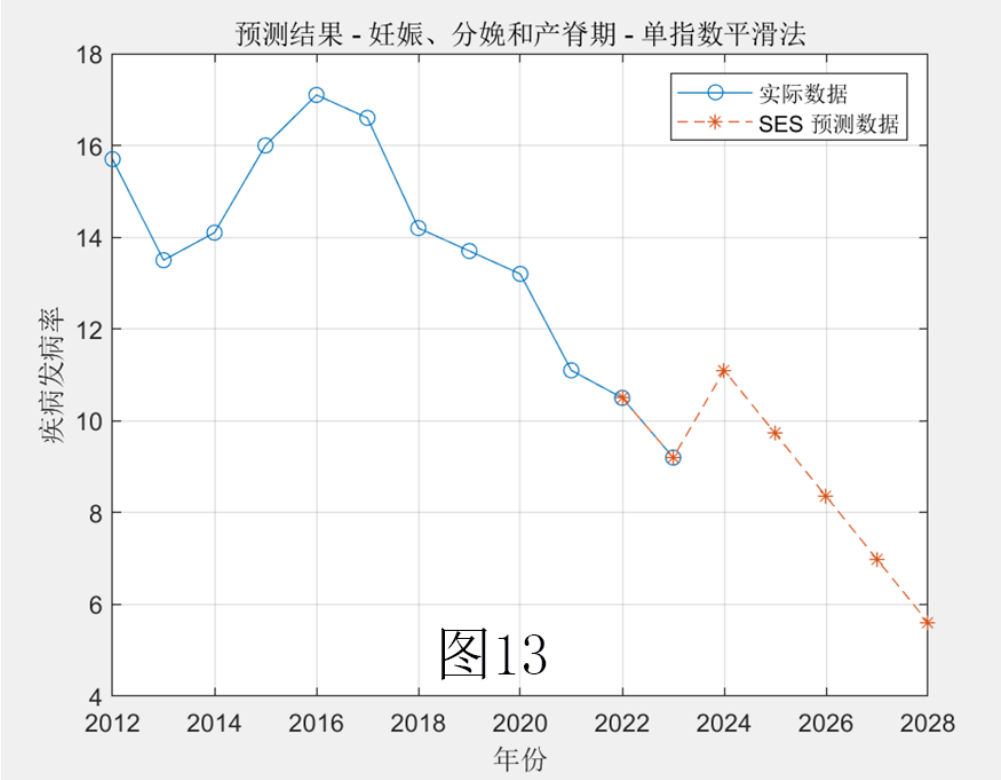
由图可以看出，未来的循环系统疾病发生率呈现一定的波动性，但总体呈现递增趋势。



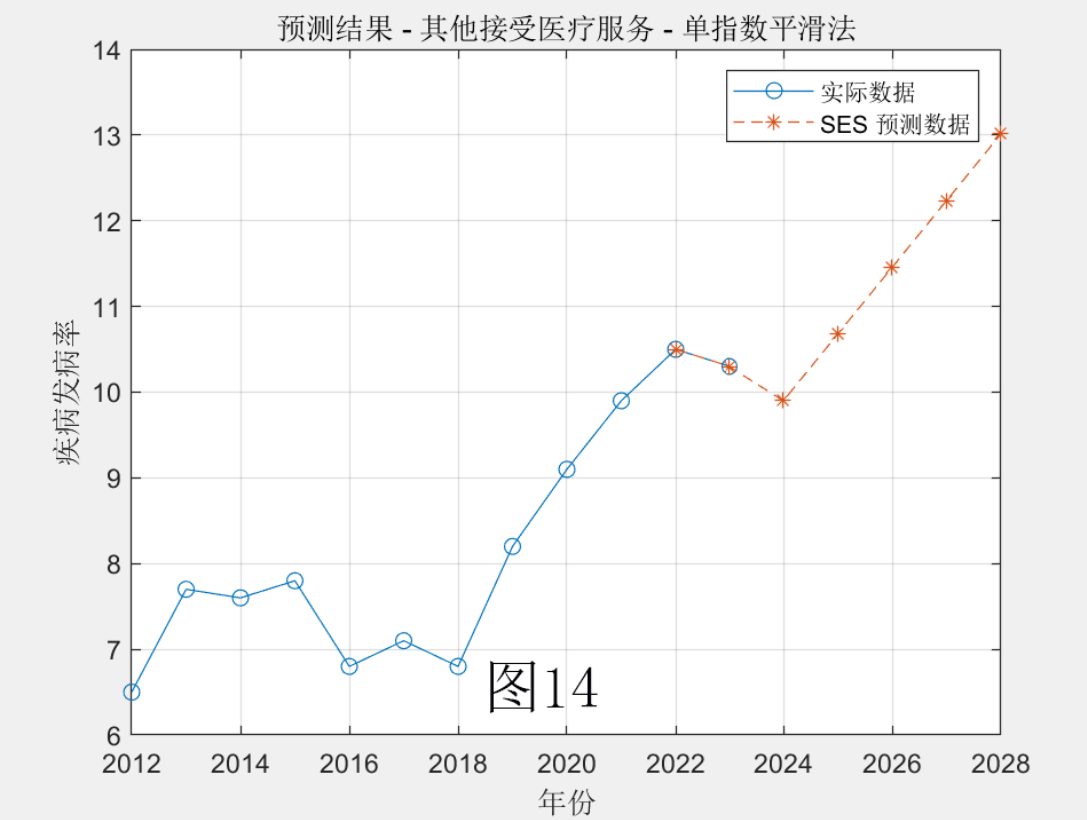
由图可以看出，未来的呼吸系统疾病发生率呈现一定的波动性，但总体呈现递减趋势。



由图可以看出，未来的消化系统疾病发生率在高峰后呈现大幅度下降，后面又慢慢上升。



由图可以看出，妊娠、分娩和产脊期类的疾病发生率呈现一定的波动性，但总体呈现递减趋势。



由图可以看出，其他接受医疗服务疾病类疾病发生率呈现一定的波动性，但总体呈现递减趋势。

在不发生突发情况和意外事件的情况下，可以看出这些预测数据与常识性认知相匹配。根据循环系统疾病的预测结果，可以看出未来几年该类疾病的发病率呈上升趋势，建议医院加大相关科室的资源配置，包括增加专科医生、病床和设备的投入，以应对患者数量的增加。根据呼吸系统疾病的预测结果，可以看出尽管呼吸系统疾病的预测发病率有下降趋势，但仍需加强对该类疾病的预防与管理，尤其是对空气质量和环境卫生的控制，以进一步降低发病率。消化系统疾病的发病率较稳定，但仍有小幅上升趋势，建议医院开展健康教育活动，倡导健康饮食习惯，降低该类疾病的发病率。妊娠、分娩和产脊期的发病率预测呈下降趋势，这可能与生育政策、人口结构和舆论有关。医院应密切关注社会变化，并根据需求调整产科服务，优化资源配置。其他接受医疗服务的预测发病率逐年上升，医院可以通过创新医疗服务模式，提升服务效率和质量，满足患者多样化的需求。总而言之，当地大型公立医院应当为了群众健康，

高效准时提供基本医疗服务和公共卫生服务，并根据发病趋势做好医院资源规划。做好医院人力分配、做好宣传疾病预防、做好医院内外提前面对疾病的心理准备等工作。

5.3 问题三的求解

5.3.1 问题三符号说明

符号	含义
λ	单位时间内到达检查点的患者数量即到达率
μ	单位时间内完成治疗服务的患者数量即服务率
c	检查点的服务台数量（单位：台）
W_q	患者在队列中的平均等待时间（平均等待时间）
L_q	排队系统中平均排队等待的请求数量（平均队列长度）
W	患者从到达系统到完成服务的平均时间（平均系统时间）
ρ	服务台的实际使用率（系统利用率）

5.3.2 模型假设

- 假设到达过程遵循**泊松过程**，即到达请求（如患者）的时间间隔服从**指数分布**。
- 假设服务时间遵循**指数分布**，即每个服务台处理请求的时间是随机的且服从指数分布。

5.3.3 模型建立

为了使建模更准确地描述题目，我们将患者分为三种类型，分别是急诊患者、紧急患者和普通门诊患者，并规定急诊患者的队列中的平均等待时间 $W_q \leq 30$，紧急患者 $W_q \leq 45$，普通门诊患者 $W_q \leq 60$，以此刻画出不同类型的患者的需求和可等待时间不同。同时，为了方便计算，我们将所有的疾病所需的检查疾病设备分为 2 个，分别为 X 光、CT，同时医院里有 1 个 X 光检查点和 1 个 CT 检查点。以下是患者看病的流程图。

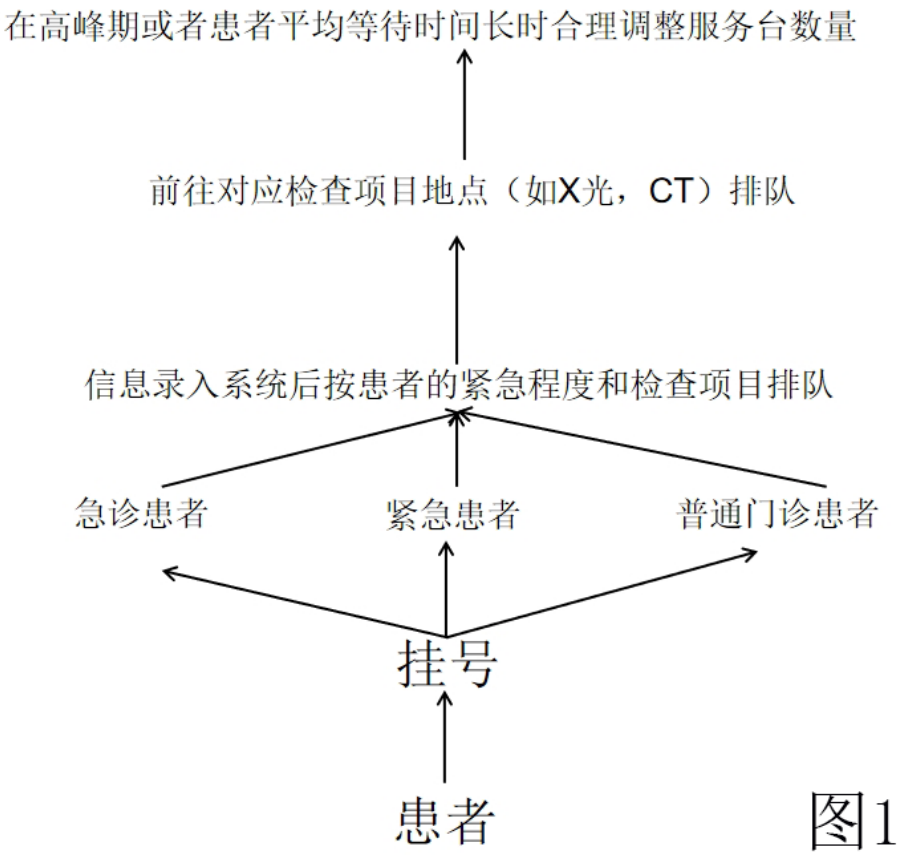


图15

我们找到了一种符合题目排队要求的模型，即 M/M/c 模型，用来表示多个检查点的服务系统，而在此模型中系统利用率、平均队列长度、平均等待时间和平均系统时间均是衡量排队关键的性能指标。

系统利用率(ρ)是指服务台的实际使用率，即在给定的时间内，服务台被占用的比例。它表示服务系统的繁忙程度，数值介于 0 和 1 之间。 ρ 越接近 1，表示系统越繁忙。其公式如下。

$$\rho = \frac{\lambda}{c \times \mu}$$

平均队列长度(L_q)表示在系统中排队等待的平均请求数量，它反映了请求在系统中等待服务的压力。其公式如下。

$$Lq = \rho \times \frac{(\lambda/\mu)^c}{c! \times (1-p)^2} \times P_0$$

平均等待时间（W_q）是指请求在系统中排队等待开始服务的平均时间。其公式如下。

$$Wq = L_q/\lambda$$

平均系统时间（W）是指请求从到达系统到完成服务所花费的平均时间。它包括了排队等待时间和服务时间。

$$W = W + 1/\mu$$

P_w 是指至少有一个请求正在排队等待服务的概率，数值越高，说明系统越可能出现排队等待，其公式如下。

$$P_w = \frac{\frac{(\lambda/\mu)^c}{c!}}{\sum_{n=0}^{c-1}\frac{(\lambda/\mu)^n}{n!} + \frac{(\lambda/\mu)^c}{c!(1-\rho)}}$$

整个模型的流程图如下。

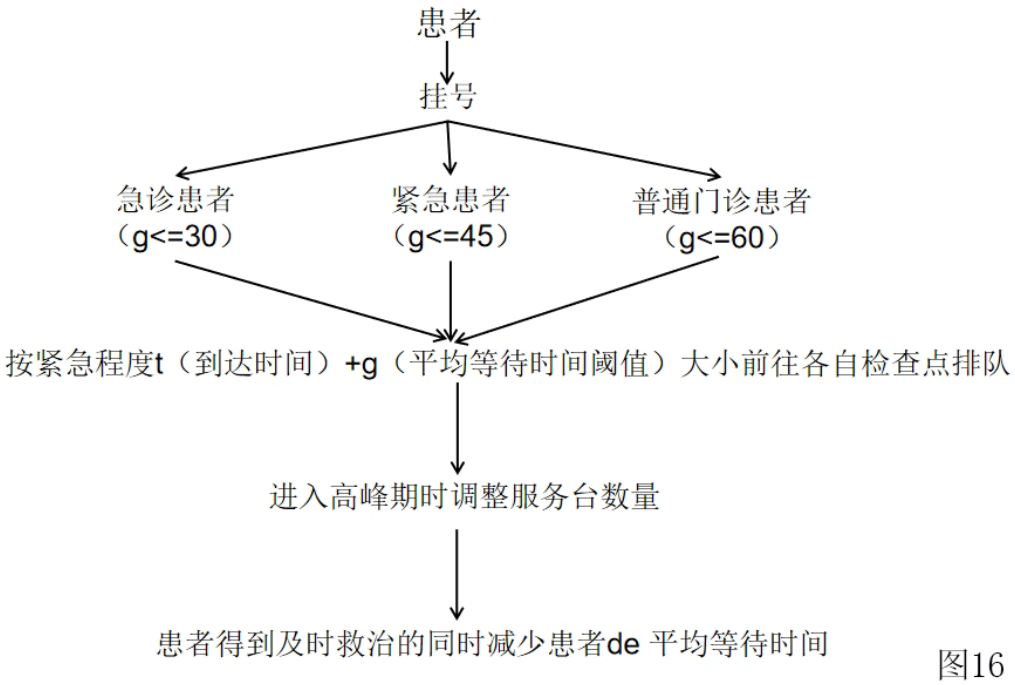


图16

5.3.4 模型求解

根据多次模拟排队和计算时间，我们将最佳方案确认为了两类，第一类是以优先级为第一排队依据，第二类是按时间紧急程度大小为第一排队依据，经过模拟和排队最后得到以下结果。

对于以优先级为第一排队依据。

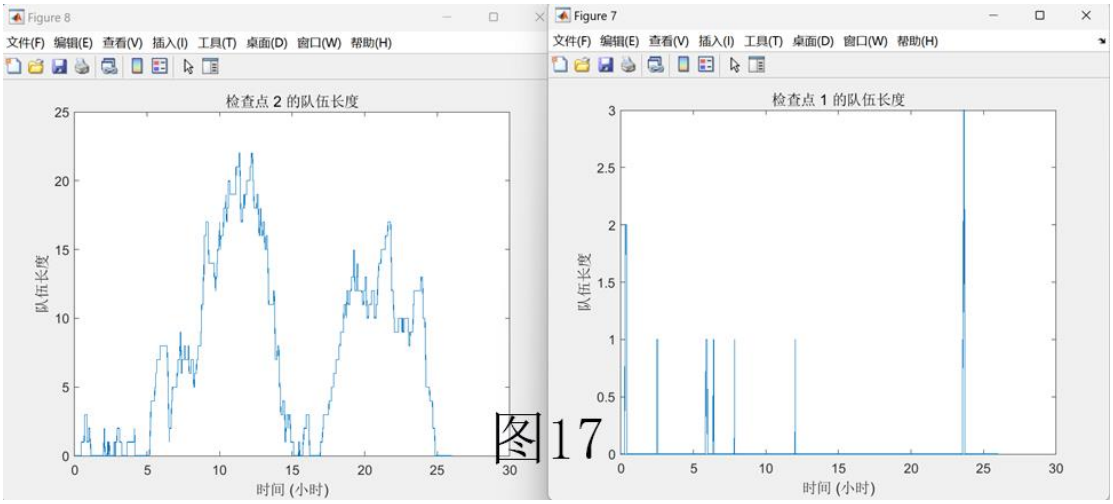


图17

```
检查点 1 每种患者类型的平均等待时间：
1 类型患者的平均等待时间为 0.27 分钟, 共有 52 人
2 类型患者的平均等待时间为 0.08 分钟, 共有 49 人
3 类型患者的平均等待时间为 0.28 分钟, 共有 43 人
计算总人口为 144
```

图18

```
检查点 2 每种患者类型的平均等待时间：
1 类型患者的平均等待时间为 14.31 分钟, 共有 62 人
2 类型患者的平均等待时间为 31.37 分钟, 共有 38 人
3 类型患者的平均等待时间为 63.00 分钟, 共有 56 人
计算总人口为 156

检查点 1 结果：
系统利用率 (ρ)：0.21
平均队列长度 (Lq)：0.02
平均等待时间 (Wq)：0.10 分钟
平均系统时间 (W)：2.32 分钟
至少有一个请求正在排队等待服务的概率 (Pw)：0.02

检查点 2 结果：
系统利用率 (ρ)：0.83
平均队列长度 (Lq)：3.59
平均等待时间 (Wq)：18.69 分钟
平均系统时间 (W)：23.00 分钟
至少有一个请求正在排队等待服务的概率 (Pw)：0.70
```

图19

对于以紧急程度为第一排队依据

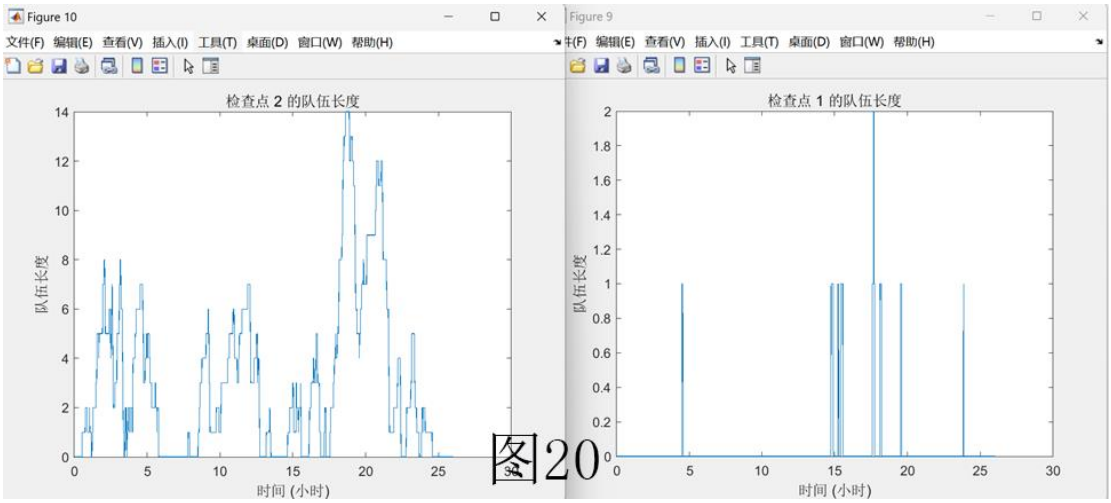


图20

```
检查点 1 每种患者类型的平均等待时间：
类型 1：0.50 分钟
类型 2：0.34 分钟
类型 3：2.73 分钟
```

图21

```
检查点 2 每种患者类型的平均等待时间：
类型 1：6.05 分钟
类型 2：9.96 分钟
类型 3：60.67 分钟

检查点 1 结果：
系统利用率 (ρ)：0.23
平均队列长度 (Lq)：0.12
平均等待时间 (Wq)：0.62 分钟
平均系统时间 (W)：3.04 分钟
有请求排队等待服务的概率 (Pw)：0.05

检查点 2 结果：
系统利用率 (ρ)：0.70
平均队列长度 (Lq)：2.56
平均等待时间 (Wq)：13.34 分钟
平均系统时间 (W)：16.98 分钟
有请求排队等待服务的概率 (Pw)：0.48
```

图22

由上我们可以提出一种通用的排队理论，即时间紧急程度大小为第一排队依据，除了先安排急诊患者、紧急患者之外，还安排快到达时间阈值的患者进行排队，这样不仅效率高，平均等待时间慢，而且比起方案一会大大提升患者的评价，更加人性化。那么，相关的最优排队方法也就是在安排急诊患者之后，针对紧急患者和普通患者，通过实时监控他们的等待时间来调整他们的排队顺序，确保那些接近时间阈值的患者能够及时得到服务。同时，医院还可以在这个过程中灵活资源配置、为患者提供透明的等待信息、收集患者对排队体验的反馈，分析数据以寻找改进点，以此进一步完善和优化制度。

5.4 问题四的求解

问题四要求在民营医院与公立医院之间的合作与竞争中找到最优解，以实现双方收益的最大化。为此，我们将建立合作竞争的博弈模型对比两者在合作

与竞争下的收益，进而确定最佳策略。

5.4.1 模型介绍与设定

假设某一时刻，某民营医院对某项目的投资基金为 G_1，收益为 R_1，某公立医院对同项目的投资基金为 G_2，收益为 R_2，双方合作总投资基金为 $G_0=G_1+G_2$，总收益为 $R_0=R_1+R_2$，假设二者的回报比例分别为 K_1 和 K_2，其中 $K_1+K_2=1$。为了简化分析，我们假设它们的投资策略相同和收益相同。

当双方都合作的时候，$R_1=K_1(R_0-G_0)$和 $R_2=K_2(R_0-G_0)$；

当双方都不合作的时候，双方收益分别为 R_1-G_1 和 R_2-G_2，设为 M_1 和 M_2；

当民营医院选择不合作而公立医院选择合作的时候，设民营医院的收益为 N，公立医院的收益为 H，则 $N=R_1-G_1+G$，而 $H=K_2(R_2-G_2)-G$；

当公立医院选择不合作而民营选择合作的时候，$H=K_1(R_1-G_1)-G$，而 $N= R_2-G_2+G$；

假设双方会找对方合作的概率为 λ，进行合作的期望收益为 $E_1=\lambda R+(1-\lambda)H$，不合作的期望收益为 $E_2=\lambda N+(1-\lambda)M$；

总的期望收益为，$E=\lambda E_1+(1-\lambda)E_2$；

整理即得 $E=\lambda^2(R+M-H-N)+\lambda(H+N-2M)+M$。

5.4.2 模型求解与结论

问题四的模型中包含四种情况，即双方合作、民营合作而公立不合作、公立合作而民营不合作、双方不合作，通过比较权衡这四种方案下的总的期望收益来确定最优策略。当 λ=1 时，拟认为模型拟合的是双方都合作，当 λ=0 时，拟认为模型拟合的是双方都拒绝合作。

考虑到医院采取合作策略的概率会根据时间的推进而发生改变，我们对 λ 求导，可得到

$$\frac{d\lambda}{dt}=\lambda(E_1-E)$$

该式表达出合作策略的概率的时间变化率。

再通过代入 E_1、E 和求解微分方程，绘制出随着 t 变化 λ 的变化图像，见

下。

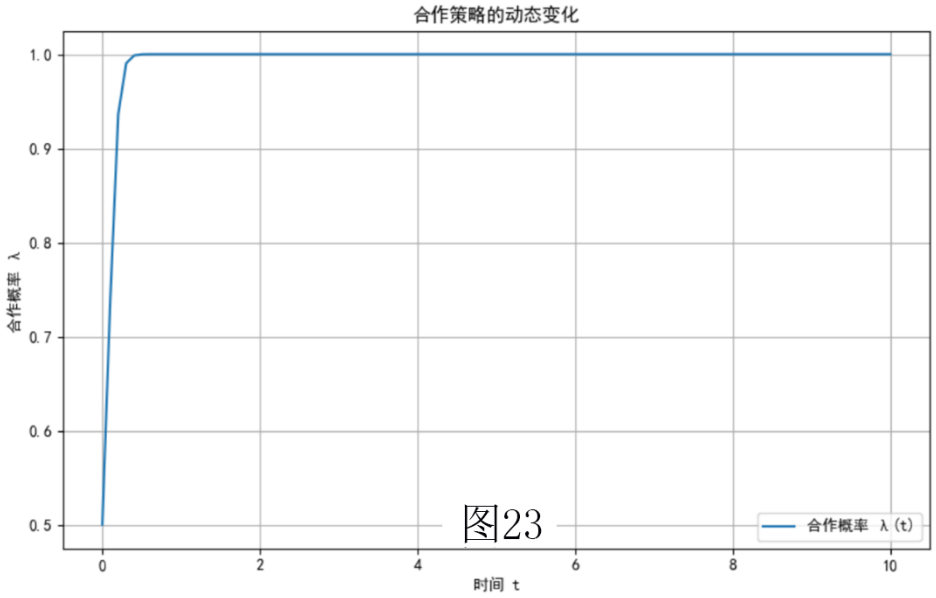


图23

有图可知，随着时间 t 的增加，合作概率 λ 非常迅速地达到了 1.0，并在接下来的时间内保持稳定。这表明在模型假设下，合作的概率很快达到最大值，并且维持在最高水平，暗示只有双方合作是一个稳定且优选的策略。

由上，可以对多家医院之间提出发展建议。首先，各家医院应当反思目前的问题与矛盾，对症下药，化劣为优，增加优势竞争力，不断提升医院水平与能力。公立医院由于自身的供需矛盾和管理制度的不完善，导致了公立医院存在着看病难、看病慢、服务差等问题，这需要公立医院进行供给侧和制度改革优化，而私立医院由于规模小，价格贵和信誉差，导致了私立医院不是人民看病的最优选或首选，这需要私立医院早日获得政府认可，完全纳入医保体系，增加医疗和服务质量，赢得公众舆论好感。其次，大力推进民营医院与公立医院互相取长补短，合作共赢，鄙弃零和博弈的思想，以此助力高质量发展与中国式现代化建设。具体而言，可以通过建立共享平台，让两者在技术、人才、和管理经验上相互学习和借鉴。例如，公立医院在疾病防控和重大疾病治疗方面通常有更丰富的经验和资源，而民营医院在服务效率和患者体验方面往往更具优势。通过合作，双方可以在保持各自优势的同时，弥补彼此的短板，提升服务质量和治疗效率。最后，尽管合作具有显著的优势，竞争仍然是推动医疗行业发展的一个关键因素。竞争激发了医院之间不断提升医疗技术、优化服务流程、降低成本的动力，从而直接影响到医疗服务的质量和效率。公立与民营

医院之间的健康竞争，能够促使各自在维持自身经济效益的同时，不断寻求创新和改进，以满足患者日益增长的医疗需求。为了实现这一点，各家医院应当不断提高医疗水平和服务质量，致力于创新医疗技术和服务政策，以此确保在激烈的市场竞争中保持竞争力，避免被市场淘汰。

六. 模型评价、改进、推广

6.1 模型优点

1. **模型的多样性与全面性**：本文采用了 ARIMA 模型、多项式回归模型、单指数平滑模型、排队理论中的 M/M/c 模型和博弈论模型。
2. **数据的充分利用**：模型建立在广泛的统计数据基础之上，利用国家统计局、卫生年鉴等权威来源的数据，保证了模型的科学性和可靠性。
3. **实际应用性强**：无论是对老龄化和医疗需求的预测，还是对常见疾病的分析，以及医院的排队优化方法，模型的结论均具有较强的实际应用价值，对医院管理决策具有指导意义。

6.2 模型缺点

1. **假设条件的限制**：模型的很多结论基于较为理想的假设条件，如医院外部环境保持稳定，忽略了突发事件、政策变化等因素的影响，这其实会影响预测结果的准确性。
2. **模型的复杂性与可操作性**：某些模型如博弈论的应用虽然在理论上可行，但在实际操作中可能较难实现。尤其是在医院管理层面，合作与竞争的平衡需要更为复杂的策略和动态调整。
3. **模型精度的评估不足**：虽然模型在评估过程中使用了一些误差分析方法，但对于预测精度的评估和验证相对简单，缺乏更为深入的误差来源分析和不确定性评估。

6.3 模型改进与推广

1.**引入动态调整机制**：在排队理论和博弈模型中，可以考虑引入动态调整机制，以应对突发事件或政策变化。这可以通过实时数据监控和反馈系统来实现，使得模型在实际应用中更具灵活性。

2.**考虑更多外部因素**：模型改进时可以引入更多的外部因素，如政策变化、经济波动、突发公共卫生事件等，

3.**推广应用**：本文的模型可以推广应用于不同地区和医院类型，尤其是在公立与私立医院的合作竞争中，通过建立一个共享平台或数据交换机制，推动更多医院之间的合作与竞争，提升整体医疗服务水平。

七.参考文献

[1] 习近平.高举中国特色社会主义伟大旗帜 为全面建设社会主义现代化国家而团结奋斗[N].人民日报,2022-10-16(1).

[2] 盛耀,谢式千,潘承毅,概率论与数理统计（第四版）,高等教育出版社,2008年.

[3] 饶海琴，唐小霞.线上外卖业务的竞合博弈探析[D].2017.

八. 附录

8.1 支撑文件

8.2 代码

8.2.1 问题一代码

导入使用的数据：

```
% 数据按年份从 1999 到 2023 排列
years = [1999; 2000; 2001; 2002; 2003; 2004; 2005; 2006; 2007; 2008; ...
2009; 2010; 2011; 2012; 2013; 2014; 2015; 2016; 2017; 2018; ...
2019; 2020; 2021; 2022; 2023];

gdp = [8.7000e+10; 1.0200e+11; 1.1800e+11; 1.3200e+11; 1.5000e+11; ...
1.8000e+11; 2.1700e+11; 2.6000e+11; 3.1000e+11; 3.4900e+11; ...
4.0000e+11; 4.9200e+11; 5.6900e+11; 6.2800e+11; 7.0000e+11; ...
7.4400e+11; 8.2700e+11; 8.7100e+11; 9.3700e+11; 9.9000e+11; ...
1.0490e+12; 1.0160e+12; 1.1400e+12; 1.2100e+12; 1.3300e+12];

income = [3485; 3721; 4070; 4532; 5007; 5661; 6385; 7229; 8584; 9957; ...
10977; 12520; 14551; 16510; 18311; 20167; 21966; 23821; 25974; ...
28228; 30733; 32189; 35128; 36883; 39218];

consumption = [2658; 2914; 3139; 3548; 3889; 4395; 5035; 5634; 6592;
7548; ...
8377; 9378; 10820; 12054; 13220; 14491; 15712; 17111; ...
18322; 19853; 21559; 21210; 24100; 24538; 26796];
```

```
healthcare_spending = [245; 173; 193; 237; 268; 305; 366; 395; 452; 519;
586; ...
625; 744; 838; 912; 1045; 1165; 1307; 1451; ...
1685; 1902; 1843; 2115; 2120; 2460];
% 年龄段比例数据（按年份从 1999 到 2023 排列）
age_0_14_ratio = [18.27; 18.86; 19.34; 19.68; 18.33; 18.27; 17.99; ...
17.69; 17.27; 17.10; 16.80; 16.73; 16.50; 16.38; ...
18.04; 18.28; 18.55; 18.65; 18.93; 19.89; 20.25; ...
20.38; 20.33; 20.55; 22.66];

age_15_64_ratio = [76.54; 75.97; 75.63; 75.41; 77.04; 76.98; 76.88; ...
76.79; 76.42; 76.08; 75.71; 75.11; 74.40; 73.53; ...
71.30; 70.24; 69.28; 68.28; 67.28; 65.59; 64.52; ...
63.95; 63.61; 62.98; 60.41];

age_65_up_ratio = [6.90; 6.96; 7.10; 7.30; 7.50; 7.58; 7.69; 7.93; ...
8.05; 8.25; 8.47; 8.87; 9.10; 9.40; 9.70; 10.10; ...
10.50; 10.80; 11.40; 11.90; 12.60; 13.50; 14.20; ...
14.86; 15.40];

%预测 2024 到 2043 的数据
income_forecast = [41666; 44097; 46616; 49223; 51917; 54699; 57569; ...
60527; 63572; 66705; 69926; 73235; 76631; 80115; ...
83687; 87346; 91093; 94928; 98851; 1.0286e+05];

age_15_64_forecast = [60.15; 59.08; 58.09; 57.14; 56.21; 55.28; 54.35; ...
53.41; 52.47; 51.52; 50.57; 49.60; 48.64; 47.66; ...
46.69; 45.71; 44.73; 43.74; 42.76; 41.77];

age_65_plus_forecast = [16.45; 17.26; 18.05; 18.84; 19.64; 20.44;
21.27; ...
22.10; 22.96; 23.82; 24.70; 25.60; 26.51; 27.42; ...
28.35; 29.29; 30.24; 31.20; 32.17; 33.14];
% 主脚本 check_multicollinearity_script.m
% 加载数据
years = [1999; 2000; 2001; 2002; 2003; 2004; 2005; 2006; 2007; 2008; ...
2009; 2010; 2011; 2012; 2013; 2014; 2015; 2016; 2017; 2018; ...
2019; 2020; 2021; 2022; 2023];

income = [3485; 3721; 4070; 4532; 5007; 5661; 6385; 7229; 8584; 9957; ...
10977; 12520; 14551; 16510; 18311; 20167; 21966; 23821; 25974; ...
28228; 30733; 32189; 35128; 36883; 39218];

age_65_up_ratio = [6.90; 6.96; 7.10; 7.30; 7.50; 7.58; 7.69; 7.93; ...
```

```
8.05; 8.25; 8.47; 8.87; 9.10; 9.40; 9.70; 10.10; ...
10.50; 10.80; 11.40; 11.90; 12.60; 13.50; 14.20; ...
14.86; 15.40];

% 将收入和 65 岁以上人口比例数据合并到一个矩阵
X = [income, age_65_up_ratio];
% 计算相关系数
correlation = corr(X(:, 1), X(:, 2));
fprintf('收入和 65 岁以上人口比例之间的相关系数: %.4f\n', correlation);

% 计算 VIF
vif_income = calculateVIF(X, 1);
vif_age_65_up = calculateVIF(X, 2);

% 输出 VIF
fprintf('收入的 VIF: %.4f\n', vif_income);
fprintf('65 岁以上人口比例的 VIF: %.4f\n', vif_age_65_up);

% 检查相关系数和 VIF
if abs(correlation) > 0.8
disp('警告: 收入和 65 岁以上人口比例之间的相关性较高。');
end

if vif_income > 10
fprintf('收入的 VIF 较高: %.4f\n', vif_income);
end

if vif_age_65_up > 10
fprintf('65 岁以上人口比例的 VIF 较高: %.4f\n', vif_age_65_up);
end

% 如果共线性较低
if abs(correlation) <= 0.8 && vif_income <= 10 && vif_age_65_up <= 10
disp('收入和 65 岁以上人口比例之间的共线性较低。');
end
```

```
%calculateVIF.m

% 计算 VIF
function vif_value = calculateVIF(X, var_idx)
% X 是预测变量的矩阵
```

```
% var_idx 是要计算 VIF 的变量的索引
% 去掉 var_idx 对应的列
X_other = X(:, setdiff(1:size(X, 2), var_idx));
Xi = X(:, var_idx);
% 拟合模型
model = fitlm(X_other, Xi);
R2 = model.Rsquared.Ordinary;
% 计算 VIF
vif_value = 1 / (1 - R2);
end
```

预测（多项式回归:二次）

```
% 主脚本
% 加载数据
years = [1999; 2000; 2001; 2002; 2003; 2004; 2005; 2006; 2007; 2008; ...
2009; 2010; 2011; 2012; 2013; 2014; 2015; 2016; 2017; 2018; ...
2019; 2020; 2021; 2022; 2023];

income = [3485; 3721; 4070; 4532; 5007; 5661; 6385; 7229; 8584; 9957; ...
10977; 12520; 14551; 16510; 18311; 20167; 21966; 23821; 25974; ...
28228; 30733; 32189; 35128; 36883; 39218];

age_0_14_ratio = [18.27; 18.86; 19.34; 19.68; 18.33; 18.27; 17.99; ...
17.69; 17.27; 17.10; 16.80; 16.73; 16.50; 16.38; ...
18.04; 18.28; 18.55; 18.65; 18.93; 19.89; 20.25; ...
20.38; 20.33; 20.55; 22.66];

age_65_up_ratio = [6.90; 6.96; 7.10; 7.30; 7.50; 7.58; 7.69; 7.93; ...
8.05; 8.25; 8.47; 8.87; 9.10; 9.40; 9.70; 10.10; ...
10.50; 10.80; 11.40; 11.90; 12.60; 13.50; 14.20; ...
14.86; 15.40];

healthcare_spending = [245; 173; 193; 237; 268; 305; 366; 395; 452;
519; ...
586; 625; 744; 838; 912; 1045; 1165; 1307; 1451; ...
1685; 1902; 1843; 2115; 2120; 2460];

% 预测数据
future_years = (2024:2043)';
income_forecast = [41666; 44097; 46616; 49223; 51917; 54699; 57569; ...
```

```
60527; 63572; 66705; 69926; 73235; 76631; 80115; ...
83687; 87346; 91093; 94928; 98851; 1.0286e+05];

age_15_64_forecast = [60.15; 59.08; 58.09; 57.14; 56.21; 55.28; 54.35; ...
53.41; 52.47; 51.52; 50.57; 49.60; 48.64; 47.66; ...
46.69; 45.71; 44.73; 43.74; 42.76; 41.77];

age_65_plus_forecast = [16.45; 17.26; 18.05; 18.84; 19.64; 20.44;
21.27; ...
22.10; 22.96; 23.82; 24.70; 25.60; 26.51; 27.42; ...
28.35; 29.29; 30.24; 31.20; 32.17; 33.14];

% 重新加载预测数据中的年龄比例，以匹配列顺序
age_0_14_forecast = 100 - age_15_64_forecast - age_65_plus_forecast;

% 将数据合并到一个矩阵
X = [income, age_0_14_ratio, age_65_up_ratio];

% 创建二次多项式特征
X_poly = [X, X(:,1).^2, X(:,2).^2, X(:,3).^2, X(:,1).*X(:,2),
X(:,1).*X(:,3), X(:,2).*X(:,3)];

% 拟合模型
mdl = fitlm(X_poly, healthcare_spending);

% 处理未来数据
X_future = [income_forecast, age_0_14_forecast, age_65_plus_forecast];
X_future_poly = [X_future, X_future(:,1).^2, X_future(:,2).^2,
X_future(:,3).^2, X_future(:,1).*X_future(:,2),
X_future(:,1).*X_future(:,3), X_future(:,2).*X_future(:,3)];

% 预测医疗支出
future_predictions = predict(mdl, X_future_poly);

% 计算均方误差和均方根误差
actual_healthcare_spending = healthcare_spending;
predicted_healthcare_spending = predict(mdl, X_poly);
mse = mean((actual_healthcare_spending -
predicted_healthcare_spending).^2);
rmse = sqrt(mse);

% 显示预测结果
disp('2024 到 2043 年医疗支出的预测值:');
for i = 1:length(future_years)
```

```
fprintf('年份: %d, 预测医疗支出: %.2f\n', future_years(i),
future_predictions(i));
end

% 显示 MSE 和 RMSE
fprintf('均方误差 (MSE): %.2f\n', mse);
fprintf('均方根误差 (RMSE): %.2f\n', rmse);

% 绘制实际数据和预测数据的图
figure;
plot(years, healthcare_spending, 'o-', 'DisplayName', '实际医疗支出');
hold on;
plot(years, predicted_healthcare_spending, '*--', 'DisplayName', '预测医疗支
出（实际数据）');
plot(future_years, future_predictions, 's--', 'DisplayName', '预测医疗支出
（未来数据）');
xlabel('年份');
ylabel('医疗支出');
title('实际医疗支出与预测医疗支出的对比');
legend('show');
grid on;
```

8.2.2 问题二代码

```
% 数据定义
years = (2012:2023)'; % 已有的数据年份
forecast_years = (2024:2028)'; % 预测未来 5 年的年份
n_forecast = length(forecast_years); % 预测年份的数量

% 疾病数据
disease_data = {
'循环系统疾病', [11.3, 12.1, 12.1, 11.1, 11.3, 11.5, 13.5, 12.1, 13.5, 13.4,
13.2, 12.4]';
'呼吸系统疾病', [14.8, 13.1, 12.7, 12.8, 12.6, 12, 13.2, 13.6, 10.2, 10.7,
12, 14.1]';
'消化系统疾病', [8.8, 9.5, 9.3, 8.6, 7.8, 7.8, 7.7, 7.6, 7.8, 8.1, 8, 7.9]';
'妊娠、分娩和产脊期', [15.7, 13.5, 14.1, 16, 17.1, 16.6, 14.2, 13.7, 13.2,
11.1, 10.5, 9.2]';
'其他接受医疗服务', [6.5, 7.7, 7.6, 7.8, 6.8, 7.1, 6.8, 8.2, 9.1, 9.9, 10.5,
10.3]'
};

alpha = 0.5; % 单指数平滑的平滑系数
```

```
for i = 1:length(disease_data)
disease_name = disease_data{i, 1};
disease_rate = disease_data{i, 2};
% 划分训练集和测试集 (假设使用前 10 年的数据训练，后 2 年的数据测试)
train_data = disease_rate(1:end-2); % 选择前 length(disease_rate) - 2 个元素
作为训练集
test_data = disease_rate(end-1:end); % 选择最后 2 个元素作为测试集

% 单指数平滑法 (SES)
% 初始化平滑值
L_ses = zeros(length(train_data), 1);

% 初始化第一个值
L_ses(1) = train_data(1);

% 单指数平滑
for t = 2:length(train_data)
L_ses(t) = alpha * train_data(t) + (1 - alpha) * L_ses(t-1);
end

% 预测测试集和未来 5 年的发病率
% 注意 SES 的预测为水平分量加上预测的趋势
trend_ses = L_ses(end) - L_ses(end-1); % 计算趋势（水平分量变化）
ses_forecast_test = L_ses(end) + (1:2)' * trend_ses; % 预测测试集的值
ses_forecast = L_ses(end) + (1:n_forecast)' * trend_ses; % 预测未来 5 年的值

% 误差计算
mse_ses = mean((test_data - ses_forecast_test).^2);
mae_ses = mean(abs(test_data - ses_forecast_test));
mape_ses = mean(abs((test_data - ses_forecast_test) ./ test_data)) * 100;

% 输出结果
fprintf('%s 单指数平滑的 MSE: %.4f, MAE: %.4f, MAPE: %.2f%%\n',
disease_name, mse_ses, mae_ses, mape_ses);

% 将实际数据和预测结果合并
all_years = [years; forecast_years];
all_data = [disease_rate; ses_forecast];

% 绘图
figure;
plot(years, disease_rate, '-o', 'DisplayName', '实际数据');
hold on;
```

```
plot([years(end-1:end); forecast_years], [test_data; ses_forecast], '--*',
'DisplayName', 'SES 预测数据');
xlabel('年份');
ylabel('疾病发病率');
title(['预测结果 - ' disease_name ' - 单指数平滑法']);
legend;
grid on;
hold off;

% 显示预测结果
disp(['基于单指数平滑法的预测结果 - ' disease_name ':']);
disp(table(forecast_years, ses_forecast, 'VariableNames', {'年份', '预测发病
率'}));
end
% 数据定义
years = (2012:2023)'; % 已有的数据年份
forecast_years = (2024:2028)'; % 预测未来 5 年的年份
n_forecast = length(forecast_years); % 预测年份的数量

% 疾病数据
disease_data = {
'循环系统疾病', [11.3, 12.1, 12.1, 11.1, 11.3, 11.5, 13.5, 12.1, 13.5, 13.4,
13.2, 12.4]';
'呼吸系统疾病', [14.8, 13.1, 12.7, 12.8, 12.6, 12, 13.2, 13.6, 10.2, 10.7,
12, 14.1]';
'消化系统疾病', [8.8, 9.5, 9.3, 8.6, 7.8, 7.8, 7.7, 7.6, 7.8, 8.1, 8, 7.9]';
'妊娠、分娩和产脊期', [15.7, 13.5, 14.1, 16, 17.1, 16.6, 14.2, 13.7, 13.2,
11.1, 10.5, 9.2]';
'其他接受医疗服务', [6.5, 7.7, 7.6, 7.8, 6.8, 7.1, 6.8, 8.2, 9.1, 9.9, 10.5,
10.3]'
};

alpha = 0.5; % 单指数平滑的平滑系数

for i = 1:length(disease_data)
disease_name = disease_data{i, 1};
disease_rate = disease_data{i, 2};
% 划分训练集和测试集 (假设使用前 10 年的数据训练，后 2 年的数据测试)
train_data = disease_rate(1:end-2); % 选择前 length(disease_rate) - 2 个元素
作为训练集
test_data = disease_rate(end-1:end); % 选择最后 2 个元素作为测试集

% 单指数平滑法 (SES)
% 初始化平滑值
```

```
L_ses = zeros(length(train_data), 1);

% 初始化第一个值
L_ses(1) = train_data(1);

% 单指数平滑
for t = 2:length(train_data)
L_ses(t) = alpha * train_data(t) + (1 - alpha) * L_ses(t-1);
end

% 预测测试集和未来 5 年的发病率
% 注意 SES 的预测为水平分量加上预测的趋势
trend_ses = L_ses(end) - L_ses(end-1); % 计算趋势（水平分量变化）
ses_forecast_test = L_ses(end) + (1:2)' * trend_ses; % 预测测试集的值
ses_forecast = L_ses(end) + (1:n_forecast)' * trend_ses; % 预测未来 5 年的值

% 误差计算
mse_ses = mean((test_data - ses_forecast_test).^2);
mae_ses = mean(abs(test_data - ses_forecast_test));
mape_ses = mean(abs((test_data - ses_forecast_test) ./ test_data)) * 100;

% 输出结果
fprintf('%s 单指数平滑的 MSE: %.4f, MAE: %.4f, MAPE: %.2f%%\n',
disease_name, mse_ses, mae_ses, mape_ses);

% 将实际数据和预测结果合并
all_years = [years; forecast_years];
all_data = [disease_rate; ses_forecast];

% 绘图
figure;
plot(years, disease_rate, '-o', 'DisplayName', '实际数据');
hold on;
plot([years(end-1:end); forecast_years], [test_data; ses_forecast], '--*',
'DisplayName', 'SES 预测数据');
xlabel('年份');
ylabel('疾病发病率');
title(['预测结果 - ' disease_name ' - 单指数平滑法']);
legend;
grid on;
hold off;

% 显示预测结果
disp(['基于单指数平滑法的预测结果 - ' disease_name ':']);
```

```
disp(table(forecast_years, ses_forecast, 'VariableNames', {'年份', '预测发病率'}));
end
```

8.2.3 问题三代码

```
close all;
clear;
% 参数设置
num_patients = 300; % 总患者数量
num_checkpoints = 2; % 检查点数量
initial_services = [2, 1]; % 初始服务台数量，分别对应检查点 1 和检查点 2
peak_services = [3, 2]; % 高峰时段服务台数量，分别对应检查点 1 和检查点 2
average_service_times = [10, 15]; % 每个检查点的平均服务时间（分钟），分别对应检查点 1 和检查点 2
min_service_time = 1; % 最小服务时间（分钟）
tys = zeros(num_checkpoints, 4); % 每个检查点的类型人数
ty_time = zeros(num_checkpoints, 4); % 每个检查点的类型等待时间

% 随机数生成器初始化
rng('shuffle'); % 初始化随机数生成器

% 生成随机患者到达时间和服务时间
arrival_times = zeros(1, num_patients); % 存储患者到达时间
service_times = zeros(num_patients, num_checkpoints); % 存储每个患者的服务时间
patient_types = zeros(1, num_patients); % 存储患者类型
checkpoints_assigned = zeros(1, num_patients); % 存储每个患者分配的检查点

prev_time = 0; % 上一个患者的到达时间
for i = 1:num_patients
arrival_interval = exprnd(60); % 生成下一个到达间隔时间（分钟）
prev_time = mod(prev_time + ceil(arrival_interval), 24 * 60); % 更新并确保到达时间在一天内
arrival_times(i) = prev_time;

% 生成服务时间
service_dist = average_service_times(mod(i-1, num_checkpoints) + 1); % 根据检查点的平均服务时间生成服务时间
raw_service_time = max(ceil(exprnd(service_dist)), min_service_time); % 确保服务时间不小于最小值
service_times(i, mod(i-1, num_checkpoints) + 1) = raw_service_time;
```

```
% 随机生成患者类型和检查点分配
patient_types(i) = randi([1, 3]); % 生成患者类型（1 到 3）
checkpoints_assigned(i) = randi([1, num_checkpoints]); % 随机选择检查点（从 1
开始）
end

% 初始化每个检查点的队列
queues = cell(1, num_checkpoints);
for i = 1:num_checkpoints
queues{i} = [];
end

% 根据检查点分配将患者分配到对应的队列
for i = 1:num_patients
checkpoint = checkpoints_assigned(i); % 当前患者的检查点
queues{checkpoint} = [queues{checkpoint}; arrival_times(i),
patient_types(i), service_times(i, checkpoint)]; % 添加患者到对应检查点的队列
end

% 对每个检查点的队列按到达时间排序
for i = 1:num_checkpoints
queues{i} = sortrows(queues{i}, [1, 2]); % 按到达时间排序，然后按优先级排序
end

% 初始化变量
results = struct(); % 用于存储每个检查点的结果
total_simulation_time = 26 * 60; % 总模拟时间（分钟）

for i = 1:num_checkpoints
% 初始化变量
total_busy_time = 0; % 总服务台忙碌时间
total_wait_time = 0; % 总等待时间
total_system_time = 0; % 总系统时间
queue_wait_times = []; % 记录每个请求的等待时间
num_requests_with_queue = 0; % 有请求排队的总次数
len_time = zeros(1, total_simulation_time + 1); % 对应时间的队伍长度
kongzt = zeros(1, initial_services(i)); % 控制台空闲下来的时间
queue_len = []; % 等待队列

for t = 0:total_simulation_time

% 把到达的人加入等待队伍
arrival_indices = find(queues{i}(:, 1) == t);
```

```
if ~isempty(arrival_indices)
queue_len = [queue_len; queues{i}(arrival_indices, :)];
end

% 给入队的人排序
if ~isempty(queue_len)
queue_len = sortrows(queue_len, [2, 1]); % 按优先级排序，然后按到达时间排序
end
len = size(queue_len, 1);
% 高峰时间调整
if len >= 10 % 如果队列长度大于等于 10
if length(kongzt) < peak_services(i)
kongzt = [kongzt, zeros(1, peak_services(i) - length(kongzt))]; % 增加服务台数到高峰值
end
else
if length(kongzt) > initial_services(i)
kongzt = kongzt(1:initial_services(i)); % 如果服务台数多于初始服务台数，调整为初始服务台数
end
end

% 寻找空闲服务台
if ~isempty(queue_len)
for p = 1:length(kongzt)
if kongzt(p) <= t
if size(queue_len, 1) > 0
a = queue_len(1, :);
queue_len(1, :) = [];
kongzt(p) = t + a(3);
total_wait_time = total_wait_time + (t - a(1));
total_system_time = total_system_time + (t - a(1) + a(3));
total_busy_time = total_busy_time + a(3);
queue_wait_times = [queue_wait_times, t - a(1)];
tys(i, a(2)) = tys(i, a(2)) + 1;
ty_time(i, a(2)) = ty_time(i, a(2)) + (t - a(1));
end
end
end
end

% 记录队伍长度
len_time(t + 1) = size(queue_len, 1);
```

```
% 记录队列中有请求的时间
if size(queue_len, 1) > 0
num_requests_with_queue = num_requests_with_queue + 1;
end
end

% 计算系统利用率 (ρ)
total_service_time = total_busy_time; % 总服务时间
rho = total_service_time / (total_simulation_time * initial_services(i)); %
系统利用率

% 计算平均队列长度 (Lq)
average_queue_length = mean(len_time);

% 计算平均等待时间 (Wq)
average_wait_time = total_wait_time / num_patients;

% 计算平均系统时间 (W)
average_system_time = total_system_time / num_patients;

% 计算 Pw（至少有一个请求正在排队等待服务的概率）
Pw = num_requests_with_queue / (total_simulation_time + 1);

% 存储结果
results(i).checkpoint = i;
results(i).rho = rho;
results(i).average_queue_length = average_queue_length;
results(i).average_wait_time = average_wait_time;
results(i).average_system_time = average_system_time;
results(i).Pw = Pw;

% 绘制每个时间点的队伍长度
figure;
hours_time = (0:total_simulation_time) / 60; % 将分钟转换为小时
plot(hours_time, len_time);
xlabel('时间 (小时)');
ylabel('队伍长度');
title(sprintf('检查点 %d 的队伍长度', i));

% 显示每个时间点的队伍长度
disp(['检查点 ', num2str(i), ' 每个时间点的队伍长度：']);
for p = 1:length(len_time)
fprintf('时间 %d 分钟时的队伍长度：%d\n', p - 1, len_time(p));
end
```

```
% 输出每种患者类型的平均等待时间
disp(['检查点 ', num2str(i), ' 每种患者类型的平均等待时间：']);
kp = 0;
for f = 1:3
if tys(i, f) ~= 0
fprintf('%d 类型患者的平均等待时间为 %.2f 分钟，共有 %d 人\n', f, ty_time(i,
f) / tys(i, f), tys(i, f));
kp = kp + tys(i, f);
end
end
fprintf('计算总人口为 %d\n', kp);
end

% 输出每个检查点的结果
for i = 1:num_checkpoints
fprintf('\n 检查点 %d 结果:\n', i);
fprintf('系统利用率 (ρ): %.2f\n', results(i).rho);
fprintf('平均队列长度 (Lq): %.2f\n', results(i).average_queue_length);
fprintf('平均等待时间 (Wq): %.2f 分钟\n', results(i).average_wait_time);
fprintf('平均系统时间 (W): %.2f 分钟\n', results(i).average_system_time);
fprintf('至少有一个请求正在排队等待服务的概率 (Pw): %.2f\n', results(i).Pw);
end


close all;
clear;

% 参数设置
num_patients = 300; % 总患者数量
num_checkpoints = 2; % 检查点数量
initial_services = [2, 1]; % 初始服务台数量，分别对应检查点 1 和检查点 2
peak_services = [3, 2]; % 高峰时段服务台数量，分别对应检查点 1 和检查点 2
average_service_times = [10, 15]; % 每个检查点的平均服务时间（分钟），分别对应
检查点 1 和检查点 2
min_service_time = 1; % 最小服务时间（分钟）
tys = zeros(num_checkpoints, 4); % 每个检查点的类型人数
ty_time = zeros(num_checkpoints, 4); % 每个检查点的类型等待时间

% 阈值时间设置
threshold_times = [30, 45, 60]; % 每种患者类型的阈值时间（分钟）

% 随机数生成器初始化
rng('shuffle'); % 初始化随机数生成器
```

```
% 生成随机患者到达时间和服务时间
arrival_times = zeros(1, num_patients); % 存储患者到达时间
service_times = zeros(num_patients, num_checkpoints); % 存储每个患者的服务时间
patient_types = zeros(1, num_patients); % 存储患者类型
checkpoints_assigned = zeros(1, num_patients); % 存储每个患者分配的检查点

prev_time = 0; % 上一个患者的到达时间
for i = 1:num_patients
arrival_interval = exprnd(60); % 生成下一个到达间隔时间（分钟）
prev_time = mod(prev_time + ceil(arrival_interval), 24 * 60); % 更新并确保到达时间在一天内
arrival_times(i) = prev_time;

% 生成服务时间
service_dist = average_service_times(mod(i-1, num_checkpoints) + 1); % 根据检查点的平均服务时间生成服务时间
raw_service_time = max(ceil(exprnd(service_dist)), min_service_time); % 确保服务时间不小于最小值
service_times(i, mod(i-1, num_checkpoints) + 1) = raw_service_time;

% 随机生成患者类型和检查点分配
patient_types(i) = randi([1, 3]); % 生成患者类型（1 到 3）
checkpoints_assigned(i) = randi([1, num_checkpoints]); % 随机选择检查点（从 1 开始）
end

% 初始化每个检查点的队列
queues = cell(1, num_checkpoints);
for i = 1:num_checkpoints
queues{i} = [];
end

% 根据检查点分配将患者分配到对应的队列
for i = 1:num_patients
checkpoint = checkpoints_assigned(i); % 当前患者的检查点
threshold_time = threshold_times(patient_types(i)); % 根据患者类型获取阈值时间
deadline = arrival_times(i) + threshold_time; % 计算截止时间
queues{checkpoint} = [queues{checkpoint}; arrival_times(i),
patient_types(i), service_times(i, checkpoint), deadline]; % 添加患者到对应检查点的队列
end
```

```
% 对每个检查点的队列按到达时间排序
for i = 1:num_checkpoints
queues{i} = sortrows(queues{i}, [4, 1, 2]); % 按到达时间排序，然后按优先级排序
end

% 初始化变量
results = struct(); % 用于存储每个检查点的结果
total_simulation_time = 26 * 60; % 总模拟时间（分钟）

for i = 1:num_checkpoints
% 初始化变量
total_busy_time = 0; % 总服务台忙碌时间
total_wait_time = 0; % 总等待时间
total_system_time = 0; % 总系统时间
queue_wait_times = []; % 记录每个请求的等待时间
num_requests_with_queue = 0; % 有请求排队的总次数
len_time = zeros(1, total_simulation_time + 1); % 对应时间的队伍长度
kongzt = zeros(1, initial_services(i)); % 控制台空闲下来的时间
queue_len = []; % 等待队列

for t = 0:total_simulation_time
% 把到达的人加入等待队伍
arrival_indices = find(queues{i}(:, 1) == t);
if ~isempty(arrival_indices)
queue_len = [queue_len; queues{i}(arrival_indices, :)];
end

% 给入队的人排序
if ~isempty(queue_len)
queue_len = sortrows(queue_len, [2, 1]); % 按阈值时间排序，按优先级排序，然后
按到达时间排序
end
len = size(queue_len, 1);

% 高峰时间调整
if len >= 10 % 如果队列长度大于等于 10
if length(kongzt) < peak_services(i)
kongzt = [kongzt, zeros(1, peak_services(i) - length(kongzt))]; % 增加服务台
数到高峰值
end
else
if length(kongzt) > initial_services(i)
```

```
kongzt = kongzt(1:initial_services(i)); % 如果服务台数多于初始服务台数，调整为初始服务台数
end
end

% 寻找空闲服务台
if ~isempty(queue_len)
for p = 1:length(kongzt)
if kongzt(p) <= t
if size(queue_len, 1) > 0
a = queue_len(1, :);
queue_len(1, :) = [];
kongzt(p) = t + a(3);
total_wait_time = total_wait_time + (t - a(1));
total_system_time = total_system_time + (t - a(1) + a(3));
total_busy_time = total_busy_time + a(3);
queue_wait_times = [queue_wait_times, t - a(1)];
tys(i, a(2)) = tys(i, a(2)) + 1;
ty_time(i, a(2)) = ty_time(i, a(2)) + (t - a(1));
end
end
end
end

% 记录队伍长度
len_time(t + 1) = size(queue_len, 1);

% 记录队列中有请求的时间
if size(queue_len, 1) > 0
num_requests_with_queue = num_requests_with_queue + 1;
end
end

% 计算系统利用率 (ρ)
total_service_time = total_busy_time; % 总服务时间等于总忙碌时间
rho = total_service_time / (total_simulation_time * initial_services(i)); % 系统利用率

% 计算平均队列长度 (Lq)
average_queue_length = mean(len_time); % 平均队列长度为所有时间点队伍长度的均值

% 计算平均等待时间 (Wq)
```

```
average_wait_time = total_wait_time / num_patients; % 平均等待时间为总等待时间 / 患者总数

% 计算平均系统时间 (W)
average_system_time = total_system_time / num_patients; % 平均系统时间为总系统时间 / 患者总数

% 计算 Pw（至少有一个请求正在排队等待服务的概率）
Pw = num_requests_with_queue / (total_simulation_time + 1); % 计算概率，分子为有请求排队的时间点数，分母为总时间点数

% 存储结果
results(i).checkpoint = i;
results(i).rho = rho;
results(i).average_queue_length = average_queue_length;
results(i).average_wait_time = average_wait_time;
results(i).average_system_time = average_system_time;
results(i).Pw = Pw;

% 绘制每个时间点的队伍长度
figure;
hours_time = (0:total_simulation_time) / 60; % 将分钟转换为小时
plot(hours_time, len_time);
xlabel('时间 (小时)');
ylabel('队伍长度');
title(sprintf('检查点 %d 的队伍长度', i));

% 显示每个时间点的队伍长度
disp(['检查点 ', num2str(i), ' 每个时间点的队伍长度：']);
for p = 1:length(len_time)
fprintf('时间 %d 分钟时的队伍长度: %d\n', p - 1, len_time(p));
end

% 输出每种患者类型的平均等待时间
disp(['检查点 ', num2str(i), ' 每种患者类型的平均等待时间：']);
kp = 0;
for f = 1:4
if tys(i, f) > 0
fprintf('类型 %d: %.2f 分钟\n', f, ty_time(i, f) / tys(i, f));
end
end
end

% 显示所有检查点的结果
```

```
for i = 1:num_checkpoints
fprintf('\n 检查点 %d 结果：\n', i);
fprintf('系统利用率 (ρ)：%.2f\n', results(i).rho);
fprintf('平均队列长度 (Lq)：%.2f\n', results(i).average_queue_length);
fprintf('平均等待时间 (Wq)：%.2f 分钟\n', results(i).average_wait_time);
fprintf('平均系统时间 (W)：%.2f 分钟\n', results(i).average_system_time);
fprintf('有请求排队等待服务的概率 (Pw)：%.2f\n', results(i).Pw);
end
```

8.2.4 问题四代码

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint
from matplotlib import rcParams

# 设置中文字体
rcParams['font.sans-serif'] = ['SimHei']  # 用黑体显示中文
rcParams['axes.unicode_minus'] = False   # 解决负号显示问题

# 定义参数
R0 = 100  # 总收益
G0 = 50   # 总投资
K1 = 0.5  # 投资比例
K2 = 0.5  # 投资比例
R1 = 70   # 公立医院的收益（不合作）
R2 = 70   # 私立医院的收益（不合作）
G1 = 50   # 公立医院的投资（不合作）
G2 = 50   # 私立医院的投资（不合作）
g = 10    # 单方合作的额外收益

# 计算收益
def compute_returns(lambda_):
    # 总收益
    R = K1 * (R0 - G0) + K2 * (R0 - G0)

    # 一方不合作时的收益
    N = R1 - G1 + g

    # 另一方合作时的收益
    H = K2 * (R0 - G0) - g

    # 一方不合作时的收益（都不合作的收益）
```

```
    M = R1 - G1

    # 合作策略的期望收益
    E_c = lambda_ * R + (1 - lambda_) * H

    # 不合作策略的期望收益
    E_n = lambda_ * N + (1 - lambda_) * M

    # 期望收益差异
    diff = E_c - E_n

    return E_c, E_n, diff

# 微分方程
def d_lambda_dt(lambda_, t):
    # 计算期望收益
    E_c, E_n, diff = compute_returns(lambda_)
    return lambda_ * (1 - lambda_) * diff

# 初始值
lambda_initial = 0.5
t = np.linspace(0, 10, 100)  # 时间范围

# 解微分方程
lambda_t = odeint(d_lambda_dt, lambda_initial, t)

# 绘图
plt.figure(figsize=(10, 6))
plt.plot(t, lambda_t, label='合作概率 λ(t)')
plt.xlabel('时间 t')
plt.ylabel('合作概率 λ')
plt.title('合作策略的动态变化')
plt.legend()
plt.grid(True)
plt.show()
```