基于 SEIRD 和改进 SARIMA 的流感影响状况预测

摘要

本文主要研究流感的传播问题，通过建立 SEIRD 模型预测流感传播情况，通过建立改进的 SARIMA 模型预测受流感影响的海外旅游人数变化，通过建立经济复苏预测模型分析流感对经济发展的影响。

针对问题一，本文建立了 **SEIRD 模型**对流感传播进行预测与分析。首先根据模型基本假设，将人群分为易染者、暴露者、已感染者、治愈者、死亡者五类，依照人群构造与转移关系建立微分方程，利用**最小化损失函数**估计模型参数，预测了 200 天内五类人群的变化情况。然后，联系实际分析模型应用所存在的困难。最后通过分析不同隔离条件下感染人群和新增病例的变化情况并参考 **“压平曲线”**，验证了在流感早期采取隔离措施可以**减少峰值确诊人数**并**延缓疫情高峰期到来**的结论正确性。

针对问题二，本文建立了**改进的 SARIMA 模型**对海外旅游人数进行预测。首先对附件二时间序列数据进行分析，发现数据存在长期趋势、季节性及外部因素影响，经 **ADF 检验**得到序列并不平稳，采用 **Box-Cox 变换**将序列平稳化。然后考虑季节性、流感影响在模型中嵌入**外部回归器**，并通过 ACF 和 PACF 确定 SARIMA 模型参数。最后预测得到 2023 年 9 月至 12 月接待海外旅游的人数分别为 **32.7891 万人**、**33.5161 万人**、**30.5 万人**、**24.6707 万人**。

针对问题三，本文以 COVID-19 为例建立**经济复苏预测模型**，并对流感情况下民航经济变化情况进行分析。首先选取 CPI、进口额、出口额、民航客运量等宏观经济指标，通过**模糊数学模型**分析得到 SARS、H1N1 流感模式与 COVID-19 的**相似度**，分别为 **0.7442**、**0.7071**。然后利用**主成分分析法**评估 COVID-19 疫情的综合经济复苏指标为**-0.1746**，判断其对经济的综合影响为**第四级**，经济复苏较缓慢，有望在半年内恢复正常。最后基于 2003 年 SARS 数据利用**指数平滑法**分别预测中国 COVID-19 期间和附件二某地区 2023 年流感期间民航经济变化趋势和复苏情况。

本文的亮点有：1. 本文引入外部回归变量改进 SARIMA 模型，针对小数据集进行更准确的时间序列预测；2. 本文基于指数平滑法的经济复苏模型，全面系统地分析了流感对民航经济的影响。

关键词：SEIRD　Box-Cox　SARIMA　指数平滑法　经济复苏预测

目录

1 问题重述

1.1 问题背景

流感全称是流行性感冒，是由甲、乙、丙三型流感病毒分别引起的急性呼吸道传染病。大多常以季节性流行形式出现，有时甚至可以引起世界性流感大流行。我国平均每年有 8.4 至 9.2 万呼吸道疾病死亡病例与流感相关，在呼吸系统疾病死亡病例总数中占比 8.2%. 全国流感造成年均经济负担为 263.81 亿元，占 2022 年国内生产总值 0.0218%. 虽然我国已建立全国范围内的流感检测系统，但仍存在通报流感病例延迟时间长、信息上报效率低、预警方法有效性低、运行成本高等问题。若能提前预测流感趋势，可以有效控制疾病传播，大幅减少经济损失。[1]

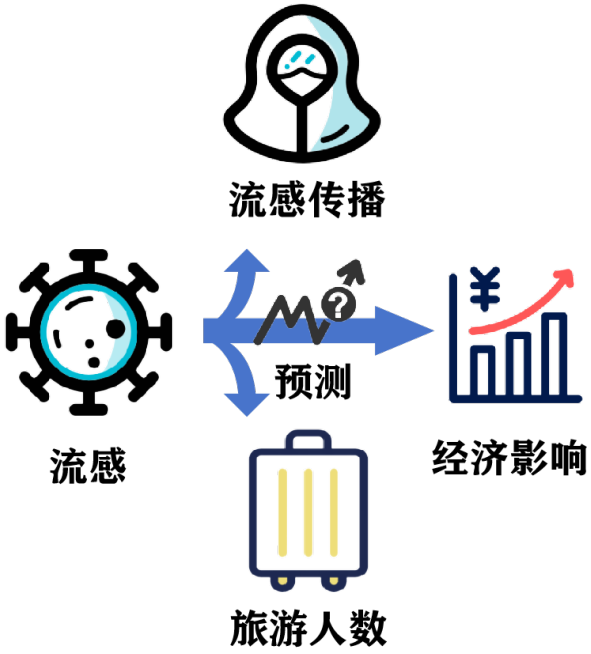

图 1　问题背景图

1.2 问题提出

问题一：参考附件一数据，要求建立数学模型，对流感的传播进行预测，并且说明利用该模型对流感的控制与预防给予充足可靠的信息的困难。利用该模型估计在流感提前或延迟 5 天采取隔离措施的条件下流感传播造成的影响。

问题二：基于附件二数据，对某地 2023 年 9 月至 12 月借贷海外旅游的人数进行预测。

问题三：参考附件二数据，并基于搜集到的流感对经济影响某个方面的数据，要求建立数学模型，对流感对于经济发展的影响进行分析和预测。

2 模型假设

(1) 假设附件一二中的数据都真实可靠;

(2) 假设问题一中的 SEIRD 模型的基本假设都成立;

（3）假设附件二中 2023 年的 2023 年旅游人数的数据异常变化是由流感引起的;

(4) 假设附件二中每月海外旅游人数可作为民航客运量，且民航客运量能衡量民航经济情况。

3 符号说明

符号	说明
S	易感染者
E	疑似已感染者
R	移出者
D	死亡者
I	已感染者
β	感染率
σ	确诊率
μ	致死率
γ	治愈率
X_t	外部回归变量
p	自回归阶数
q	移动平均阶数

注：表中未列出及重复的符号均以首次出现处为准。

4 问题一模型的建立与求解

4.1 问题一分析

问题一主要分为两个小问，第一小问要求建立流感的传染病模型并对模型应用困难进行说明，第二小问要求研究不同隔离条件下的流感传播情况。

对于第一小问，我们将人群分为暴露者、已感染者、移出者、死亡者五类，根据人群构造与转移关系构建 **SEIRD 模型**，并根据**最小损失函数**求取模型中的感染率、确诊率、致死率、治愈率四个参数。在模型建立过程中，由于考虑的外部因素较少，可能在实际应用过程中会面临困难，因此需要根据实际情况，适时调整模型假设与参数。

对于第二小问，我们分为不干预、提前 5 天隔离、正常隔离、延后 5 天隔离四类研究，通过对比**不同隔离条件下的感染人数数量、每日新增病例变化分析**对流感传播造成的影响，最后通过查阅资料用**压平曲线**说明结果的可靠性。

问题一思路图如下。

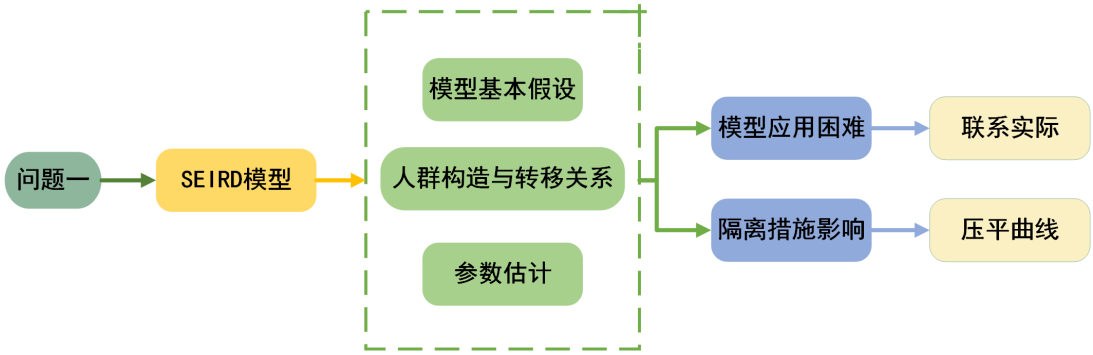

图 2　问题一思路图

4.2 基于 SEIRD 的流感传播模型

问题的核心是研究流感的传播动态，本文决定采用传染病模型来对流感的传播进行预测。SIR 模型是一个经典的传染病动力学模型, 常用于模拟某一疫区的传染病传播过程, 能够有效合理地预测疾病的传播和扩散趋势。本文对传统的 SIR 模型做出改进，在原模型的基础上，增加 D 人群即死亡人群和 E 人群即暴露者人群，建立 SEIRD 模型[1] [8]来预测流感的传播态势。

4.2.1 模型基本假设

为了模型的简洁，本文对该传染病模型做如下假设：

(1) 由于一次流感的周期较短，假设所考察地区的总人数 (包括死亡人数) 不变为 10000 人，其他疾病与自然出生、死亡的影响为零。总人群仅由流感的易感染者 (suscep-

tible)、暴露者 (exposed)、已感染者 (infective)、移出者 (removed)、死亡者 (dead) 五类人群构成。

(2) 假设移出者存在免疫性即不会再次感染流感，并且无传染性即不会再将流感传染给他人；

(3) 假设忽略治疗过程中治愈的时间延迟 τ_r 和死亡的时间延迟 τ_d；

(4) 假设暴露者都已与易感染者接触且成功被感染，但当前处于潜伏期还未被确诊，最终均会被确诊。

4.2.2 人群构造与转移关系

基于上述假设并联系实际，本文用下图说明各个人群的构造与转移关系

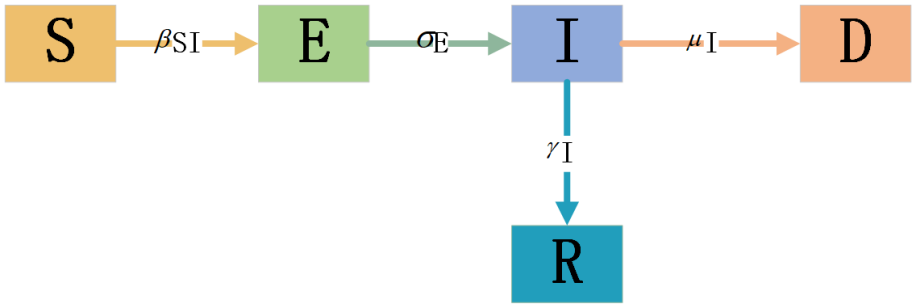

图 3　人群构造与转移关系图

上述变量的具体表示含义如下

1)N 表示总人数，即始终满足 $N = S(t) + E(t) + I(t) + R(t) + D(t)$；

2)$S(t)$ 表示易感染者，即未染病的人群，无免疫性。初始未染病的人群数量可通过总人群数减去其他四类人群数求得；

3)$E(t)$ 表示暴露着，即即附件一中的疑似病例人群；

4)$I(t)$ 表示已感染者，即附件一中的确诊病例人群，有强传染性；

5)$R(t)$ 表示移出者，即附件一中的治愈出院人群，有免疫性，无传染性；

6)$D(t)$ 表示死亡者，即附件一中的死亡人群；

7)β 表示感染率，即易感染者被感染成为已感染者的概率；

8)σ 表示确诊率，即暴露者被确诊为已感染者的概率；

9)μ 表示致死率，即已感染者未被成功治愈最终死亡的概率；

10)γ 表示治愈率，即已感染者被成功治愈的概率。

4.2.3 转移关系方程

根据上述人群构造与转移关系, 可将 SEIRD 模型表示为如下方程形式。

$$\begin{cases} N = S(t) + E(t) + I(t) + R(t) + D(t) \\ \frac{dS}{dt} = -\frac{\beta \times I(t) \times S(t)}{N} \\ \frac{dE}{dt} = -\sigma \times E\left(t\right) + \frac{\beta \times I(t) \times S(t)}{N} \\ \frac{dI}{dt} = \sigma \times E\left(t\right) - \gamma \times I\left(t\right) - \mu \times I\left(t\right) \\ \frac{dR}{dt} = \gamma \times I\left(t\right) \\ \frac{dD}{dt} = \mu \times I\left(t\right) \end{cases} \tag{1}$$

4.2.4 参数估计

基于附件一数据，本文利用最小化损失函数来估计上述模型中的感染率、确诊率、致死率、治愈率四个参数。损失函数为 SEIRD 模型预测值与实际值之间的差异的平方和表示如下。

$$L(\beta, \sigma, \gamma, \mu) = \sum_{t=1}^{T} \left(\text{SEIRD}(\beta, \sigma, \gamma, \mu) - y_t^{data}\right)^2 \tag{2}$$
$$(\beta, \sigma, \gamma, \mu \in [0, 1])$$

其中 T 表示附件一时间跨度即 64 天；y_t 表示第 t 天易感染者、暴露者、已感染者、移出者、死亡者人群人数形成的组合矩阵；SEIRD$(y_t, \beta, \sigma, \gamma, \mu)$ 表示第 t 天使用参数 $\beta, \sigma, \gamma, \mu$ 的 SEIRD 模型计算得到的易感染者、暴露者、已感染者、移出者、死亡者人群五类人数的预测值；y_t^{data} 表示该五类人群的实际数量。

通过使用 MATLAB 中的'scipy.optimiza' 库中的'minimize' 函数，并运用'L-BFGS-B' 方法进行优化, 求得模型的各个参数的最优取值如下表所示。

表 1　参数估计结果

参数	感染率 β	确诊率 σ	致死率 μ	治愈率 γ
取值	0.6502	0.3608	0.0228	0.0060

4.3 结果分析

经过上述模型求解，得到五类不同人群累计数量变化情况如下图所示。

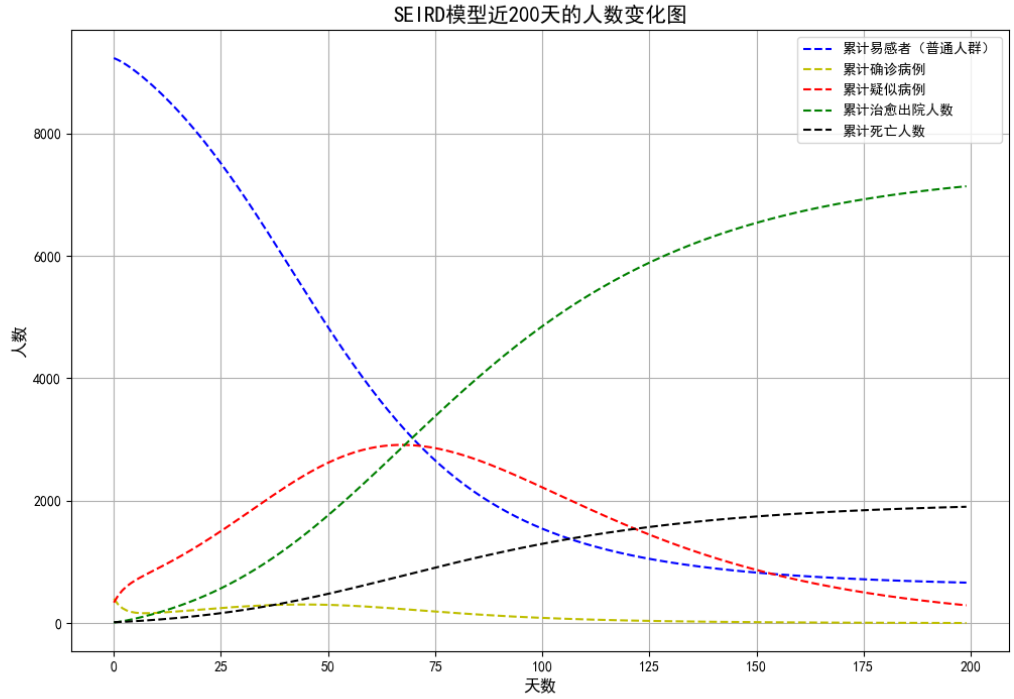

图 4　各类人群累计数量的变化情况

从图中可以看出，在最初流感传播较快，感染人数迅速上升；后随着时间的推移，流感逐渐稳定，感染人数的增长率逐渐减小；最终几乎所有人都被感染，少数人死亡，大部分人被治愈出院。通过上述模型，我们可以预测流感的传播情况，从而及时做出防控政策，减小流感规模，使损失最小化。

4.4 模型应用困难

由于为了模型的简洁性，在模型的建立过程中忽略了很多外界因素，导致在实际生活的流感的预防和控制中应用该模型存在一些困难。如随着时间的推移，人群可能会发展出对流感产生抗异性或免疫，这在模型中可能难以体现。除此之外，流感病毒可能会发生变异，导致新的流感毒株出现，多种流感同时发生，相互影响，若出现此种情况，难以用该单一的传染病模型求解。再者，模型虽然考虑了部分干预行为如隔离措施，但在实际中，如何、何时、在何地实施着这些举措都会对模型预测产生不同的影响，因素错综复杂，不易研究。尽管在应用过程中存在很多困难，但建立的 SEIRD 模型仍然是帮助预防和控制传染病疾病的有效工具。我们可以根据实际情况，适时调整模型假设与参数，限制实际情况中的不确定性，灵活解释与应用模型。

4.5 估计隔离措施对流感传播的影响

在遏制流感传播的过程中，隔离措施的实行十分重要。为排除外界因素干扰，本文的隔离方式仅考虑居家隔离。本文利用上述模型针对流感提前或者延后 5 天采取隔离措施，对流感传播造成的影响做出估计，具体分析隔离措施对流感传播的影响。

参考新冠和非典疫情数据，设采取隔离措施可将感染率降为原先的 50%。以流感传播第 10 天作为采取隔离措施的节点，并取其前后 5 天作为提前或者延后的时间节点。为反映流感的传播情况，本文统计流感初始传播 200 天的人群感染情况和每天的新增病例情况，其中新增病例表示确诊病例与累计出院和死亡人数的差值。图中黄色曲线表示没有任何隔离措施干预的情况，红色曲线表示提前 5 天采取隔离措施的情况，绿色曲线表示延后 5 天采取隔离措施的情况，蓝色曲线表示正常即第 10 天采取隔离措施的情况。

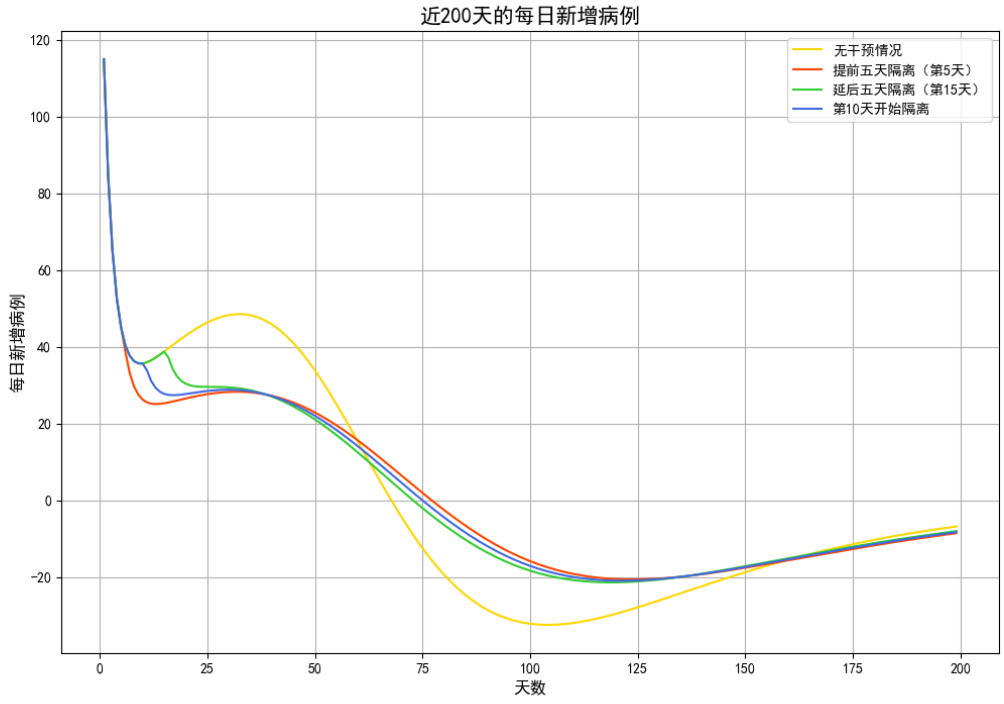

图 5　不同隔离条件下每日新增病例的变化情况

从图中可以看出在隔离开始前，日新增感染者数量基本相同。然而在隔离初期，日新增感染者数量仍然较高。这可能是因为流感的潜伏期，导致疑似已感染者在隔离开始后转化为感染者，从而使初期新增感染者数量激增。此外，也可能是虽然通过隔离使感染率 β 减小，但由于感染者数量规模加大，感染率可能还没有减少到足够低的水平来立即减缓流感的传播。隔离措施可能需要更长时间才能显现其效果。并且，根据图分析可知，及时的隔离措施可以显著减少日新增感染者的数量，尤其是在隔离开始后的几天。

延后隔离措施虽然仍然有效，但效果不如提前采取措施。

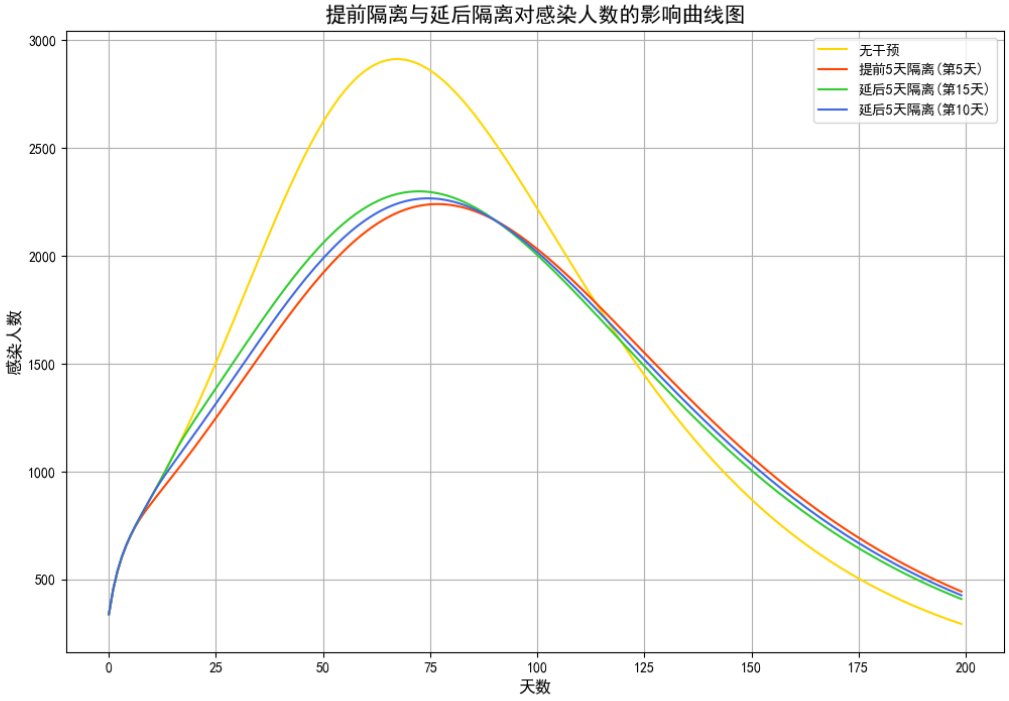

图 6　不同隔离条件下感染人群累计数量的变化情况

表 2　不同隔离条件下感染人数峰值变化情况

措施	感染人数峰值	峰值出现时间
无干预	2913	67
第 5 天隔离	2240	77
第 10 天隔离	2267	75
第 15 天隔离	2299	72

经纵向对比四条曲线和不同隔离条件下感染人数峰值变化情况，可以看出无干预情况下的峰值最大且最早出现。随着隔离开始时间的延后，感染人数峰值逐渐提高，峰值出现时间逐渐提前。并且发现只要有实行隔离措施，都可以有效减少感染者人数。从图 b 中可以看出及时的隔离措施可以显著减少日新增感染者的数量，尤其是在隔离开始后的几天。延后隔离措施虽然仍然有效，但效果不如提前采取措施。

以上结果充分说明了及时采取隔离措施对于控制疾病传播的重要性。经查找资料发

现，隔离措施是一种行政强制措施，具有法律与医学双重属性，《突发公共卫生事件应急条例》《传染病防治法》和《突发事件应对法》等法律法规都对隔离措施作出了详细规定，由此可见隔离措施对于控制疾病传播具有至关重要的意义。同时，美国 CDC 顾问 Drew Harris 改编的“**压平曲线**”也可以证明隔离措施对流感传播的影响，即在流感早期采取隔离措施可以延缓疫情高峰期的到来并减少峰值确诊人数，进而验证了本题结论的正确性。

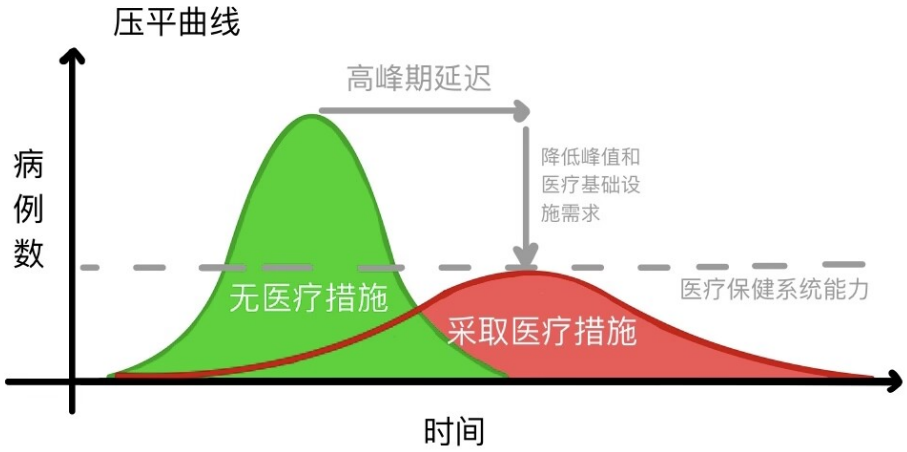

图 7　压平曲线

5 问题二模型的建立与求解

5.1 问题二分析

问题二要求根据 2017 年 1 月至 2023 年 8 月得数据，对 2023 年 9 月至 12 月接待海外旅游的人数进行预测。

首先，我们对附件二已知数据进行分析，发现关于该地接待海外旅游的人数得时间序列呈现强季节性，且 2023 年数据明显受外界干扰，本文猜测是流感原因。然后，我们对数据进行 **ADF 检验**，也说明了序列具有不稳定性，因此我们进行 **Box-Cox 变换**对数据平稳化。接着，由于考虑数据受外部因素影响，我们引入是否属于旺季、是否属于淡季、是否处于流感爆发影响期，是否处于流感结束恢复期四个外部回归变量，将**外部回归回归器**嵌入模型中。之后，建立 **SARIMA 模型并**通过 ACF 和 PACF 求取模型参数。最后通过建立得 SARIMA 模型对海外旅游的人数进行预测。

问题二思路图如下。

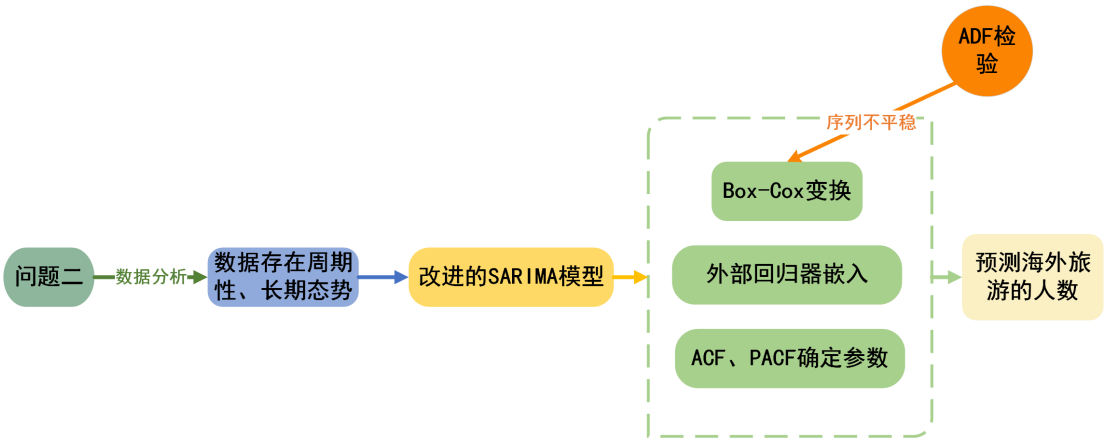

图 8　问题二思路图

5.2 基于改进的 SARIMA 预测模型

5.2.1 数据分析

根据附件二数据，观察 2017-2023 年各月份海外旅游人变化情况，绘制折线图如下。

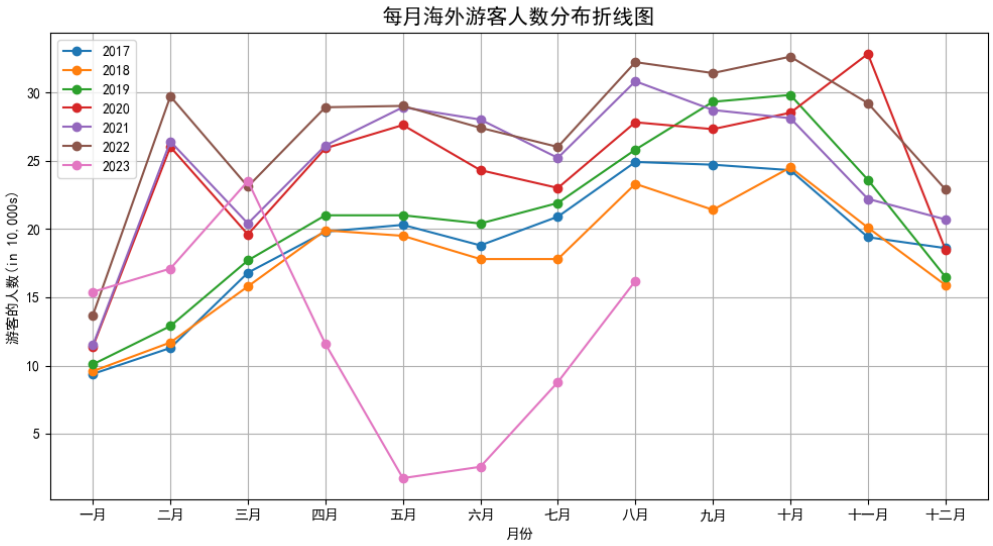

图 9　2017-2023 年各月份海外旅游人变化折线图

从横向上可以看出，2017-2022 年海外旅游人数均呈现长期趋势和季节性波动。8 月至 10 月是旅游旺季，而后可能由于天气寒冷，旅游人数开始跌落；11 月至次年 1 月是旅游淡季，之后可能由于天气回春，旅游人数又开始回升。但 2023 年旅游人数在 2 月至 4 月份骤降，这可能是由于流感如新冠疫情的负反馈影响，为阻止流感传播人们被居家隔离，无法出门到海外旅游。5 月份过后旅游人数急剧回升，这可能是隔离解除，人们外出不再受限制，出游欲望强烈，体现了疫情的正反馈作用。因此本文初步预测 2023

年 8 月份过后人数会呈现上升趋势。

从纵向上看，海外旅游人数逐年上升，初步预测 2023 年的旅游人数回升后，达到的稳态人数会大于前面年份。

5.2.2 ADF 检验

ADF 检验是一种常用的序列平稳性检验方法，通过判断一个时间序列是否具有单位根来检验是否平稳。ADF 检验的原假设是时间序列具有单位根即非平稳性，备择假设是时间序列不具有单位根即平稳性。对附件二进行一阶 12 步差分序列的平稳性检验，结果如下。

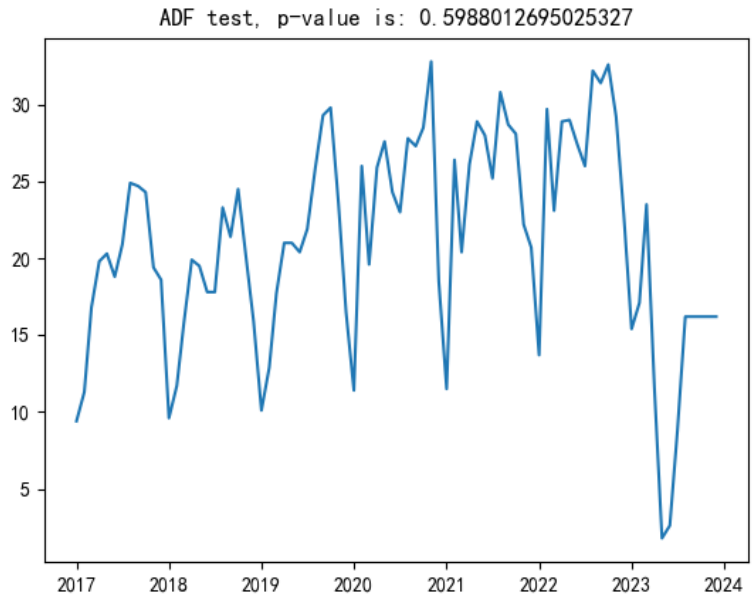

图 10　ADF 检验

ADF 的统计量为-11.4243，p 值为 0.5988，在 1% 的显著性水平下满足了存在单位根的原假设, 说明该时间序列不是平稳的，也与我们前面对数据的初始分析结果相符。

5.2.3 Box-Cox 变换

经过 ADF 检验时间序列并不平稳，为后续应用 SARIMA 模型，本文决定先通过 Box-Cox 变换对序列进行平稳化处理。

Box-Cox 变换[2]是一种处理数据的常用统计方法，可以让数据更加具有正态性，从而调整数据的分布形态，是数据更加平稳化。Box-Cox 变换公式表示如下：

$$N(\lambda) = \begin{cases} \frac{N^{\lambda}-1}{\lambda}, & \text{if } \lambda \neq 0; \\ \ln N, & \text{if } \lambda = 0. \end{cases} \tag{3}$$

其中 N 表示附件中每月份海外旅游人数的原始数据；$N(\lambda)$ 表示经过 Box-Cox 变换后的数据；λ 表示变换参数，本文通过最大似然估计法确定 λ 的取值为 1.34867.

5.2.4 外部回归器嵌入

为了提高时间序列模型的预测准确性和解释能力，考虑外部因素对时间序列的影响，使得建模更贴近实际情况，本文在 SARIMA 模型嵌入中外部回归器[4] [7]。

关于旅游人数的时间序列存在旅游旺淡季季节性影响、流感外部因素影响。基于上述数据分析，每年 8 月至 10 月为旅游旺季，每年 11 月至次年 1 月为旅游淡季，2023 年 2 月至 4 月为受流感影响月份，2023 年 5 月至 8 月为流感消失恢复月份。本文引入四个外部回归变量，分别是相应的月份是否属于旺季、是否属于淡季、是否处于流感爆发影响期，是否处于流感结束恢复期：

$$X_t = \left\{X_{(1,t)}, X_{(2,t)}, ..., X_{(c,t)}\right\}. \tag{4}$$

其中 c 表示外部回归变量个数及 4，四个变量均为布尔变量，当取值为 1 时表示是，当取值为 0 时表示否。

外部回归变量的系数是通过训练以下 SARIMAX 模型并通过最大似然估计求得，其系数表示如下：

$$\beta_i = \{\beta_1, \beta_2, ..., \beta_c\}. \tag{5}$$

5.2.5 SARIMA 模型

由于附件二数据既存在季节效应又含长期趋势效应，本文决定采用 SARIMA 模型(乘积季节模型)[3] [9]对该地 2023 年 9 月至 12 月接待海外旅游的人数进行预测。它是由 ARIMA 模型的基础上改进而来，通过引入季节性差分即将时间序列与其某个固定滞后期的值相减，来去除季节性变动。通过引入季节性自回归（SAR）和移动平均（SMA）的部分，同时考虑了季节性差分和非季节性差分，从而使得 SARIMA 模型能够适应具有季节性模式的数据。

首先，对海外旅游人数时间序列 $\{N_t\}$ 进行 D 次季节差分来去序列季节性，一次季节差分可表示为

$$\Delta s \cdot N_t = (1 - L^s)N_t = N_t - N_{t-s}. \tag{6}$$

其中 s 表示季节周期的长度，根据题目 s 取值为 12 个月；Δs 表示季节差分算子；L 表示滞后算子。

从而经过 D 次季节差分, 建立关于周期为 s 的 P 阶自回归 Q 阶移动平均季节时间序列模型为

$$A_P(L^s)\Delta s^D N_t = B_Q(L^s)u_t. \tag{7}$$

其中 P 表示季节性自回归阶数，$A_P(L^s)$ 表示季节自回归多项式，Q 表示季节性移动平均阶数，$B_Q(L^s)$ 表示季节移动平均多项式，u_t 表示经过季节差分处理后的时间序列。

然后，再进行 d 次非季节差分来去序列趋势性得到

$$\varphi_p(L)\Delta^d u_t = \Phi_q(L)v_t. \tag{8}$$

其中 p 表示自回归阶数，$\varphi_p(L)$ 表示非季节自回归多项式，q 表示移动平均阶数，$\Phi_q(L)$ 表示非季节移动平均多项式，v_t 是误差项，是一个高斯白噪声序列。

最后，得到差分后的序列，并嵌入外部回归器，建立如下 SARIMA 模型

$$\varphi_p\left(L\right)A_p\left(L^s\right)\left[\Delta^d\Delta s^D\left(N_t+\sum_{i=1}^{c}\beta_i X_{(i,t)}\right)\right]=\Phi_q\left(L\right)B_Q\left(L^s\right)v_t. \tag{9}$$

其一般表示式可写作

$$\text{SARIMA}\left[(p,d,p)\times(P,D,Q)_s+\sum_{i=1}^{c}\beta_i X_{(i,t)}\right]. \tag{10}$$

5.2.6 确定参数

本文通过利用 $\{N_t\}$ 时时间序列的自相关函数图 (ACF) 与偏自相关函数图确自相关图 (PACF) 确定 SARIMA 模型的移动平均阶数 q 和自回归阶数 p.

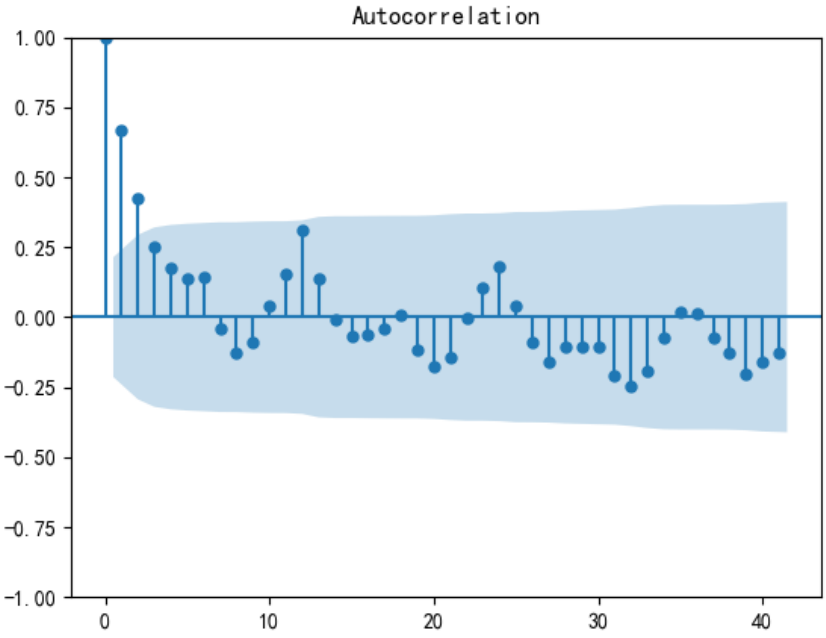

图 11　自相关图

ACF 图像是以滞后项（lag）为横轴，对应的自相关系数为纵轴的图像。每个滞后项上的自相关系数表示该滞后项与原序列在该滞后项处的相关性。若在某个滞后项处

自相关系数为正且显著不为零，表示在该滞后项处存在一个正相关的关系。若为负且显著不为零，表示在该滞后项处存在一个负相关的关系。若接近于零，说明在该滞后项处不存在明显的自相关关系。通过观察 ACF 图中滞后项超过置信区间的截尾点，可以确定 MA 模型的阶数 q. 本文将置信度设置为 95%，从 (a) 图中可以看出 ACF 在 2 阶出现截尾，因此 q 确定为 2.

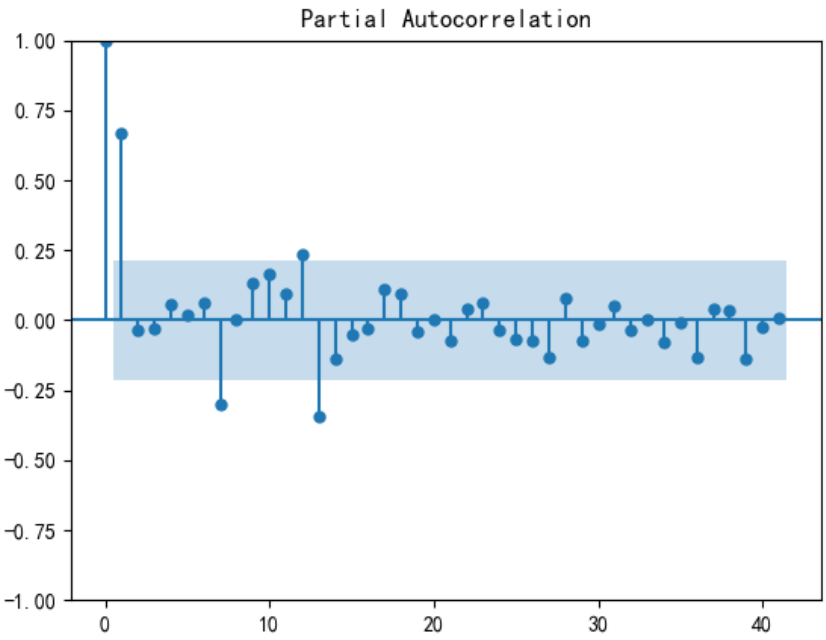

图 12　偏自相关图

PACF 图表示在不同滞后项下的偏自相关系数。每个滞后项上的偏自相关系数表示该滞后项与原序列在该滞后项处的相关性，而排除了较短滞后项的影响。在 PACF 图中，通常有一个垂直线通过滞后项为 0 的位置，表示偏自相关系数为 1，因为每个序列与其本身的相关性总是完全正相关的。若在某个滞后项处偏自相关系数为正且显著不为零，表明在该滞后项处的观察值与当前值之间存在直接的关联。若为负且显著不为零，表明在该处的观察值与当前值之间存在直接的反向关联。若接近于零，说明在该滞后项处不存在明显的偏相关关系。PACF 图可以帮助识别隐含在时间序列中的 AR 模型结构，以确定其阶数 p。从（b）图中可以看出 PACF 在 2 阶出现截尾，因此 p 也确定为 2.

由此, 将时间序列的 SARIMA 的基本模型确定为 SARIMA$(2,1,2)\times(1,1,0)_{12}$. 经计算，该模型的 AIC 和 BIC 值分别为 250.653、255.937，两者比较接近并且较小，说明使用该参数的 SARIMA 模型在拟合效果和模型复杂度方面都相对较好。

5.3 结果分析

将 2017 年 1 月至 2023 年 8 月的海外旅游人数数据作为训练集，2019 年 1 月至 2023 年 8 月作为验证集，以 2023 年 9 月至 12 月数据作为测试集，代入上述改进的 SARIMA 模型，求解结果如下图所示。其中验证集的实际观测值与预测值的均方误差为 **12.5284**，说明预测效果良好。

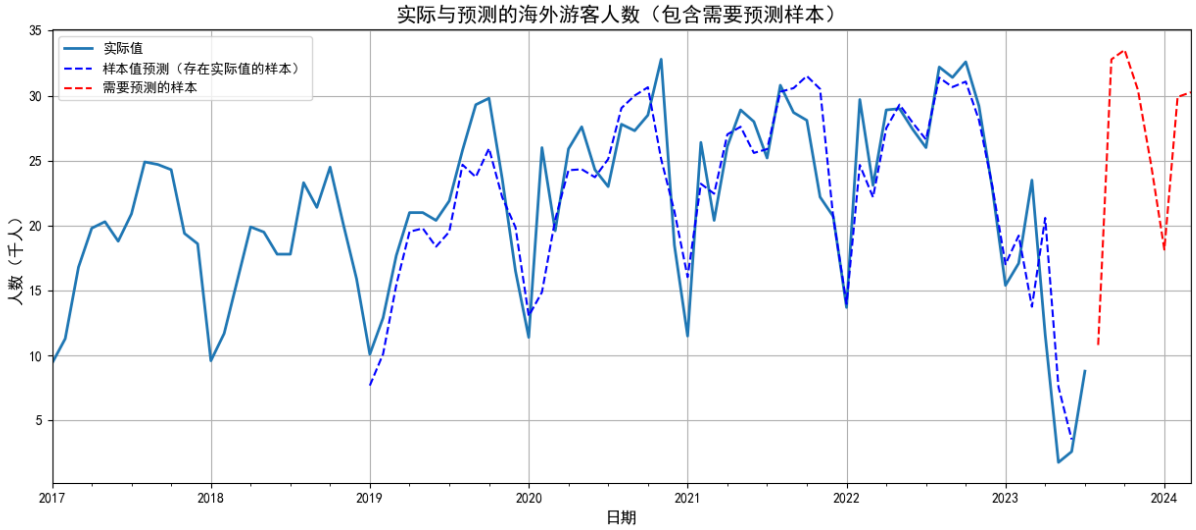

图 13　实际与预测的海外旅游人数

在初始阶段，游客人数呈现急剧上升趋势，本文猜测主要有以下三方面原因。一是流感刚刚结束，“封城”、“封村”、“封路”、关闭旅游景点、暂停娱乐场所、控制人口流动、禁止人口聚集等相关措施取消，人们出行没有了限制；二是为了解决因流感导致的经济低迷问题，多地发布旅游优惠政策促进旅游经济发展，激发了人们的消费热情；三是人们长时间处于被隔离状态，出游热情高涨。

在后续阶段，旅游人数波动也大体符合 2017 年至 2022 年变化趋势，旅游淡季游客数量较少，旅游旺季数量回升，处于正常波动状态。

综合来看，预测的 2023 年 9 月至 12 月旅游人数变化趋势符合上述数据分析的结论，预测人数如下表所示，一定程度印证了预测结果的准确性。

表 3　2023 年 9-12 月海外旅游人数预测结果表

时间	2023 年 9 月	2023 年 10 月	2023 年 11 月	2023 年 12 月
人数 (万人)	**32.7891**	**33.5161**	**30.5**	**24.6707**

6 问题三模型的建立与求解

6.1 问题三分析

问题三要求分析并预测流感对经济发展的影响。本文从民航经济的角度出发，分析在疫情影响下民航经济变化并建立**民航经济复苏预测模型**。

首先，选取美国 H1N1 和中国 SARS 两种流行病作为 COVID-19 的对照，利用**模糊数学模型**，选取 **CPI**、**进口额**、**出口额**、**民航客运量**四项宏观经济指标的变化率为特征影响因子，比较两者与 COVID-19 的相似性，选取出相似度大的流行病 SARS。然后，通过**主成分分析法**计算 COVID-19 的经济复苏程度综合指标大小，从而对经济复苏程度进行评价。最后，以 SARS 作为流感模型，用**指数平滑法**预测 COVID-19 期间中国的民航客运量和附件二中某地 2023 年流感期间的海外旅游人数，并将其作为衡量民航经济变化的指标。

问题三思路图如下。

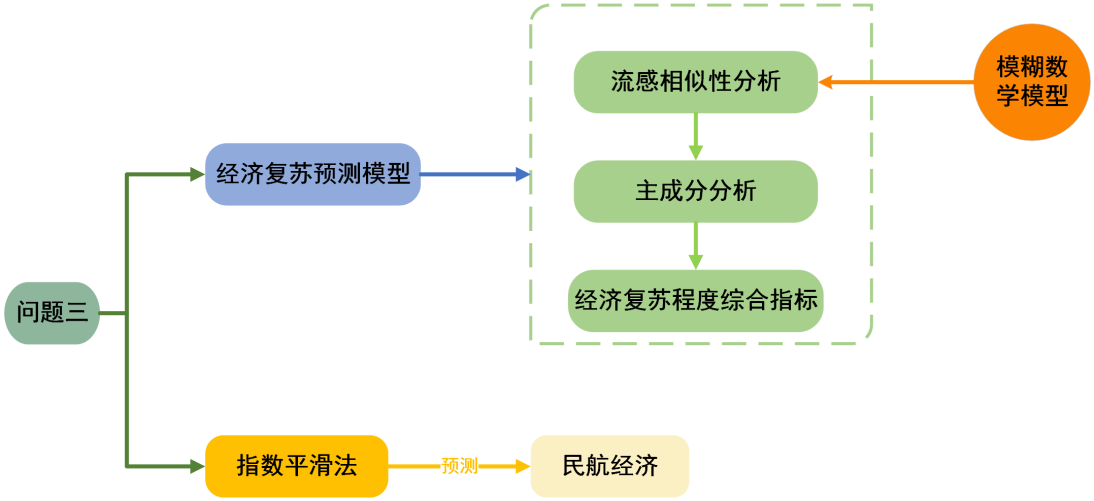

图 14　问题三思路图

6.2 经济复苏预测模型

本题要求研究流感对经济发展的影响，但流感包含的疾病众多，本文选取 COVID-19 疫情为例，根据历史流感数据，预测 2019 年 10 月至 2020 年 8 月 COVID-19 疫情期间中国经济的变化情况。本文绘制了 COVID-19 疫情期间中国客运量的变化折线图，如下所示。

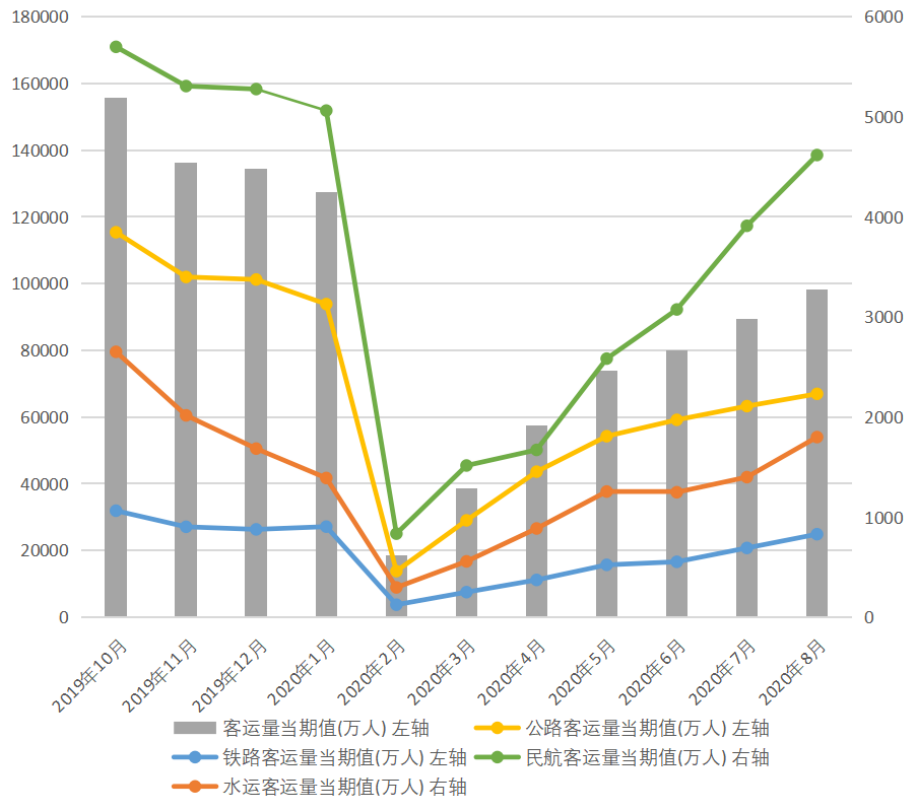

图 15　COVID-19 期间中国客运量折线图

由上图可以分析，由于 COVID-19 影响，我国采取相关措施，客运量情况在 2020 年 2 月断崖式下跌，这会使得我国客运行业的经济严重下滑，导致经济发展减慢甚至倒退。为此，本文决定对流感情况下民航经济变化情况进行分析，并对全球性流感大流行情况下的经济复苏情况进行预测[5] [10]。

6.2.1 流感相似性分析

查找相关研究，本文初步选取美国 H1N1、中国 SARS 两个历史流行病作为中国 COVID-19 疫情的对照。为选取与 COVID-19 最相似的流行病，对两者与 COVID-19 疫情的相似性进行计算。考虑到疫情间的相似性和不相似性之间无绝对明确的界限，本文将其视为模糊现象，因此选择模糊数学模型进行预测研究。具体求解步骤如下。

(1) 选取对经济复苏有显著影响的指标。本文通过研究问题二中发现疫情会影响海外旅游人数，一定程度上反映了疫情会对海外旅游经济产生影响，因此本文选取了民航客运量作为反映经济复苏的指标之一。结合有关学者研究和问题背景，本文还选取了 CPI、国家进口额、出口额三个宏观经济指标。

通过数据爬取，本文得到的 2008 年 11 月至 2009 年 1 月美国 H1N1 疫情期间和 2003 年 1 月至 2003 年 11 月中国 SARS 疫情期间的四个宏观经济指标的数据。为反映经济恢复情况，本文用指标的每个月与前一个月的数据变化率来进行衡量。

(2) 以 CPI、国家进口额、出口额、民航客运量的变化率为特征的影响因子，建立关于 H1N1、SARS 两个流行病的模糊特征矩阵如下

$$X = (X_1, X_2, ..., X_n) = \begin{pmatrix} x_{11} & \cdots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{pmatrix} \tag{11}$$

其中 m 表示影响因子数即 4，n 表示流行病事件数即 2，x_{ij} 表示其中第 i 次流行病中第 j 个特征影响因素的隶属函数值，满足 $x_{ij} \in [0, 1]$ 。

(3) 计算 H1N1、SARS 两个流行病与 COVID-19 的特征影响因素的隶属度之间欧氏距离，并用其衡量与 COVID-19 的相似度。欧氏距离越小，与 COVID-19 疫情越相似。计算公式为

$$\delta_i = 1 - \frac{1}{\sqrt{n}} \sqrt{\sum_{j=1}^{m} \left(x_{ij} - x_j^*\right)^2}, (i = 1, 2, ..., n) \tag{12}$$

其中 δ_i 表示欧氏距离，x_j^* 表示 COVID-19 的第 j 特征影响因子的隶属函数值。(4) 按升序方式对控制流行病的接近度进行排序 $(\delta_1, \delta_2, \ldots, \delta_n) \to (\delta_1^*, \delta_2^*, \ldots, \delta_n^*)$，采用指数平滑法预测流感过后的经济复苏情况，具体如下：

$$r_i^* = \delta_{i-1}^* r_{i-1} + (1 - \delta_{i-1}^*) r_{i-1}^* \quad (i = 1, 2, \ldots, n) \tag{13}$$

其中 ri 表示第 i 个流感的经济整体恢复指标，并且将三次控制流感的经济整体恢复平均值作为初始条件即

$$r_1^* = \frac{\sum_{i=1}^{n} \delta_i r_i}{\sum_{i=1}^{n} \delta_i} \tag{14}$$

最终求得的 r_n 为用 H1N1 和 SARS 预测得到的 COVID-19 恢复的总体指标。

6.2.2 主成分分析

运用多元统计分析中的主成分分析方法来评估综合经济恢复程度。选择上述四项经济指标作为 1 年期经济恢复考察的绩效因素，得到关于 COVID-19 的原始数据信息矩阵，并对矩阵数据进行标准化得

$$Y = (y_1, y_2, ..., y_n) = \begin{pmatrix} y_{11} & \cdots & y_{1m} \\ \vdots & \ddots & \vdots \\ y_{n1} & \cdots & y_{pm} \end{pmatrix} \tag{15}$$

其中 y_{ij} 表示第 i 个月份的第 j 个经济指标变化率大小，p 表示时间跨度为 10. 然后，利用如下公式计算得到协方差矩阵为 W.

$$W = (w_1, w_2, ..., w_n) = \begin{pmatrix} w_{11} & \cdots & w_{1m} \\ \vdots & \ddots & \vdots \\ w_{n1} & \cdots & w_{pm} \end{pmatrix} \tag{16}$$

之后，计算 W 的特征值与特征向量，并用其计算累计贡献率

$$\alpha_j = \frac{\sum_{k=1}^{j} \lambda_{\mathrm{k}}}{\sum_{k=1}^{m} \lambda_k} (j = 1, 2, \cdots, m) \tag{17}$$

其中 α_j 表示第 j 个指标的累计贡献率，λ_k 表示第 k 个指标的特征值。

接着，根据累计贡献率取值得到主成分，一般取累计贡献率超过 90% 的特征值对应的指标作为主成分。第 i 个主成分可表示为

$$F_i = a_{1i}Y_1 + a_{2i}Y_2 + \cdots + a_{di}Y_m \quad (i = 1, 2, \cdots, r) \tag{18}$$

其中，F_i 表示数据在第 i 主成分上的投影，$a_i = (a_{1i}, a_{2i}..., a_{mi})$ 表示第 i 个指标的特征向量,r 表示主成分个数。

经上述步骤求得前三个的累计方差贡献率达到了 96.76%, 所以选取前三个作为主成分。

最后，以方差贡献率为权重，计算加权平均值的主成分作为控制疫情综合经济复苏度。

$$V = \sum_{i=1}^{3} \omega_i \times F_i \tag{19}$$

其中 ω_i 是主成分 i 的贡献率，将其作为该主成分计算综合得分时的权重。

6.2.3 指数平滑预测

经上述流感相似性分析，得到 SARS 与 COVID-19 的相似度相比于与 SARS 的相似度更大，因此本文选取中国地区的 SARS 疫情期间的民航数据作为参照，预测 COVID-19 疫情、附件二某地区 2023 年流感分别对中国和该地区民航经济的影响。本文已经假设了 2023 年附件二数据来源地区在 2023 年遭受了某种流感影响，并且海外旅游人数可以当作当地的民航客运量来计算，且民航客运量可以看作衡量民航经济的重要指标。由于问题二对附件二该地区 2023 年 9 月至 12 月的海外旅游人数已经进行预测，我们将该结果作为该月可能的真实值。本文利用指数平滑法[6]分别对 2023 年该地区海外旅游人数和进行预测，最后将预测结果与实际结果进行对比。

6.3 结果分析

计算得到的 SRAS、H1N1 与 COVID-19 的相似度分别为 0.7442、0.7071, 可见 SRAS 与 COVID-19 更加相似，现有文献也推断，在利用模糊数学模型对发生在同一地点的流行病进行经济预测时，模型的准确性会有所提高，证明了结果的合理性。

然后，计算得到 COVID-19 疫情的综合经济复苏指标为 -0.1746。为了更准确地描述具体的经济复苏情况，本文查阅资料制定了经济复苏指标评价表，如下所示。可以判断，COVID-19 疫情对中国经济的综合影响应该是第四级，但数值结果比较接近第三级，

表明中国经济的复苏程度与预测结果还有一定误差，经济可能还需要半年甚至更长时间才能恢复正常。

表 4　经济复苏指标评价表

复苏程度	意义
一级 [-2,-1.25]	经济复苏程度问题重重，经济趋于长期低迷
二级 [-1.25,-0.75)	经济复苏程度不尽如人意，经济逐步恢复可能需要 1 年多时间
三级 [-0.75,-0.25)	经济复苏程度缓慢，但有望在 1 年内恢复正常
四级 [-0.25,0.25)	经济复苏程度较缓慢，但有望在半年内恢复正常
五级 [0.25,0.75)	经济已基本恢复到原来的水平，但仍需要后期支持工作
六级 [0.75,1.25)	经济不仅恢复正常，而且走在正常的经济发展轨道上
七级 [1.25,2)	经济复苏超出预期，灾后经济稳步增长

最后，使用 2003 年 SARS 期间的数据作为基础，来预测 COVID-19 爆发后中国和附件二流感爆发后某地区的民航经济复苏走势，如下所示。

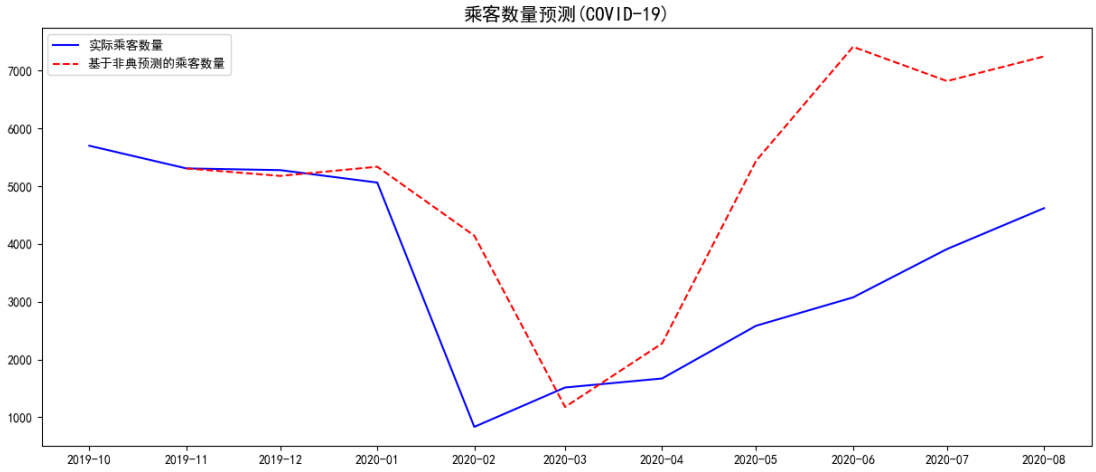

图 16　COVID-19 期间中国民航客运量预测

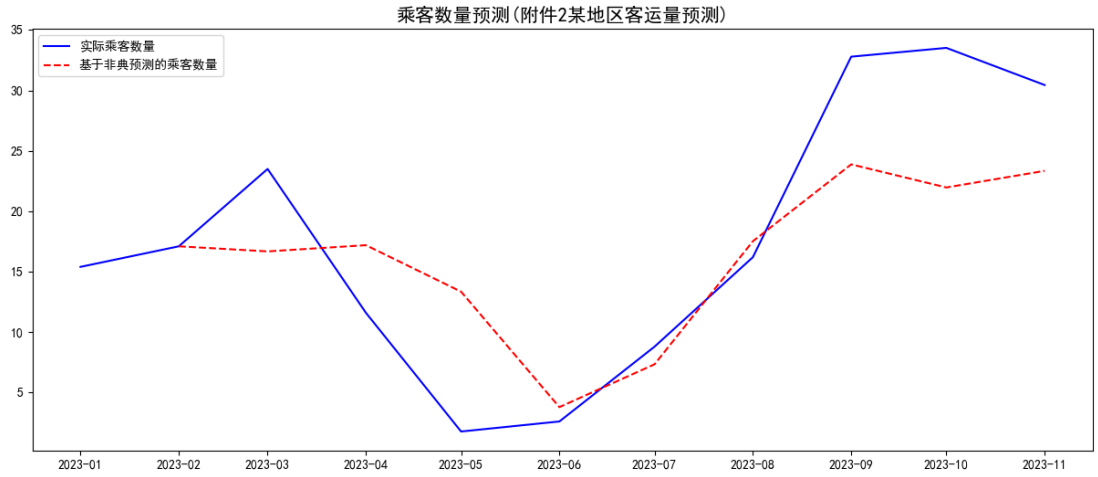

图 17　附件二流感期间某地区外地旅游人数预测

可见实际结果与预测结果还是有一定误差，这可能是由于存在某些外部因素对结果造成影响，但我们还是能够通过结果大致分析流感出现后未来的民航经济复苏趋势，这对未来的世界性流感爆发后的民航甚至其他方面的经济复苏预测有很大的意义。

7 模型总结与评价

7.1 模型总结

7.1.1 模型优点

1)1. 问题一结合实际建立 SEIRD 模型，更全面地考量流感传播过程中存在的不确定性因素，并采用 L-BFGS 算法进行优化，提高算法效率和稳定性。

2) 问题二本文采用改进的 SARIMA 模型，通过 Box-Cox 进行数据变换，使数据更符合模型要求，从而提升模型精度；此外，本文还引入了外部变量，进行外部回归，解决时间序列存在的滞后问题，充分考虑突发情况，使预测结果更贴合实际。

3) 问题三本文建立经济复苏预测模型，选取其他与本题流感相似的疫情作为对照疫情，采用模糊相似性度量的方法，衡量数据集的相似性，证明补充数据集的与问题的相关性。

4) 问题三本文建立基于差分法的预测模型，加入重采样方法以解决时间预测滞后问题，针对流感对民航经济的影响进行较为全面的分析。

7.1.2 模型缺点

1) 问题二由于数据集较小，导致 SARIMA 预测模型训练效果一般，后期可以考虑加入更多数据，在大数据集下进行更完善的训练。

2) 问题三收集的数据集特征维度较少导致经济复苏模型受限于某一方面的预测和分析，后续可以继续完善数据集，进而完善经济复苏模型。

7.2 模型推广

1) 本文建立的 SEIRD 模型与分析思路不仅适用于本题的流感传播预测，也可以推广至其他传染疾病的分析，具有普适性。问题二中建立的改进的 SARIMA 模型，设计了时间滞后修正算法，有效解决了时序数据预测可能存在的时间滞后问题，可以应用到其他时序数据预测模型中。

2) 本文在问题三中建立的经济复苏预测模型，不仅证明了分析往年疫情传播对于处理突发具有传染性疾病的情况具有重要参考意义，同时本模型能在一定程度上预测经济复苏的趋势，这对经济复苏计划的制订有一定的参考价值。

参考文献

[1] 唐梁鸿绪, 王卫苹, 王昊等. 新冠疫情下的 SEIRD 时滞性谣言传播模型及辟谣策略 [J]. 工程科学学报,2022,44(06):1080-1089.

[2] 李文勇, 李俊卓, 王涛. 基于 Box-Cox 指数变换改进的 ARIMA 模型交通流预测方法 [J]. 武汉理工大学学报 (交通科学与工程版),2020,44(06):974-977.

[3] 张雪凝, 施学忠, 赵浩等.SARIMA 和 SARIMA-GRNN 模型在流行性腮腺炎发病率预测中的应用对比 [J]. 中国卫生统计,2020,37(04):489-492.

[4] 薛红新, 刘绕星, 况立群等. 基于百度指数与机器学习方法的流感样病例预测 [J/OL]. 中国预防医学杂志:1-8.

[5] 沙爱龙, 刘泽隆. 禽流感及其对经济的影响 [J]. 生命科学仪器, 2007, 5(4): 10-14.

[6] 何钱, 张旭刚, 张华等. 基于平滑指数法的废旧零部件再制造成本预测 [J]. 机床与液压,2022,50(01):19-24.

[7] Chikobvu D , Sigauke C .Regression-SARIMA modelling of daily peak electricity demand in South Africa[J].Journal of Energy in Southern Africa, 2012, 23(3):23-30.

[8] Siam Z S , Hasan R T , Ahamed H ,et al.Integration of fuzzy logic in the SEIRD compartmental model for the analysis of intervention and transmission heterogeneity on SARS-CoV-2 transmission dynamics[J].International Journal of Modern Physics, C. Physics and Computers, 2022, 33(07): 2250091.

[9] lvaro Alberto López-Lambrao.SARIMA Approach to Generating Synthetic Monthly Rainfall in the Sinú River Watershed in Colombia[J].Atmosphere, 2020, 11(6):602.

[10] Li X .Analysis of economic forecasting in the post-epidemic era: evidence from China[J].Scientific Reports, 2023, 13(1): 2696.

附录 A　总客运量信息表

指标	客运量当期值 (万人)	客运量同比增长 (%)
2020 年 8 月	98214	-37.8
2020 年 7 月	89292	-42
2020 年 6 月	80051	-43.7
2020 年 5 月	73748	-49.3
2020 年 4 月	57320	-60.2
2020 年 3 月	38573	-73
2020 年 2 月	18645	-88.3
2020 年 1 月	127415	-10.1
2019 年 12 月	134466	-3.3
2019 年 11 月	136363	-1.6
2019 年 10 月	155616	-2.7

附录 B　公路客运量信息表

指标	公路客运量当期值 (万人)	公路客运量同比增长 (%)
2020 年 8 月	66934	-39.6
2020 年 7 月	63246	-42.5
2020 年 6 月	59140	-43.1
2020 年 5 月	54234	-49.2
2020 年 4 月	43603	-58.8
2020 年 3 月	29007	-73.1
2020 年 2 月	13792	-88.7
2020 年 1 月	93839	-11.9
2019 年 12 月	101200	-5.5
2019 年 11 月	101962	-4.1
2019 年 10 月	115364	-4.9

附录 C　铁路客运量信息表

指标	铁路客运量当期值 (万人)	铁路客运量同比增长 (%)
2020 年 8 月	24864	-34.4
2020 年 7 月	20737	-41.7
2020 年 6 月	16589	-46
2020 年 5 月	15675	-49.1
2020 年 4 月	11160	-63.5
2020 年 3 月	7495	-73.1
2020 年 2 月	3723	-87.2
2020 年 1 月	27126	-4.3
2019 年 12 月	26306	4.5
2019 年 11 月	27080	7.6
2019 年 10 月	31903	4.7

附录 D　水运客运量信息表

指标	水运客运量当期值 (万人)	水运客运量同比增长 (%)
2020 年 8 月	1799	-41.2
2020 年 7 月	1399	-45.9
2020 年 6 月	1248	-46
2020 年 5 月	1255	-51
2020 年 4 月	886	-61.8
2020 年 3 月	557	-72.3
2020 年 2 月	297	-87.2
2020 年 1 月	1390	-7.1
2019 年 12 月	1684	-4.2
2019 年 11 月	2016	-3.6
2019 年 10 月	2651	-2.4

附录 E　民航客运量信息表

指标	民航客运量当期值 (万人)	民航客运量同比增长 (%)
2020 年 8 月	4617	-24.6

2020 年 7 月	3910	-34.1
2020 年 6 月	3074	-42.4
2020 年 5 月	2583	-52.6
2020 年 4 月	1671	-68.5
2020 年 3 月	1515	-71.7
2020 年 2 月	834	-84.5
2020 年 1 月	5061	-5.3
2019 年 12 月	5276	5.1
2019 年 11 月	5306	6
2019 年 10 月	5698	5.3

附录 F　SARS 流感期间中国部分宏观经济指标数据表

时间	CPI	出口总额 (百万）	进口总额 (百万）	民航客运量当期值 (万人)
2003 年 1 月	103.9	29776	31016	734
2003 年 2 月	96.1	24456	23780	716
2003 年 3 月	92.2	32091	32549	738
2003 年 4 月	93.3	35617	34601	573
2003 年 5 月	105.3	33842	31604	163
2003 年 6 月	105	34476	32336	315
2003 年 7 月	105.2	38110	36514	752
2003 年 8 月	114.4	37415	34621	1025
2003 年 9 月	107.6	41941	41652	943
2003 年 10 月	105.1	40926	35193	1002
2003 年 11 月	101.2	41762	36893	889

附录 G　H1N1 流感期间美国部分宏观经济指标数据表

时间	CPI	出口总额 (百万）	进口总额 (百万）	民航客运量当期值 (万人)
2008 年 11 月	5.943	87992	139234	5449.2
2008 年 12 月	5.978	82415	129116	5779.4
2009 年 1 月	5.245	84903	121807	5225.7

2009 年 2 月	4.721	83500	121885	5021.7
2009 年 3 月	4.336	81088	120572	6140.3
2009 年 4 月	3.314	83369	118736	5996.6
2009 年 5 月	2.756	85737	122156	6016.7
2009 年 6 月	2.174	87833	129912	6400.5
2009 年 7 月	1.056	87767	128770	6860.4
2009 年 8 月	0.558	91673	136267	6554.9
2009 年 9 月	-0.039	94829	139309	5516.8

附录 H　COVID-19 流感期间中国部分宏观经济指标数据表

时间	CPI	出口总额 (百万)	进口总额 (百万)	民航客运量当期值 (万人)
2019 年 10 月	99.7	212916.095	169898.075	5698
2019 年 11 月	93.2	221337.589	183411.013	5306
2019 年 12 月	92	211590.876	191057.245	5276
2020 年 1 月	95	238267.589	156901.124	5061
2020 年 2 月	94.4	80380.286	142499.591	834
2020 年 3 月	93.99	185146.278	165213.108	1515
2020 年 4 月	89.5	200233.646	154900.580	1671
2020 年 5 月	80.7	206812.825	143886.746	2583
2020 年 6 月	71	213574.174	167152.978	3074
2020 年 7 月	72.3	237631.279	175301.874	3910
2020 年 8 月	80.2	235259.186	176333.799	4617

附录 I　问题一源代码

9.1　SEIRD 模型变化曲线图绘制 python 代码

```
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号

# Load the data
data_updated = pd.read_excel("E:/Desktop/附件1某地流感数据.xlsx")
```

```
# Calculate daily new cases, recoveries, and deaths
data_updated['新确诊'] = data_updated['确诊病例'].diff().fillna(data_updated['确诊病例'].iloc[0])
data_updated['新死亡'] = data_updated['死亡累计'].diff().fillna(data_updated['死亡累计'].iloc[0])
data_updated['新恢复'] =
    data_updated['治愈出院累计'].diff().fillna(data_updated['治愈出院累计'].iloc[0])

# Define the total population
total_population_updated = 10000

# Calculate the susceptible population
data_updated['S'] = total_population_updated - data_updated['确诊病例'] -
    data_updated['疑似病例'] - data_updated['死亡累计'] - data_updated['治愈出院累计']

# Estimate parameters
data_updated[' '] = data_updated['新确诊'] / (data_updated['S'] * data_updated['确诊病例'] /
    total_population_updated)
data_updated[' '] = data_updated['新恢复'] / data_updated['确诊病例']
data_updated[' '] = data_updated['新死亡'] / data_updated['确诊病例']
data_updated[' '] = data_updated['新确诊'] /
    (data_updated['疑似病例'].shift(1).fillna(data_updated['疑似病例'].iloc[0]))

# Function to plot the estimated parameters over time
def plot_parameters(data):
   fig, axes = plt.subplots(4, 1, figsize=(12, 16))

   # Plotting beta
   axes[0].plot(data['日期'], data[' '], label=' （感染率）', color='blue')
   axes[0].set_title(' （感染率）随时间的变化',fontsize=15)
   axes[0].set_ylabel(' ',fontsize=15)
   axes[0].legend()
   axes[0].grid(True)


   # Plotting sigma
   axes[1].plot(data['日期'], data[' '], label=' （从暴露者转变为感染者的比例）', color='green')
   axes[1].set_title(' （从暴露者转变为感染者的比例）随时间的变化',fontsize=15)
   axes[1].set_ylabel(' ',fontsize=15)
   axes[1].legend()
   axes[1].grid(True)

   # Plotting gamma
   axes[2].plot(data['日期'], data[' '], label=' （恢复率）', color='orange')
   axes[2].set_title(' （恢复率）随时间的变化',fontsize=15)
   axes[2].set_ylabel(' ',fontsize=15)
   axes[2].legend()
   axes[2].grid(True)
```

```
    # Plotting mu
    axes[3].plot(data['日期'], data[' '], label=' （死亡率)', color='red')
    axes[3].set_title(' （死亡率）随时间的变化',fontsize=15)
    axes[3].set_ylabel(' ',fontsize=15)
    axes[3].legend()
    axes[3].grid(True)

    plt.tight_layout()
    plt.savefig("E:/Desktop/变化率.png")
    plt.show()

# Call the function to plot the parameters
plot_parameters(data_updated)
```

9.2 SEIRD 模型 python 代码

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

# Load data
data = pd.read_excel("E:/Desktop/附件1某地流感数据.xlsx")

# Define the SEIRD model
def SEIRD_model(y, t, beta, sigma, gamma, mu):
    S, E, I, R, D = y
    dS_dt = -beta * S * I / total_population_updated
    dE_dt = beta * S * I / total_population_updated - sigma * E
    dI_dt = sigma * E - gamma * I - mu * I
    dR_dt = gamma * I
    dD_dt = mu * I
    return [dS_dt, dE_dt, dI_dt, dR_dt, dD_dt]

# Define the SEIRD model with intervention affecting only beta
def SEIRD_intervention_beta_only(y, t, beta, sigma, gamma, mu, intervention_day,
    reduction_factor):
    if t >= intervention_day:
        beta *= reduction_factor
    S, E, I, R, D = y
    dS_dt = -beta * S * I / total_population_updated
    dE_dt = beta * S * I / total_population_updated - sigma * E
    dI_dt = sigma * E - gamma * I - mu * I
    dR_dt = gamma * I
    dD_dt = mu * I
```

```
    return [dS_dt, dE_dt, dI_dt, dR_dt, dD_dt]

# Loss function to minimize (difference between model and data)
def loss(params):
    beta, sigma, gamma, mu = params
    sol = odeint(SEIRD_model, [S_first_day_updated, data.loc[0, 'E'], data.loc[0, 'I'],
        data.loc[0, 'R'], data.loc[0, 'D']], np.arange(len(data)), args=(beta, sigma, gamma,
        mu))
    loss = np.sum((sol - model_data_updated.values)**2)
    return loss

# Prepare the dataset for SEIRD modeling
data['S'] = total_population_updated - data['确诊病例'] - data['疑似病例'] - data['死亡累计'] -
    data['治愈出院累计']
data['E'] = data['疑似病例']
data['I'] = data['确诊病例']
data['R'] = data['治愈出院累计']
data['D'] = data['死亡累计']

# Select the relevant columns for modeling
model_data_updated = data[['S', 'E', 'I', 'R', 'D']]

# Update the total population and the initial susceptible count
total_population_updated = 10000
S_first_day_updated = total_population_updated - data.loc[0, '确诊病例'] - data.loc[0,
    '疑似病例'] - data.loc[0, '死亡累计'] - data.loc[0, '治愈出院累计']
S_first_day_updated

# Initial guess for the parameters
initial_guess = [0.5, 0.5, 0.5, 0.5]

# Estimate the best parameters using minimize function
result = minimize(loss, initial_guess, method='L-BFGS-B', bounds=[(0, 5), (0, 5), (0, 5), (0,
    5)])
best_params_updated = result.x

# Parameters for intervention
reduction_factor = 0.8 # 50% reduction in transmission due to intervention

# Simulate for 200 days
t_extended = np.arange(200)

# No intervention
simulated_no_intervention = odeint(SEIRD_model, [S_first_day_updated, data.loc[0, 'E'],
    data.loc[0, 'I'], data.loc[0, 'R'], data.loc[0, 'D']], t_extended,
    args=tuple(best_params_updated))
```

```
# Intervention 5 days earlier
simulated_early_intervention_corrected = odeint(SEIRD_intervention_beta_only,
    [S_first_day_updated, data.loc[0, 'E'], data.loc[0, 'I'], data.loc[0, 'R'], data.loc[0,
    'D']], t_extended, args=(*best_params_updated, 10, reduction_factor))

# Intervention 5 days later
simulated_late_intervention_corrected = odeint(SEIRD_intervention_beta_only,
    [S_first_day_updated, data.loc[0, 'E'], data.loc[0, 'I'], data.loc[0, 'R'], data.loc[0,
    'D']], t_extended, args=(*best_params_updated, 20, reduction_factor))

# Calculate the daily new cases for each scenario for the entire simulation
daily_new_cases_no_intervention_full = np.diff(simulated_no_intervention[:, 2])
daily_new_cases_early_intervention_full = np.diff(simulated_early_intervention_corrected[:, 2])
daily_new_cases_late_intervention_full = np.diff(simulated_late_intervention_corrected[:, 2])

# Plot the daily new cases for the entire simulation
fig, ax = plt.subplots(figsize=(12, 8))
ax.plot(np.arange(1, 200), daily_new_cases_no_intervention_full, 'r', label='No Intervention')
ax.plot(np.arange(1, 200), daily_new_cases_early_intervention_full, 'b', label='Intervention 5
    Days Earlier')
ax.plot(np.arange(1, 200), daily_new_cases_late_intervention_full, 'g', label='Intervention 5
    Days Later')
ax.legend()
ax.set_title('Daily New Cases for 200 Days')
ax.set_xlabel('Days')
ax.set_ylabel('Daily New Cases')

plt.grid() # 生成网格
plt.show()

# Simulate the SEIRD model for 200 days using the newly estimated parameters
t_extended = np.arange(200)
simulated_SEIRD_200days = odeint(SEIRD_model, [S_first_day_updated, data.loc[0, 'E'],
    data.loc[0, 'I'], data.loc[0, 'R'], data.loc[0, 'D']], t_extended,
    args=tuple(best_params_updated))

# Plot the simulated data for 200 days
fig, ax = plt.subplots(figsize=(12, 8))
ax.plot(t_extended, simulated_SEIRD_200days[:, 0], 'b--', label='累计易感者（普通人群）')
ax.plot(t_extended, simulated_SEIRD_200days[:, 1], 'y--', label='累计确诊病例')
ax.plot(t_extended, simulated_SEIRD_200days[:, 2], 'r--', label='累计疑似病例')
ax.plot(t_extended, simulated_SEIRD_200days[:, 3], 'g--', label='累计治愈出院人数')
ax.plot(t_extended, simulated_SEIRD_200days[:, 4], 'k--', label='累计死亡人数')

# # If the actual data is available for those 200 days, plot that too
# if len(model_data_updated) >= 200:
#     ax.plot(t_extended, model_data_updated['S'].values[:200], 'b', label='Actual Susceptible')
```

```
#     ax.plot(t_extended, model_data_updated['E'].values[:200], 'y', label='Actual Exposed')
#     ax.plot(t_extended, model_data_updated['I'].values[:200], 'r', label='Actual Infected')
#     ax.plot(t_extended, model_data_updated['R'].values[:200], 'g', label='Actual Recovered')
#     ax.plot(t_extended, model_data_updated['D'].values[:200], 'k', label='Actual Deceased')

ax.legend()
ax.set_title('SEIRD模型近200天的人数变化图',fontsize=15)
ax.set_xlabel('天数',fontsize=12)
ax.set_ylabel('人数',fontsize=12)
plt.grid() # 生成网格
plt.savefig("E:/Desktop/SEIRD模型近200天的人数变化图.png")
plt.show()

from scipy.integrate import odeint
# Re-simulate with intervention affecting only beta, ensuring beta is reduced properly
simulated_early_intervention_corrected = odeint(SEIRD_intervention_beta_only,
    [S_first_day_updated, data.loc[0, 'E'], data.loc[0, 'I'], data.loc[0, 'R'], data.loc[0,
    'D']], t_extended, args=(*best_params_updated, 5, reduction_factor))
simulated_late_intervention_corrected = odeint(SEIRD_intervention_beta_only,
    [S_first_day_updated, data.loc[0, 'E'], data.loc[0, 'I'], data.loc[0, 'R'], data.loc[0,
    'D']], t_extended, args=(*best_params_updated, 15, reduction_factor))
simulated_10_intervention_corrected = odeint(SEIRD_intervention_beta_only,
    [S_first_day_updated, data.loc[0, 'E'], data.loc[0, 'I'], data.loc[0, 'R'], data.loc[0,
    'D']], t_extended, args=(*best_params_updated, 10, reduction_factor))
# Plotting
fig, ax = plt.subplots(figsize=(12, 8))
ax.plot(t_extended, simulated_no_intervention[:, 2], 'gold', label='无干预')
ax.plot(t_extended, simulated_early_intervention_corrected[:, 2], 'orangered',
    label='提前5天隔离(第5天)')
ax.plot(t_extended, simulated_late_intervention_corrected[:, 2], 'limegreen',
    label='延后5天隔离(第15天)')
ax.plot(t_extended, simulated_10_intervention_corrected[:, 2], 'royalblue',
    label='延后5天隔离(第10天)')
ax.legend()
ax.set_title('提前隔离与延后隔离对感染人数的影响曲线图',fontsize=15)
ax.set_xlabel('天数',fontsize=12)
ax.set_ylabel('感染人数',fontsize=12)

y1_max=np.argmax(simulated_early_intervention_corrected[:, 2])
# show_max='['+str(y1_max)+' '+str(y1[y1_max])+']'
# # 以绘制最大值点和最小值点的位置
# plt.plot(y1_max,simulated_no_intervention[:, 2][y1_max],'ko')
# plt.annotate(show_max,xy=(y1_max,(simulated_no_intervention[:,
    2])[y1_max]),xytext=(y1_max,(simulated_no_intervention[:, 2])[y1_max]))
plt.savefig("E:/Desktop/提前隔离与延后隔离对感染人数的影响曲线图.png")
plt.show()
```

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

# Load data
data = pd.read_excel("E:/Desktop/附件1某地流感数据.xlsx")

# Define the SEIRD model
def SEIRD_model(y, t, beta, sigma, gamma, mu):
    S, E, I, R, D = y
    dS_dt = -beta * S * I / total_population_updated
    dE_dt = beta * S * I / total_population_updated - sigma * E
    dI_dt = sigma * E - gamma * I - mu * I
    dR_dt = gamma * I
    dD_dt = mu * I
    return [dS_dt, dE_dt, dI_dt, dR_dt, dD_dt]

# Define the SEIRD model with intervention affecting only beta
def SEIRD_intervention_beta_only(y, t, beta, sigma, gamma, mu, intervention_day,
    reduction_factor):
    if t >= intervention_day:
        beta *= reduction_factor
    S, E, I, R, D = y
    dS_dt = -beta * S * I / total_population_updated
    dE_dt = beta * S * I / total_population_updated - sigma * E
    dI_dt = sigma * E - gamma * I - mu * I
    dR_dt = gamma * I
    dD_dt = mu * I
    return [dS_dt, dE_dt, dI_dt, dR_dt, dD_dt]

# Update the total population and the initial susceptible count
total_population_updated = 10000
S_first_day_updated = total_population_updated - data.loc[0, '确诊病例'] - data.loc[0,
    '疑似病例'] - data.loc[0, '死亡累计'] - data.loc[0, '治愈出院累计']

# Prepare the dataset for SEIRD modeling
data['S'] = total_population_updated - data['确诊病例'] - data['疑似病例'] - data['死亡累计'] -
    data['治愈出院累计']
data['E'] = data['疑似病例']
data['I'] = data['确诊病例']
data['R'] = data['治愈出院累计']
data['D'] = data['死亡累计']

# Select the relevant columns for modeling
model_data_updated = data[['S', 'E', 'I', 'R', 'D']]
```

```
# Parameters
reduction_factor = 0.8 # 50% reduction in transmission due to intervention
# best_params_updated = ... # Use the previously estimated parameters
result = minimize(loss, initial_guess, method='L-BFGS-B', bounds=[(0, 5), (0, 5), (0, 5), (0,
    5)])
best_params_updated = result.x

# Simulate for 200 days
t_extended = np.arange(200)

# No intervention
simulated_no_intervention = odeint(SEIRD_model, [S_first_day_updated, data.loc[0, 'E'],
    data.loc[0, 'I'], data.loc[0, 'R'], data.loc[0, 'D']], t_extended,
    args=tuple(best_params_updated))

# Intervention 5 days earlier
simulated_early_intervention_corrected = odeint(SEIRD_intervention_beta_only,
    [S_first_day_updated, data.loc[0, 'E'], data.loc[0, 'I'], data.loc[0, 'R'], data.loc[0,
    'D']], t_extended, args=(*best_params_updated, 5, reduction_factor))

# Intervention 5 days later
simulated_late_intervention_corrected = odeint(SEIRD_intervention_beta_only,
    [S_first_day_updated, data.loc[0, 'E'], data.loc[0, 'I'], data.loc[0, 'R'], data.loc[0,
    'D']], t_extended, args=(*best_params_updated, 15, reduction_factor))

# Intervention 10 days later
simulated_10_intervention_corrected = odeint(SEIRD_intervention_beta_only,
    [S_first_day_updated, data.loc[0, 'E'], data.loc[0, 'I'], data.loc[0, 'R'], data.loc[0,
    'D']], t_extended, args=(*best_params_updated, 10, reduction_factor))

# Calculate the daily new cases for each scenario for the entire simulation
daily_new_cases_no_intervention_full = np.diff(simulated_no_intervention[:, 2])
daily_new_cases_early_intervention_full = np.diff(simulated_early_intervention_corrected[:, 2])
daily_new_cases_late_intervention_full = np.diff(simulated_late_intervention_corrected[:, 2])
daily_new_cases_10_intervention_full = np.diff(simulated_10_intervention_corrected[:, 2])

# Plot the daily new cases for the entire simulation
fig, ax = plt.subplots(figsize=(12, 8))
ax.plot(np.arange(1, 200), daily_new_cases_no_intervention_full, 'gold', label='无干预情况')
ax.plot(np.arange(1, 200), daily_new_cases_early_intervention_full, 'orangered',
    label='提前五天隔离（第5天）')
ax.plot(np.arange(1, 200), daily_new_cases_late_intervention_full, 'limegreen',
    label='延后五天隔离（第15天）')
ax.plot(np.arange(1, 200), daily_new_cases_10_intervention_full, 'royalblue',
    label='第10天开始隔离')
ax.legend()
ax.set_title('近200天的每日新增病例',fontsize=15)
```

```
ax.set_xlabel('天数',fontsize=12)
ax.set_ylabel('每日新增病例',fontsize=12)
plt.savefig("E:/Desktop/近200天的每日新增病例.png")
plt.show()
```

附录 J　问题二源代码

10.1　数据预处理 python 代码

```
   import pandas as pd
import matplotlib.pyplot as plt

plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签

plt.rcParams['axes.unicode_minus']=False #用来正常显示负号
# Load the data
tourism_data = pd.read_excel("E:/Desktop/附件2某地接待海外旅游人数0.xlsx")

# Data preprocessing
tourism_long = pd.melt(tourism_data, id_vars=['某地接待海外旅游的人数（单位：万人）'],
                    value_vars=tourism_data.columns[1:],
                    var_name='Month', value_name='Tourists')
tourism_long['Month'] = tourism_long['Month'].str.extract('(\d+)').astype(int)
tourism_long['Date'] =
    pd.to_datetime(tourism_long['某地接待海外旅游的人数（单位：万人）'].astype(str) + '-' +
    tourism_long['Month'].astype(str) + '-01')
tourism_long['Year'] = tourism_long['Date'].dt.year

# Plotting the data for each year
plt.figure(figsize=(12, 6))
for year in tourism_long['Year'].unique():
   yearly_data = tourism_long[tourism_long['Year'] == year]
   plt.plot(yearly_data['Month'], yearly_data['Tourists'], label=str(year), marker='o')

plt.title('每月海外游客人数分布折线图',fontsize=15)
#设置坐标标签标注和字体大小
plt.xlabel("月份",fontsize=10)
plt.ylabel("游客的人数(in 10,000s)",fontsize=10)
plt.xticks(ticks=range(1, 13), labels=['一月', '二月', '三月', '四月', '五月', '六月', '七月',
    '八月', '九月', '十月', '十一月', '十二月'])
plt.legend()
plt.grid(True)
plt.savefig("E:/Desktop/每月海外游客人数分布折线图.png")
plt.show()
```

10.2 绘制 2017-2023 年各月份海外旅游人变化折线图 python 代码

```
   import pandas as pd
import matplotlib.pyplot as plt

plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签

plt.rcParams['axes.unicode_minus']=False #用来正常显示负号
# Load the data
tourism_data = pd.read_excel("E:/Desktop/附件2某地接待海外旅游人数0.xlsx")

# Data preprocessing
tourism_long = pd.melt(tourism_data, id_vars=['某地接待海外旅游的人数（单位：万人）'],
                    value_vars=tourism_data.columns[1:],
                    var_name='Month', value_name='Tourists')
tourism_long['Month'] = tourism_long['Month'].str.extract('(\d+)').astype(int)
tourism_long['Date'] =
    pd.to_datetime(tourism_long['某地接待海外旅游的人数（单位：万人）'].astype(str) + '-' +
    tourism_long['Month'].astype(str) + '-01')
tourism_long['Year'] = tourism_long['Date'].dt.year

# Plotting the data for each year
plt.figure(figsize=(12, 6))
for year in tourism_long['Year'].unique():
    yearly_data = tourism_long[tourism_long['Year'] == year]
    plt.plot(yearly_data['Month'], yearly_data['Tourists'], label=str(year), marker='o')

plt.title('每月海外游客人数分布折线图',fontsize=15)
#设置坐标标签标注和字体大小
plt.xlabel("月份",fontsize=10)
plt.ylabel("游客的人数(in 10,000s)",fontsize=10)
plt.xticks(ticks=range(1, 13), labels=['一月', '二月', '三月', '四月', '五月', '六月', '七月',
    '八月', '九月', '十月', '十一月', '十二月'])
plt.legend()
plt.grid(True)
plt.savefig("E:/Desktop/每月海外游客人数分布折线图.png")
plt.show()
```

10.3 SARIMA python 代码

```
    ##SARIMA##
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.statespace.sarimax import SARIMAX
```

```
from scipy.stats import boxcox
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号
# Load the data
data = pd.read_excel("E:/Desktop/processed_overseas_visitors.xlsx")
data.set_index('Date',inplace = True)

# Box-Cox transformation
transformed_series, lambda_value = boxcox(data['Visitors'])

# External regressors
time_series_data_filled = data.asfreq('MS', method='ffill')
time_series_data_filled['off_season'] = ((time_series_data_filled.index.month == 11) |
                                  (time_series_data_filled.index.month == 12) |
                                  (time_series_data_filled.index.month == 1)).astype(int)

time_series_data_filled['peak_season'] = ((time_series_data_filled.index.month == 8) |
                                   (time_series_data_filled.index.month == 9) |
                                   (time_series_data_filled.index.month == 10)).astype(int)

time_series_data_filled['covid_effect'] = ((time_series_data_filled.index.year == 2023) &
                                   ((time_series_data_filled.index.month >= 2) &
                                    (time_series_data_filled.index.month <= 4))).astype(int)

time_series_data_filled['covid_recovery'] = ((time_series_data_filled.index.year == 2023) &
                                     ((time_series_data_filled.index.month >= 5) &
                                      (time_series_data_filled.index.month <= 7))).astype(int)

# SARIMA model
model_sarima_final_adjusted = SARIMAX(transformed_series[:-5],
                                exog=time_series_data_filled[['off_season', 'peak_season',
                                     'covid_effect', 'covid_recovery']][:-5],
                                order=(1, 1, 1),
                                seasonal_order=(1, 1, 1, 12))
results_sarima_final_adjusted = model_sarima_final_adjusted.fit()

transformed_series = transformed_series.tolist()

# Adjusted exogenous data for forecasting
extended_dates = pd.date_range(start='2023-06-01', end='2024-02-01', freq='MS')
extended_exog = time_series_data_filled[['off_season', 'peak_season', 'covid_effect',
    'covid_recovery']].reindex(extended_dates).fillna(0)
forecast_exog_extended = extended_exog['2023-07-01':'2024-03-01']

# In-sample prediction up to July 2023
forecast_in_sample_extended = results_sarima_final_adjusted.get_prediction(start='2019-01-01',
    end='2023-07-01', dynamic=False).predicted_mean
```

```
forecast_in_sample_original_scale_extended = np.exp(np.log(lambda_value *
    forecast_in_sample_extended + 1) / lambda_value)

# Out-of-sample forecast from August 2023 to March 2024
forecast_out_sample_extended_corrected = results_sarima_final_adjusted.get_forecast(steps=8,
    exog=forecast_exog_extended)
forecast_original_scale_extended_corrected = np.exp(np.log(lambda_value *
    forecast_out_sample_extended_corrected.predicted_mean + 1) / lambda_value)

# Plotting the comparison
plt.figure(figsize=(15, 6))
time_series_data_filled['Visitors'][:'2023-07-01'].plot(label='实际值', linewidth=2)
forecast_in_sample_original_scale_extended.plot(label='样本值预测（存在实际值的样本）',
    linestyle='--', color='blue')
forecast_original_scale_extended_corrected.plot(label='需要预测的样本', linestyle='--',
    color='red')
plt.title('实际与预测的海外游客人数（包含需要预测样本）',fontsize=15)
plt.xlabel('日期',fontsize=12)
plt.ylabel('人数（千人）',fontsize=12)
plt.legend()
plt.grid(True)
plt.savefig("E:/Desktop/实际与预测的海外游客人数（包含需要预测样本）.png")
plt.show()
```

附录 K　问题三源代码

11.1　经济复苏综合指标预测 python 代码

```
    import pandas as pd
import numpy as np
from sklearn.decomposition import PCA

# Load the data
sars_data = pd.read_excel("E:/Desktop/SARS.xlsx")
h1n1_data = pd.read_excel("E:/Desktop/H1N1.xlsx")
covid_data = pd.read_excel("E:/Desktop/COVID.xlsx")

# Standardize the data
def standardize_data(df):
    for col in df.columns[1:]:
        mean = df[col].mean()
        std = df[col].std()
        df[col] = (df[col] - mean) / std
    return df
```

```
sars_data_standardized = standardize_data(sars_data.copy())
h1n1_data_standardized = standardize_data(h1n1_data.copy())
covid_data_standardized = standardize_data(covid_data.copy())

# Compute the closeness degree
def compute_closeness_degree(reference_data, comparison_data):
    n = len(reference_data.columns) - 1 # Number of economic indicators
    closeness_degrees = []

    for _, row in comparison_data.iterrows():
        sum_squared_diff = 0
        for col in reference_data.columns[1:]:
            sum_squared_diff += (row[col] - reference_data.iloc[0][col]) ** 2
        beta_i = 1 - (1 / np.sqrt(n)) * np.sqrt(sum_squared_diff)
        closeness_degrees.append(beta_i)

    return closeness_degrees

sars_closeness_standardized = compute_closeness_degree(covid_data_standardized,
    sars_data_standardized)
h1n1_closeness_standardized = compute_closeness_degree(covid_data_standardized,
    h1n1_data_standardized)

# Recursive exponential smoothing
def recursive_exponential_smoothing(reference_data, comparison_data, betas):
    predictions = []
    for i in range(1, len(comparison_data)):
        s_star = betas[i] * comparison_data.iloc[i-1, 1] + (1 - betas[i]) *
            reference_data.iloc[i-1, 1]
        predictions.append(s_star)
    return predictions

sars_predictions = recursive_exponential_smoothing(covid_data_standardized,
    sars_data_standardized, sars_closeness_standardized)
h1n1_predictions = recursive_exponential_smoothing(covid_data_standardized,
    h1n1_data_standardized, h1n1_closeness_standardized)

# PCA
combined_data = pd.concat([sars_data_standardized, h1n1_data_standardized,
    covid_data_standardized])
X = combined_data.iloc[:, 1:].values
pca = PCA()
principal_components = pca.fit_transform(X)
coefficients = pca.components_[:3]

def compute_recovery_indicator(data, coefficients):
    return data.dot(coefficients.T)
```

```
recovery_indicator = compute_recovery_indicator(X, coefficients)
sars_recovery = recovery_indicator[:len(sars_data_standardized)]
h1n1_recovery =
    recovery_indicator[len(sars_data_standardized):len(sars_data_standardized)+len(h1n1_data_standardized)]
h1n1_recovery = recovery_indicator[-len(covid_data_standardized):]
```

11.2 经济复苏情况预测 python 代码

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

plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号

# 1. 加载数据集
sars_df = pd.read_excel("E:/Desktop/SARS.xlsx")
covid_df = pd.read_excel("E:/Desktop/COVID.xlsx")

# 2. 提取非典和COVID-19数据
sars_passenger = sars_df['民航客运量当期值(万人)'].values
# sars_profit = sars_df['利润'].values
covid_passenger = covid_df['民航客运量当期值(万人)'].values
# covid_profit = covid_df['利润'].values

# 3. 计算非典期间的月度增长率
sars_passenger_growth = np.diff(sars_passenger) / sars_passenger[:-1]
# sars_profit_growth = np.diff(sars_profit) / sars_profit[:-1]

# 4. 基于非典增长率预测COVID-19后的恢复
predicted_passenger = [covid_passenger[1]]
# predicted_profit = [covid_profit[1]]
for i in range(1, len(covid_passenger)-1):
    predicted_passenger.append(predicted_passenger[-1] * (1 + sars_passenger_growth[i-1]))
#     predicted_profit.append(predicted_profit[-1] * (1 + sars_profit_growth[i-1]))

# 5. 绘制实际数据和预测
months = covid_df['时间'].values
x = pd.date_range(start='2003/01/01', end='2003/11/01', freq='D')
plt.figure(figsize=(15, 6))

# 乘客数量预测
plt.subplot(1, 1, 1)
plt.plot(months, covid_passenger, label='实际乘客数量', color='blue')
```

```
plt.plot(months[1:], predicted_passenger, '--', label='基于非典预测的乘客数量', color='red')
plt.title('乘客数量预测',fontsize = 15)
plt.legend()

# 利润预测
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号# plt.subplot(1, 2, 2)
# plt.plot(months, covid_profit, label='实际利润', color='red')
# plt.plot(months[1:], predicted_profit, '--', label='基于非典预测的利润', color='orange')
# plt.title('利润预测')
# plt.legend()
plt.savefig("E:/Desktop/乘客数量预测.png")
plt.show()
```