赛区评阅编号（由赛区组委会填写）：

2020 高教社杯全国大学生数学建模竞赛

承　诺　书

我们仔细阅读了《全国大学生数学建模竞赛章程》和《全国大学生数学建模竞赛参赛规则》（2019 年修订稿，以下简称为“竞赛章程和参赛规则”，可从全国大学生数学建模竞赛网站下载）。

我们完全清楚，在竞赛开始后参赛队员不能以任何方式，包括电话、电子邮件、“贴吧”、QQ 群、微信群等，与队外的任何人（包括指导教师）交流、讨论与赛题有关的问题；无论主动参与讨论还是被动接收讨论信息都是严重违反竞赛纪律的行为。

我们完全清楚，抄袭别人的成果是违反竞赛章程和参赛规则的行为；如果引用别人的成果或资料（包括网上资料），必须按照规定的参考文献的表述方式列出，并在正文引用处予以标注。

我们以中国大学生名誉和诚信郑重承诺，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。

我们授权全国大学生数学建模竞赛组委会，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）。

我们参赛选择的题号（从 A/B/C/D/E 中选择一项填写）：A

我们的报名参赛队号（12 位数字全国统一编号）：xxx

参赛学校（完整的学校全称，不含院系名）：福州大学

参赛队员 (打印并签名) ：1. 谢润锋

2. 陈翰泽

3. 程灵飞

指导教师或指导教师组负责人 (打印并签名)：xxx

（指导教师签名意味着对参赛队的行为和论文的真实性负责）

日期: 2020 年 07 月 13 日

（请勿改动此页内容和格式。此承诺书打印签名后作为纸质论文的封面，注意电子版论文中不得出现此页。以上内容请仔细核对，如填写错误，论文可能被取消评奖资格。）

赛区评阅编号（由赛区组委会填写）：

2020 高教社杯全国大学生数学建模竞赛

编 号 专 用 页

赛区评阅记录（可供赛区评阅时使用）：

评阅人						
备注						

送全国评阅统一编号（由赛区组委会填写）：

全国评阅随机编号（由全国组委会填写）：

（请勿改动此页内容和格式。此编号专用页仅供赛区和全国评阅使用，参赛队打印后装订到纸质论文的第二页上。注意电子版论文中不得出现此页。）

基于微分方程和迭代模拟的新冠肺炎传播规律建模研究

摘要

2020 年初，新冠肺炎病毒在武汉被最先发现，并开始在全球广泛传播，至今人类仍未摆脱此病毒所引起的影响。意大利是欧洲最早收到疫情冲击的国家之一，所幸意大利政府果断采取了有效的防疫措施，及时做出了封城的决定，中国也多次派出医疗团队并三次援助了医疗物资，有效缓解了意大利国内的疫情压力，目前意大利的疫情已经进入平稳期。自 2020 年 2 月中下旬开始，意大利经历了较为完整的疫情防控过程，又处在欧洲和欧盟的核心区域，十分具有代表性。

本文以 COVID-19 病毒的具体特征和疫情相关数据出发，结合了意大利封城措施执行时间和中国的三次医疗援助的抵达时间，以基于微分方程的 SIR 模型为基础提出了一种 SEIRD 模型和一种迭代模拟的模型来描述意大利的疫情发展变化趋势，评估了意大利政府和中国医疗援助的有效性和及时性，能够作为其他国家的疫情防控的参考，并为意大利未来的疫情发展做出预测。

关键字： COIVID-19　传染病模型　微分方程　迭代模拟

一、 问题重述

自 2020 年初开始，一种名为 COVID-19 的新型冠状病毒在中国武汉最先被发现，并开始在全球大范围传播，由此病毒引发的疾病被称为新型冠状病毒肺炎（简称新冠肺炎），医学研究和防疫经验表明这是一种传染性极强的流行病，迄今为止仍有许多国家尚未有效遏制其传播，它的爆发和蔓延给不但给各国医疗卫生系统带来巨大压力，还严重影响了经济社会的正常运转，成为二十一世纪人类面临的最大灾难之一。

若能定性、定量地分析、研究这种传染病的传播规律并总结有效的防控措施，就能为预测和控制其蔓延创造条件。

问题 1 选择一个疫情较为严重的地区或城市，自行查找相关数据，建立一个能够反映在疫情防控期间该地区或城市受感染人数、死亡人数变化规律的数学模型，并评价你们所建模型的合理性和实用性。

问题 2 按照问题 1 所建立的模型，对当地政府部门所采取的措施做出评价，如：提前或延后 5 天采取严格的隔离措施，对疫情传播所造成的影响（如受感染人数、死亡人数、传播高潮期来到的时间）做出估计。

问题 3 收集该地区或城市某一个行业的相关数据，预测疫情对该行业经济指标在短期将会造成什么样的影响。

二、 模型假设

1. 假设此模型所分析的地区人口迁入迁出大致相等，且迁入人口和迁出人口中病毒携带者的比例大致相同。
2. 假设模型一中多用的接触感染率、治愈率、死亡率都是分段常数函数
3. 选择意大利的 COVID-19 作为分析地区，收集到了 2020 年 2 月 22 日到 2020 年 7 月 9 日的数据 [1]。
4. 根据公开资料显示，COVID-19 病毒的潜伏期约为 3 到 7 天，故本文在 SEIRD 模型中使用了 5 天作为潜伏期数据。

三、 符号说明

符号	意义
S	健康人群占比。
I	确诊病人占比。
R	治愈病人比例，已免疫。
D	死亡病人比例。
E	病毒携带者比例。
N	总人数。
α_i	确诊病人的平均接触感染率。
α_e	病毒携带者的平均接触感染率。
β	治愈率。
γ	死亡率。
s_0	初始易感人群。
i_0	初始感染人数。
r_0	初始的治愈人数。
d_0	初始死亡人数。
e_0	初始疑似感染者人数。
$incu$	潜伏期。
y	每日新增确诊人数
t	时间，以天为单位

四、 问题分析

4.1 对问题 1 的回答

4.1.1 模型一

对于 COVID-19 传染病，我们直接采用了最为常见的模型 SIR 作为基础。由于潜伏期人群在实际中也具有感染性以及题目要求，所以我们最终将建模方法定为 SEIRD，即

易感人群、潜伏期人群、确诊感染人群、痊愈人群和死亡人群，其中潜伏期和确诊感染人群各有自己的感染接触率 α_e 和 α_i。对于潜伏期人群转化至确诊感染人群，我们设置一个 $incu$ 潜伏期参数，对于某个阶段潜伏者人群中某个的潜伏者，平均将会有 $\frac{1}{incu}$ 的概率被确诊；而对于易感人群转化至潜伏期人群，我们根据不同阶段、不同措施、不同医疗资源和水平拟合且区分开了潜伏期和确诊感染人群各自的感染能力，通过感染接触率进行加权，以灵活体现感染能力的总体变化和潜在传染源的威胁。综上，由该疾病的特征和疾病发展的正常规律，可以得到以下方程：

$$\begin{cases} \frac{\mathrm{d}S}{\mathrm{d}t} = -\alpha_i SI + \alpha_e SE, \\ \frac{\mathrm{d}I}{\mathrm{d}t} = \frac{E}{incu} - \beta I - \gamma I, \\ \frac{\mathrm{d}R}{\mathrm{d}t} = \beta I, \\ \frac{\mathrm{d}D}{\mathrm{d}t} = \gamma I, \\ \frac{\mathrm{d}E}{\mathrm{d}t} = \alpha_i SI + \alpha_e SE - \frac{E}{incu}. \end{cases}$$

我们根据实际意大利全国总人口、感染人群的比例、治愈人群比例、死亡人群比例和它们的日变化量，针对参数 α_e、α_i、γ、β 进行拟合。由于使用微分方程并反复迭代，我们根据措施的实行分阶段对参数进行逐步拟合，采用先大幅调整后小幅微调的方式，使图像趋向于接近实际图像。以 2020 年 2 月 22 日的意大利的情况作为初始值，开始迭代。

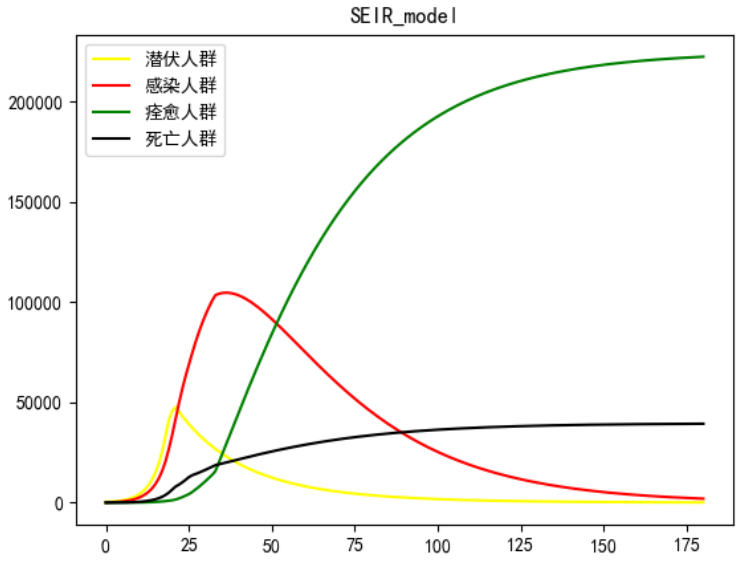

图 **1**　模型一

4.1.2 对模型一的评估

合理性

1. 医疗救援的有效程度：为了将医疗资源从“暂时满足”，到” 资源开始大规模紧缺”，再到” 得到支援补充并开始大规模且高水平救治” 的实际变化趋势展现出来，我们将其体现在治愈率 β 上，从一个较高水平，迅速大幅下降约 20∼30% 的水平，之后逐步提高，并在最后提升了 2∼3 倍的治疗效率。
2. 从微分方程的曲线可以看出治愈数、感染数的变化和实际收集到的数据反映的状况比较接近，所以该模型正确表达了治愈 (R) 和感染 ($\frac{I}{E}$) 二者之间的变化关系。
3. 其次模型充分考虑了病毒在潜伏期会传染这个特性并且引入了潜伏者，通过引入潜伏者感染率参数 α_e，使潜伏者在模型中充当了易感人群和确诊感染人群之间的桥梁。通过查阅资料得到绝大多数患者的潜伏期中位数多为 3∼7 天，我们利用这个信息，实现了易感人群到潜伏者再到确诊感染者的有效过渡。
4. 再者，在模型的迭代过程中对感染率、治愈率以及死亡率进行动态变化来表示封城措施和获得外界援助对病毒发展趋势的影响。1）为了将医疗资源从“暂时满足”，到” 资源开始大规模紧缺”，再到” 得到支援补充并开始大规模且高水平救治” 的实际变化趋势展现出来，我们将其体现在治愈率 β 上，从一个较高水平，迅速大幅下降约 20∼30% 的水平，之后逐步提高，并在最后提升了 2∼3 倍的治疗效率；2）而考虑到意大利的封城措施对疫情防控的结果有限，所以在接触感染率的下降上并不明显。据资料显示，意大利采取封城以后，部分地区的部分人群并没有认真遵守和执行封闭措施，也没有普及口罩的日常穿戴，所以病毒携带者的接触传染率 α 有所下降但仍比较高。
5. 当然，我们的模型也很好地区分开了潜伏期人群和确诊感染人群的接触感染率，分别以 α_e 和 α_i 表示，并演绎了隔离的作用。由常识可以推断，潜伏者的接触感染率将仅受到措施和医疗手段的限制，而确诊感染者被隔离后几乎不再感染，因此潜伏期人群在疫情前中期的接触感染率将持续在一定水平并远高于确诊感染者，确诊感染者仅在疫情初期造成大规模感染。对于 α_e，我们根据采取措施的时机进行极大幅度降低，而对于 α_i，我们根据措施的强度和时间的推移，分段缩减。

实用性

1. 模型较为合理的反映出了疫情的发展趋势，特别表现了采取封城和获得援助等措施的有效性，给其他地区对应疫情提供了一定的参考。
2. 模型考虑了病毒的传播特点和新冠肺炎的病理特征。此模型可以根据不同的病毒调整参数和配置，能够适应更多的传染病。

4.1.3 模型二

该模型采用迭代模拟的方法，从开始模拟至今，根据持有的数据对模型带有的参数不断优化调整，使模型能够准确表现从开始到现在疫情数据的变化规律。所以该模型的

优点就是不断根据前一天的数据进行拟合，方便我们从中更改状态，并且具有较强的拟合性。模型的原型是 SEIR 模型，利用前一天的数据在某个特定的参数空间下迭代到今日，与今日的数据相吻合。我们对于收集到的数据集，去除偏离较大的点后，利用 pyhon 语言的 numpy 库进行迭代模拟。已知 COVID-19 的平均潜伏期为 14 天，故以意大利的封城时间（3 月 10 日）后 14 天，即 3 月 24 日为分解线，对前后的数据分别进行迭代模拟。

核心迭代过程如下：

```
# 迭代过程
# 易感人群的变化来自感染者的感染和潜伏的转化以及死亡者
S[i+1] = S[i] - r1 * i_infect_s * S[i] * I[i]/N - r2 * e_infect_s * S[i] * E[i]/N - dead
    * I[i]
# 潜伏者的变化来自潜伏着转化为感染者以及易感者转化为潜伏者
E[i+1] = E[i] - e_to_i * E[i] + r1 * i_infect_s * S[i] * I[i]/N + r2 * e_infect_s * S[i]
    * E[i]/N
# 感染者的变化来自潜伏者转化为感染者以及治愈者
I[i+1] = I[i] + e_to_i * E[i] - recover * I[i]
R[i+1] = R[i] + recover * I[i]
D[i+1] = D[i] + dead * I[i]
```

每日死亡人数

1. 封城前 $y = -0.002606t^4 + 0.1831t^3 - 3.053t^2 + 18.36t - 26.47$
2. 封城后 $y = -0.00002334t^4 + 0.008104t^3 - 0.9067t + 27.87t + 588.8$

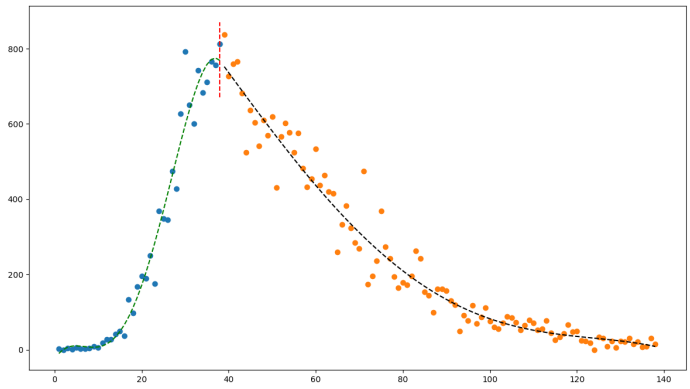

图 **2**　模型二，每日死亡人数

每日新增确诊

1. 封城前 $y = -0.01859t^4 + 1.06t^3 - 11.2t^2 + 56.29t - 4.17$

2. 封城后 $y = -0.00008293t^4 + 0.02521t^3 - 1.834t - 75.25t + 9303$

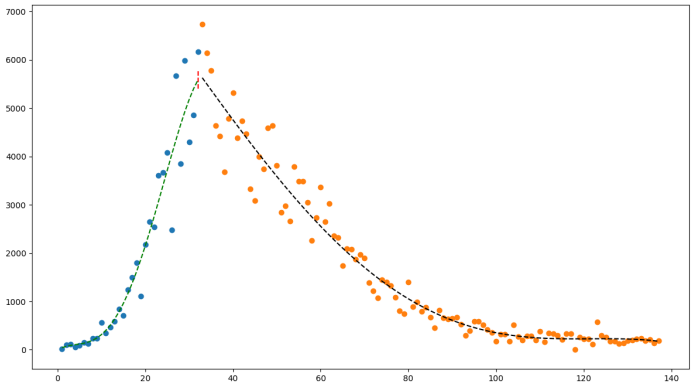

图 **3**　模型二，每日确诊人数

4.1.4 对模型二的评估

1. 合理性：此模型根据具体数据，结合实际执行封城措施的情况，将数据分为两段进行迭代模拟，实际的结果均大致分布在拟合结果的曲线附近，误差相对较小。
2. 实用性：此模型能够有效地评估政府的抗疫措施实施的效果。可以发现，执行封城措施 14 天后意大利的疫情大致迎来拐点。可以修改模型所用的封城时间评估延迟封城或提前封城带来的后果或收益。

4.2 对问题 2 的回答

4.2.1 模型一

将模型一中封城、三次医疗援助的时间均向后推迟 5 天，得到的结果如图：

可以看到，最终趋于平稳后死亡人数达到了 10 万人以上，治愈人群数量接近 80 万，也就是累计感染接近九十万，但是考虑到医疗承载能力限度问题，实际上在感染人群快速增长以后治愈率可能无法大幅上升，达到疫情的平稳期可能会耗费更长时间。由此可见，封城措施和医疗援助的效果相当显著，虽然只是 5 天的差距，预计的死亡人数增加了一倍以上，预计累计的感染人数将会增加接近三倍。

将模型一中封城、三次医疗援助的时间均向前提前 5 天，得到的结果如图：

显然，感染人群的峰值提前到了第 25 天附近，且感染人群的最大保有量下降到 3 万人以下，累计死亡人数小于 1 万，累计感染人数大约为 7 万人。由此可见封城措施和医疗援助的必要性，如果能够提前五天封城或者获得援助，将有两万人以上免于死亡，累计感染人数将会缩小六倍以上，疫情也会提前进入平稳期。除此之外，由于感染人数

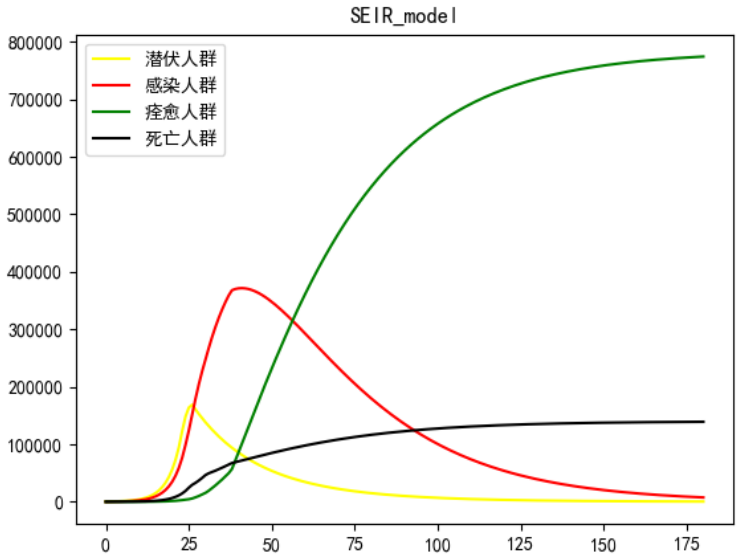

图 **4**　模型一：防疫措施延后 **5** 天

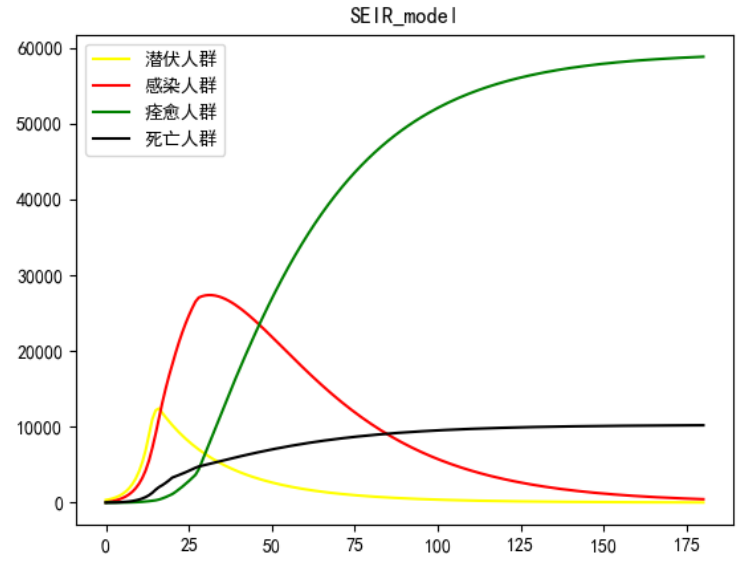

图 **5**　模型一：防疫措施提前 **5** 天

的减少，医疗系统收到的压力减小，整体的治愈率可能有所提高，死亡率极有可能降低，死亡人群的数量可能进一步减小。

4.2.2 模型二

我们使用模型二拟合得到的多项式方程，将封城的时间延后 5 天，可以看到确诊人数（图 6）和死亡人数（图 7）的峰值进一步上探，原有的下降曲线已经无法满足峰值的

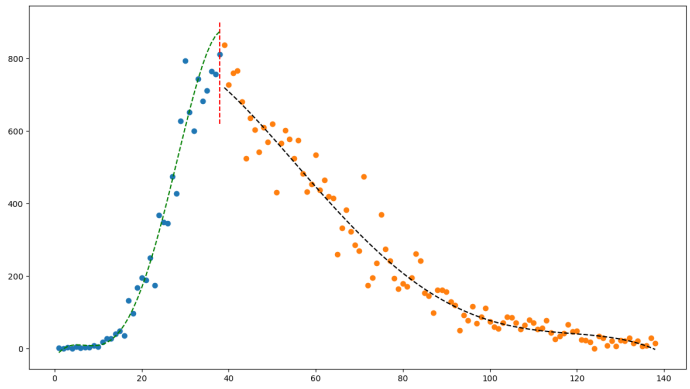

图 **6**　模型二：防疫措施延后 **5** 天的每日新增确诊人数预测

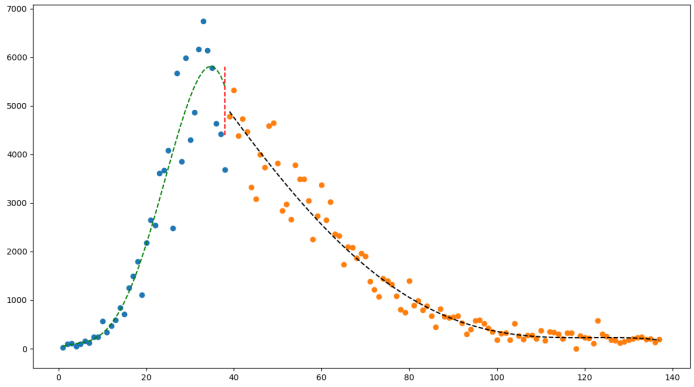

图 **7**　模型二：防疫措施延后 **5** 天的每日新增死亡人数预测

需求，也就是说拖延封城会导致医疗资源的压力进一步加大，平稳期也会进一步延后，疫情对经济生活的影响会进一步扩大。

4.3 对问题 **3** 的回答

本文选取意大利的旅游收入描述疫情对于旅游业的影响。本团队通过查询公开资料获得了意大利 2010 年至 2019 年各个月份的旅游业收入数据 [2]，以此为依据预测 2020 年的旅游业收入数据，并与实际情况进行对比。灰度预测模型 (Gray Forecast Model) 是通过少量的、不完全的信息，建立数学模型并作出预测的一种预测方法，是处理小样本数据预测问题的有效工具。由于旅游业收入数据量较小，没有规律性的分布，且我们只需要短期预测，故选择使用灰度预测模型的 GM(1,1) 模型进行旅游业收入预测。

4.3.1 等权邻值生成数

灰色系统理论认为，客观现象虽然复杂，但是必然蕴含某种内在规律。灰色系统通过对数据的整理来挖掘变化规律，即灰色序列的产生。灰色序列能够弱化随机性，凸显出规律性。假设原始数据为：$x^{(0)} = (x^{(0)}(1), x^{(0)}(2), ..., x^{(0)}(n))$，则称 $x^{(0)}(k-1)$ 和 $x^{(0)}(k)$ 为一对紧邻值，$x^{(0)}(k-1)$ 称为前值，$x^{(0)}(k)$ 称为后值。

对于常数 $\alpha \in [0,1]$，则称

$$z^{(0)}(k) = \alpha x^{(0)}(k) + (1-\alpha)x^{(0)}(k-1)$$

为由数列 $x^{(0)}$ 的邻值在生成系数（权）α 下的邻值生成数。

当 $\alpha = 0.5$ 时，称 $z^{(0)}(k) = 0.5x^{(0)}(k) + 0.5x^{(0)}(k-1)$ 为等权邻值生成数。

4.3.2 GM(1,1) 的灰微分方程模型

令 $z^{(1)}$ 为 $x^{(1)}$ 的紧邻均值数列，即，

$$z^{(1)}(k) = \alpha x^{(1)}(k) + (1-\alpha)x^{(1)}(k-1), k = 2, 3, ...n$$

则 $z^{(1)} = (z^{(1)}(2), z^{(1)}(3), ...z^{(1)}(n))$。于是定义 GM(1,1) 的灰微分方程模型为

$$d(k) + az^{(1)}(k) = b$$

即

$$x^{(0)}(k) + az^{(1)}(k) = b$$

其中 $x^{(0)}(k)$ 称为灰导数，a 称为发展系数，$z^{(1)}(k)$ 称为白化背景值，b 称为灰作用量。以 $x^{(0)} = (x^{(0)}(1), x^{(0)}(2), ..., x^{(0)}(n))$ 为数据列建立 GM(1,1) 模型，用回归分析求 a，b 的估计值，于是相应的白化模型为

$$\frac{dx^{(1)}(t)}{dt} + ax^{(1)}(t) = b$$

解为

$$x^{(1)}(t) = (x^{(0)}(1) - \frac{b}{a})e^{-ak} + \frac{b}{a}$$

于是得到预测值

$$\hat{x}^{(1)}(k+1) = (x^{(0)}(1) - \frac{b}{a})e^{-ak} + \frac{b}{a}, k = 1, 2, ..., n-1$$

故进一步得到相应的预测值

$$\hat{x}^{(0)}(k+1) = \hat{x}^{(1)}(k+1) - \hat{x}^{(1)}(k), k = 1, 2, ..., n-1$$

4.3.3 对意大利 2020 年旅游业的收入预测

本文使用 python 进行灰度预测（代码见附录），函数 GM 先求每一年某个月的原始数据的一次累加序列，再对得到的一次累加序列做紧邻值生成。构造完数据矩阵 B 之后使用最小二乘法算出白化方程的参数 a,b，通过推导出来的白化方程的解计算预测值，并且得出每个序列在进行 GM 模型运算过程中的级比和小概率误差，通过对二者作图得出模型的合理性和有效性。得到的预测值与正常值的对比图如下：

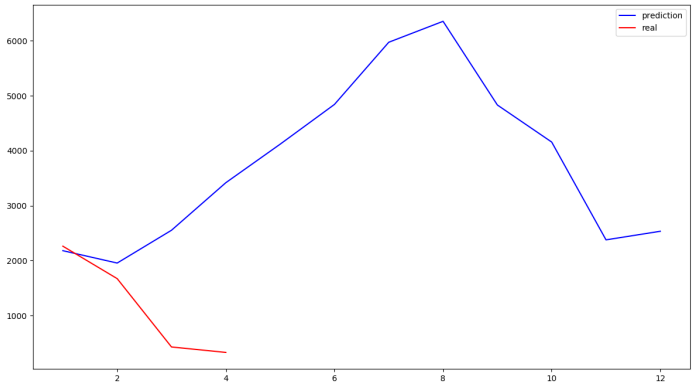

图 **8**　灰度模型预测

从前四个月的预测值和真实值来看：一月份意大利并未重视疫情，也没有对其他国家采取禁令，直到一月底禁止中国游客入境，所以图像反映的真实值和预测值较为接近。到二三四月份开始各国开始陆续爆发疫情，二月份的疫情震中在中国，并且疫情还未开始往周边国家蔓延，所以意大利的旅游收入并未产生断崖式下跌。二月份实际值和预测值的差异大概有三千万欧元，可以判断疫情对人们的出行选择造成了一定程度的影响，但是影响不大。由之前收集到的数据可知，从二月开始到四月这段时间，意大利的确诊数和死亡数增长迅速，并且欧洲各国也陆续暴发疫情，欧洲变成了疫情震中，因此旅游收入呈现断崖式下跌。从预测值和实际值的关系来看，在四月份最高下跌了约 90.2%。鉴于意大利在 5 月 4 日开始全国解封并且欧洲各国疫情相对好转，我预测接下来几个月（意大利旅游旺季）旅游收入会开始小幅回升，并且总体的趋势和预测曲线的趋势类似。根据意大利旅游联合会的研究报告，今年旺季国外游客会减少 43.4%，国内游客下降 11.6%，以中国为例，该组织预测 2020 年大约有 600 多万人次的中国游客前往意大利，人均消费 1197 欧元。所以不难得知 2020 年中国游客对意大利旅游业的贡献大概减少 60 亿欧元。根据类似数据粗略计算意大利的旅游收入会损失 170 亿欧，所以每个月的旅游收入大概会是预测曲线的 60% 左右。

级比和小概率误差图如下：

一般而言，我们认为级比 $C < 0.35$ 并且 $P > 0.95$ 我们就认为 GM 模型的精度较为

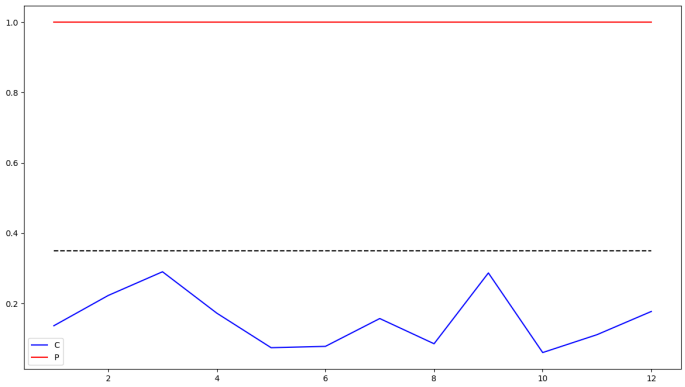

图 **9**　级比和小概率误差

优异，从图像上看级比 C 明显低于 0.35 水平线，而小误差概率 P 也满足条件，所以该模型表现优异，能够较为正确地预测意大利 2020 年每个月旅游收入的情况。

五、 总结

本文从 COVID-19 病毒的特性出发，以意大利为主要分析目标，提出了一种基于微分方程的 SEIRD 传染病模型和一种迭代模拟模型，讨论了这两种模型的合理性和实用性。基于这两种模型分析了尽早执行封城与医疗援助防疫的有效性和必要性，还基于灰度预测模型分析了意大利旅游业受到疫情的影响。

参考文献

[1] https://github.com/BlankerL/DXY-COVID-19-Data

[2] https://zh.tradingeconomics.com/italy/tourism-revenues

附录 A　模型一 SIRDED 模型代码

```
import scipy.integrate as spi
import numpy as np
import matplotlib.pyplot as plt

# 显示设置
plt.rcParams['font.sans-serif'] = 'SimHei'
plt.rcParams['axes.unicode_minus'] = False

# 初始人群
```

```
N = 60000000
# 治愈率
beta = 0.01
# 感染者传染率
alpha_i = 0.1
# 潜伏者传染率
alpha_e = 0.4
# 死亡率
gamma = 0.05

# 潜伏期
Incu = 5
# 初始感染人数
I_0 = 20
# 潜伏者人数
E_0 = 350
# 痊愈者人数
R_0 = 0
# 死亡人数
D_0 = 2
# 易感人群
S_0 = N - I_0 - R_0 - E_0 - D_0

T_start = 0
T_end = 180

def SEIR_model (inp, T_range):
    global alpha_i, alpha_e, gamma, beta, Incu
    if 19 <= T_range <= 20:
        # 封城+隔离，确诊者(I)被隔离感染率大幅下降，但是潜伏者(E)依然在接触
        # 启动全部医疗资源，治愈率上升
        alpha_i = 0.001
        alpha_e = 0.3
        beta = 0.007
    elif 21<= T_range <= 25:
        # 第一次得到医疗援助
        # 因为有试剂盒和检测手段，两种感染源的感染率都大幅下降
        # 治愈率上升，医疗资源得到补充，死亡率下降
        alpha_i = 0.001
        alpha_e = 0.15
        gamma = 0.021
        beta = 0.012
    elif 26 <= T_range <= 32:
        # 第二次
        # 医疗资源得到补充，感染率下降，治愈率上升
        alpha_i = 0.001
        alpha_e = 0.15
```

```
            gamma = 0.008
            beta = 0.016
        elif T_range >= 33:
            # 第三次
            # 治愈率再次上升
            alpha_i = 0.001
            alpha_e = 0.15
            gamma = 0.004
            beta = 0.04

        Y = np.zeros(len(inp))

        #易感人群变化率
        Y[0] = (- inp[0] * (alpha_e * inp[1] + alpha_i * inp[2]) )/N

        #潜伏人群变化率
        Y[1] = (inp[0] * (alpha_e * inp[1] + alpha_i * inp[2]))/N - (inp[1] / Incu)

        #感染人群变化率
        Y[2] = inp[1] / Incu - beta * inp[2] - inp[2] * gamma

        #痊愈人群变化率
        Y[3] = beta * inp[2]

        #死亡人数变化率
        Y[4] = inp[2] * gamma

        return Y


    Initial = (S_0, E_0, I_0, R_0, D_0)
    T_range = np.arange(T_start, T_end + 1)
    RES = spi.odeint(SEIR_model, Initial, T_range)

    # plt.plot(RES[:,0], color = 'blue', label = "易感人群")
    plt.plot(RES[:,1], color = 'yellow', label = "潜伏人群")
    plt.plot(RES[:,2], color = 'red', label = "感染人群")
    plt.plot(RES[:,3], color = 'green', label = "痊愈人群")
    plt.plot(RES[:,4], color = 'black',label = "死亡人群")
    plt.title("SEIR_model")
    plt.legend()
    plt.show()
```

附录 B　模型二迭代模拟模型代码

```
import scipy.integrate as spi
import numpy as np
import matplotlib.pyplot as plt

N = 10000000
E = [0 for i in range(0,141)]
I = [0 for i in range(0,141)]
I[0] = 1
S = [0 for i in range(0,141)]
S[0] = N - I[0]
R = [0 for i in range(0,141)]
D = [0 for i in range(0,141)]

r1 = 20 # 感染者接触易感者的人数
i_infect_s = 0.031 # 感染者能够成功感染易感者的概率
e_to_i = 0.1 # 潜伏者转变为感染者的概率
r2 = 20 # 潜伏者接触易感者的人数
e_infect_s = 0.025 # 潜伏者感染易感者的概率
recover = 0.06 # 治愈率
dead = 0.0095 #死亡率

for i in range(0,140):
    # 根据数据考虑十天之后封城，那么感染者和潜伏者接触易感者的机会大大减少，
    # 表示为接触易感者的数量变少，就不用动态表示了，
    if (i>=10):
        r1 = 2
        r2 = 5
    else:
        r1 = 20
        r2 = 20

    # 迭代过程
    # 易感人群的变化来自感染者的感染和潜伏的转化以及死亡者
    S[i+1] = S[i] - r1 * i_infect_s * S[i] * I[i]/N - r2 * e_infect_s * S[i] * E[i]/N
        - dead * I[i]
    # 潜伏者的变化来自潜伏着转化为感染者以及易感者转化为潜伏者
    E[i+1] = E[i] - e_to_i * E[i] + r1 * i_infect_s * S[i] * I[i]/N + r2 * e_infect_s
        * S[i] * E[i]/N
    # 感染者的变化来自潜伏者转化为感染者以及治愈者
    I[i+1] = I[i] + e_to_i * E[i] - recover * I[i]

    R[i+1] = R[i] + recover * I[i]
    D[i+1] = D[i] + dead * I[i]


Time = [i for i in range(1,142)]
```

```
plt.plot(Time,E,color = "yellow",label = "suspect")
plt.plot(Time,I,color = "red",label = "infect")
plt.plot(Time,R,color = "green",label = "recover")
plt.plot(Time,D,color = "black",label = "death")
plt.legend()
plt.xlabel("Time")
plt.ylabel("Varience")
plt.show()
print(E[140],I[140],R[140],D[140])
```

附录 C　问题二 SEIRD 模型模拟防疫措施推迟五天代码

```
import scipy.integrate as spi
import numpy as np
import matplotlib.pyplot as plt

# 显示设置
plt.rcParams['font.sans-serif'] = 'SimHei'
plt.rcParams['axes.unicode_minus'] = False

# 初始人群
N = 60000000
# 治愈率
beta = 0.01
# 感染者传染率
alpha_i = 0.1
# 潜伏者传染率
alpha_e = 0.4
# 死亡率
gamma = 0.05

# 潜伏期
Incu = 5
# 初始感染人数
I_0 = 20
# 潜伏者人数
E_0 = 350
# 痊愈者人数
R_0 = 0
# 死亡人数
D_0 = 2
# 易感人群
S_0 = N - I_0 - R_0 - E_0 - D_0

T_start = 0
```

```
T_end = 180

def SEIR_model (inp, T_range):
    global alpha_i, alpha_e, gamma, beta, Incu
    if 24 <= T_range <= 25:
        # 封城+隔离，确诊者(I)被隔离感染率大幅下降，但是潜伏者(E)依然在接触
        # 启动全部医疗资源，治愈率上升
        alpha_i = 0.001
        alpha_e = 0.3
        beta = 0.007
    elif 26<= T_range <= 30:
        # 第一次得到医疗援助
        # 因为有试剂盒和检测手段，两种感染源的感染率都大幅下降
        # 治愈率上升，医疗资源得到补充，死亡率下降
        alpha_i = 0.001
        alpha_e = 0.15
        gamma = 0.021
        beta = 0.012
    elif 31 <= T_range <= 37:
        # 第二次
        # 医疗资源得到补充，感染率下降，治愈率上升
        alpha_i = 0.001
        alpha_e = 0.15
        gamma = 0.008
        beta = 0.016
    elif T_range >= 38:
        # 第三次
        # 治愈率再次上升
        alpha_i = 0.001
        alpha_e = 0.15
        gamma = 0.004
        beta = 0.04

    Y = np.zeros(len(inp))

    #易感人群变化率
    Y[0] = (- inp[0] * (alpha_e * inp[1] + alpha_i * inp[2]) )/N

    #潜伏人群变化率
    Y[1] = (inp[0] * (alpha_e * inp[1] + alpha_i * inp[2]))/N - (inp[1] / Incu)

    #感染人群变化率
    Y[2] = inp[1] / Incu - beta * inp[2] - inp[2] * gamma

    #痊愈人群变化率
    Y[3] = beta * inp[2]
```

```
    #死亡人数变化率
    Y[4] = inp[2] * gamma

    return Y


Initial = (S_0, E_0, I_0, R_0, D_0)
T_range = np.arange(T_start, T_end + 1)
RES = spi.odeint(SEIR_model, Initial, T_range)

# plt.plot(RES[:,0], color = 'blue', label = "易感人群")
plt.plot(RES[:,1], color = 'yellow', label = "潜伏人群")
plt.plot(RES[:,2], color = 'red', label = "感染人群")
plt.plot(RES[:,3], color = 'green', label = "痊愈人群")
plt.plot(RES[:,4], color = 'black',label = "死亡人群")
plt.title("SEIR_model")
plt.legend()
plt.show()
```

附录 D　问题二 SEIRD 模型模拟防疫措施提前五天代码

```
import scipy.integrate as spi
import numpy as np
import matplotlib.pyplot as plt

# 显示设置
plt.rcParams['font.sans-serif'] = 'SimHei'
plt.rcParams['axes.unicode_minus'] = False

# 初始人群
N = 60000000
# 治愈率
beta = 0.01
# 感染者传染率
alpha_i = 0.1
# 潜伏者传染率
alpha_e = 0.4
# 死亡率
gamma = 0.05

# 潜伏期
Incu = 5
# 初始感染人数
I_0 = 20
# 潜伏者人数
```

```
E_0 = 350
# 痊愈者人数
R_0 = 0
# 死亡人数
D_0 = 2
# 易感人群
S_0 = N - I_0 - R_0 - E_0 - D_0

T_start = 0
T_end = 180

def SEIR_model (inp, T_range):
    global alpha_i, alpha_e, gamma, beta, Incu
    if 14 <= T_range <= 15:
        # 封城+隔离，确诊者(I)被隔离感染率大幅下降，但是潜伏者(E)依然在接触
        # 启动全部医疗资源，治愈率上升
        alpha_i = 0.001
        alpha_e = 0.3
        beta = 0.007
    elif 16<= T_range <= 20:
        # 第一次得到医疗援助
        # 因为有试剂盒和检测手段，两种感染源的感染率都大幅下降
        # 治愈率上升，医疗资源得到补充，死亡率下降
        alpha_i = 0.001
        alpha_e = 0.15
        gamma = 0.021
        beta = 0.012
    elif 21 <= T_range <= 27:
        # 第二次
        # 医疗资源得到补充，感染率下降，治愈率上升
        alpha_i = 0.001
        alpha_e = 0.15
        gamma = 0.008
        beta = 0.016
    elif T_range >= 28:
        # 第三次
        # 治愈率再次上升
        alpha_i = 0.001
        alpha_e = 0.15
        gamma = 0.004
        beta = 0.04

    Y = np.zeros(len(inp))

    #易感人群变化率
    Y[0] = (- inp[0] * (alpha_e * inp[1] + alpha_i * inp[2]) )/N
```

```
    #潜伏人群变化率
    Y[1] = (inp[0] * (alpha_e * inp[1] + alpha_i * inp[2]))/N - (inp[1] / Incu)

    #感染人群变化率
    Y[2] = inp[1] / Incu - beta * inp[2] - inp[2] * gamma

    #痊愈人群变化率
    Y[3] = beta * inp[2]

    #死亡人数变化率
    Y[4] = inp[2] * gamma

    return Y


Initial = (S_0, E_0, I_0, R_0, D_0)
T_range = np.arange(T_start, T_end + 1)
RES = spi.odeint(SEIR_model, Initial, T_range)

# plt.plot(RES[:,0], color = 'blue', label = "易感人群")
plt.plot(RES[:,1], color = 'yellow', label = "潜伏人群")
plt.plot(RES[:,2], color = 'red', label = "感染人群")
plt.plot(RES[:,3], color = 'green', label = "痊愈人群")
plt.plot(RES[:,4], color = 'black',label = "死亡人群")
plt.title("SEIR_model")
plt.legend()
plt.show()
```

附录 E　问题三灰度预测模型代码

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
pd.set_option('display.max_columns',None)
pd.set_option('display.max_rows',None)

data = pd.read_csv("Italy_Tourism.csv")
cols = list(data.columns)

def GM(x0): #自定义灰色预测函数
    x1 = x0.cumsum() #1-AGO序列
    z1 = (x1[:len(x1) - 1] + x1[1:]) / 2.0 #紧邻均值（MEAN）生成序列
    z1 = z1.reshape((len(z1),1))
    B = np.append(-z1, np.ones_like(z1), axis = 1)
    Yn = x0[1:].reshape((len(x0) - 1, 1))
```

```
    [[a],[b]] = np.dot(np.dot(np.linalg.inv(np.dot(B.T, B)), B.T), Yn) #计算参数
    f = lambda k: (x0[0] - b / a) * np.exp(-a * (k - 1)) - (x0[0] - b / a) *
        np.exp(-a * (k - 2)) #还原值
    delta = np.abs(x0 - np.array([f(i) for i in range(1,len(x0) + 1)]))
    C = delta.std() / x0.std()
    P = 1.0 * (np.abs(delta - delta.mean()) < 0.6745 * x0.std()).sum() / len(x0)
    return (f, a, b, x0[0], C, P) #返回灰色预测函数、a、b、首项、方差比、小残差概率

# 方差比列表
C_all = []
# 小残差概率列表
P_all = []

data.index = range(2010,2020)
data.loc[2020] = None
for col in cols:
    res = GM(data.loc[range(2010,2020),col].as_matrix())
    f = res[0]
    C_all.append(res[4])
    P_all.append(res[5])
    data.loc[2020,col] = f(len(data))
    data[col] = data[col].round(2)

# 做级比和小概率误差的图像
timeline_one = range(1,13)
plt.plot(timeline_one,C_all,color="blue",label="C")
plt.plot(timeline_one,P_all,color="red",label="P")
plt.hlines(0.35,1,12,linestyles="dashed")
plt.legend()
plt.show()

# 做预测值和真实值的图像并根据实际进行预测分析
real_data = pd.read_csv("2020_real.csv")
real_data = list(real_data.loc[0])
timeline_two = range(1,5)
plt.plot(timeline_one,list(data.loc[2020]),color = "blue",label="prediction")
plt.plot(timeline_two,real_data,color = "red",label = "real")
plt.legend()
plt.show()
print(data.loc[2020])
```