### **疾病传播微分方程的Python实现实验报告**

**标题：**  
 基于微分方程的疾病传播模型及其Python实现

**实验者：**084622109吴泽同 084622114任俊杰084622107陈可涵

**指导教师：**尹雪妍

**日期：**2024/12/22

#### 1. 引言

在流行病学中，疾病传播模型为理解疾病在不同人群中的传播过程提供了理论基础。常见的模型包括SIR模型（易感-感染-恢复模型）、SEIR模型（易感-暴露-感染-恢复模型）等。本报告基于SIR模型，探讨如何通过微分方程描述疾病传播，并使用Python实现该模型的数值求解。通过模拟不同参数下的疾病传播情况，评估其传播特征。

#### 2. 理论背景

疾病传播模型通常涉及微分方程来描述疾病的传播动态。以下是一些常见的疾病传播模型及其对应的微分方程：

1. **SI模型（易感者-感染者模型）**：
   * 该模型假设人群中只有易感者和感染者两类。
   * 微分方程为：
   * 其中， 表示易感者数量， 表示感染者数量， 是总人口数， 是传染率系数。

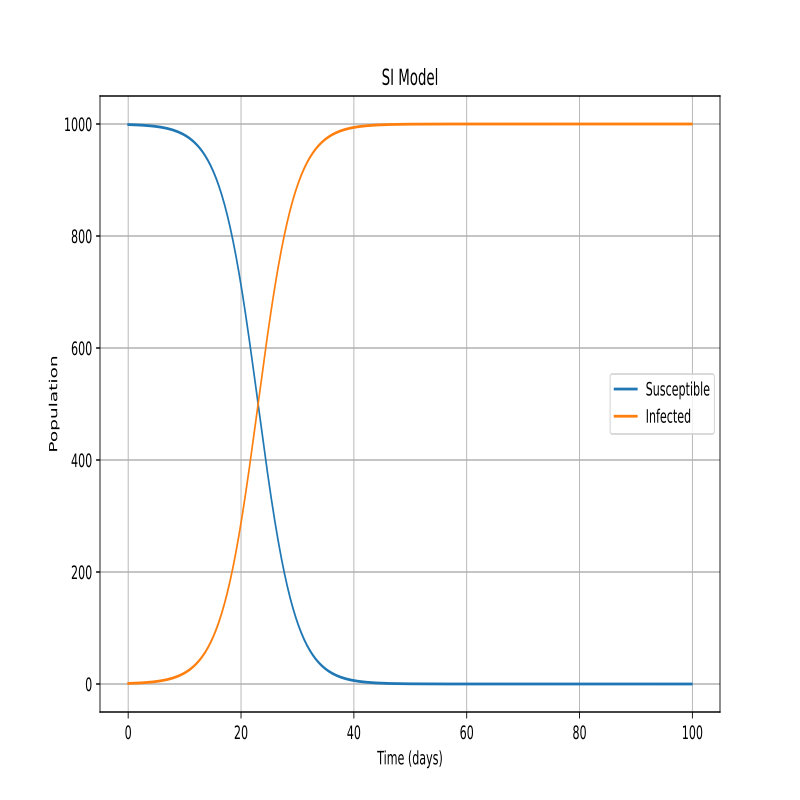


图 1

1. **SIS模型（易感者-感染者-易感者模型）**：
   * 该模型考虑了感染者恢复后再次变为易感者的情况。
   * 微分方程为：
   * 其中， 是恢复率系数。

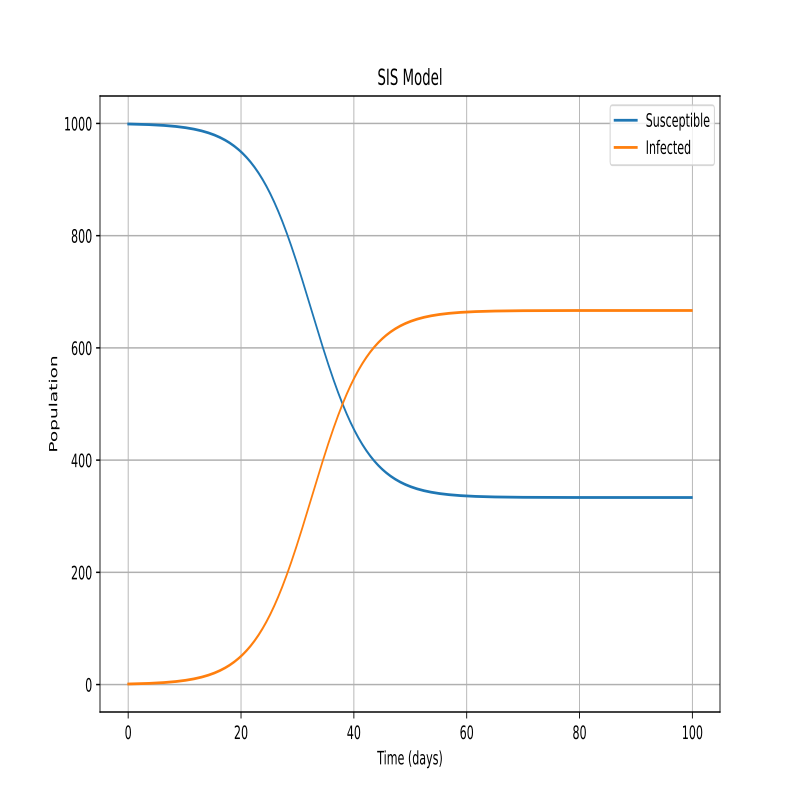
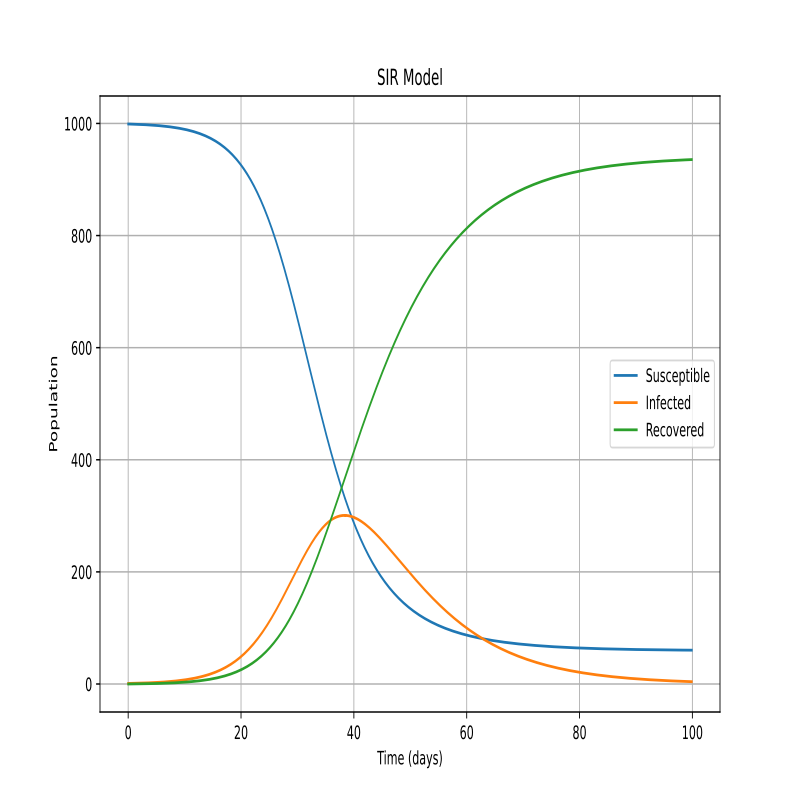


图 2

1. **SIR模型（易感者-感染者-移除者模型）**：
   * 该模型假设感染者在恢复后获得免疫力，成为移除者。
   * 微分方程为：
   * 其中， 表示移除者（康复并获得免疫力的个体）数量。

图 3

1. **SIRS模型（易感者-感染者-移除者-易感者模型）**：
   * 该模型考虑了移除者随时间失去免疫力，再次变为易感者的情况。
   * 微分方程为：
   * 其中， 是失去免疫力的率系数。

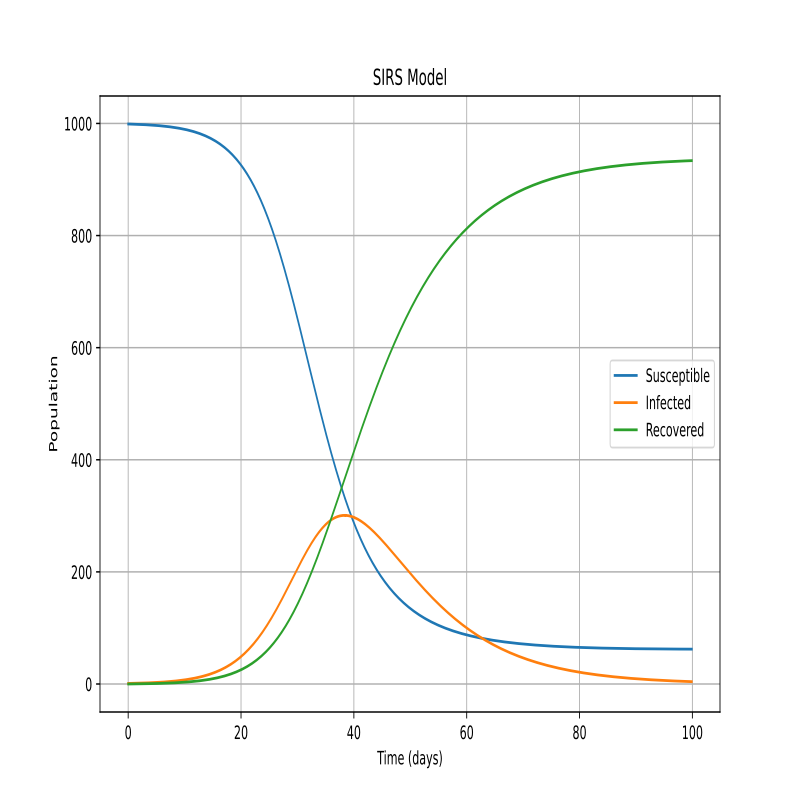
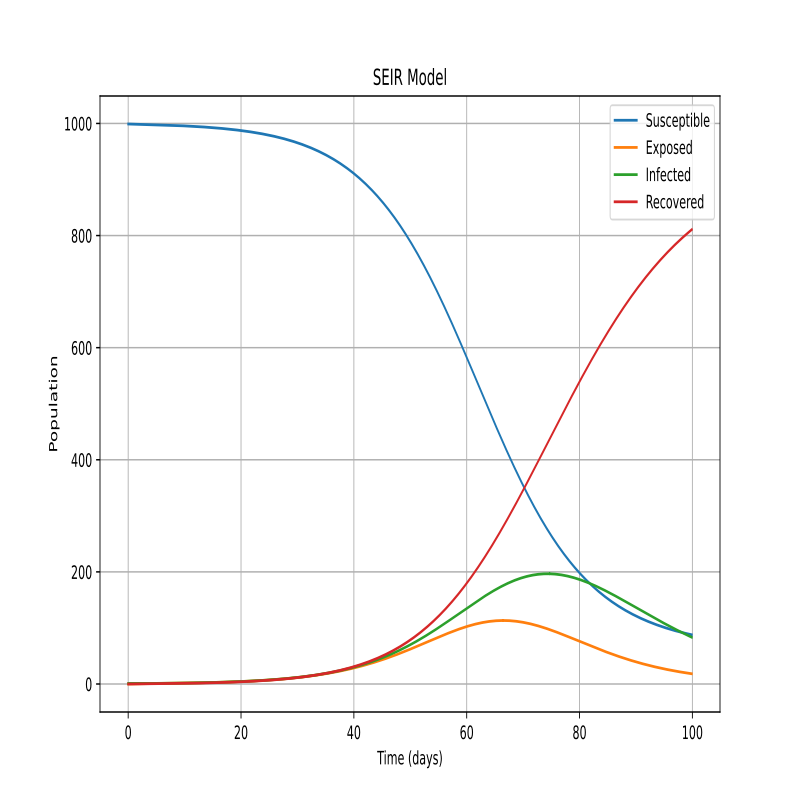


图 4

1. **SEIR模型（易感者-潜伏者-感染者-移除者模型）**：
   * 该模型考虑了潜伏期，即个体在感染后需要一段时间才会变得具有传染性。
   * 微分方程为：
   * 其中， 表示潜伏者数量， 是潜伏期。

图 5

这些模型通过微分方程描述了疾病在人群中的传播过程，可以用于预测疾病的传播趋势和评估控制措施的效果。使用Python和Matplotlib，可以对这些模型进行模拟和可视化，从而更直观地理解疾病传播的动态变化。

#### 3. 疾病传播模型的Python实现

本实验将基于SIR模型，使用Python的scipy.integrate库求解常微分方程，模拟疾病传播过程。以下是Python代码实现：

import numpy as np

from scipy.integrate import odeint

import matplotlib.pyplot as plt

# 定义微分方程

def SI\_model(y, t, beta, N):

    S, I = y

    dSdt = -beta \* S \* I / N

    dIdt = beta \* S \* I / N

    return [dSdt, dIdt]

def SIS\_model(y, t, beta, gamma, N):

    S, I = y

    dSdt = -beta \* S \* I / N + gamma \* I

    dIdt = beta \* S \* I / N - gamma \* I

    return [dSdt, dIdt]

def SIR\_model(y, t, beta, gamma, N):

    S, I, R = y

    dSdt = -beta \* S \* I / N

    dIdt = beta \* S \* I / N - gamma \* I

    dRdt = gamma \* I

    return [dSdt, dIdt, dRdt]

def SIRS\_model(y, t, beta, gamma, sigma, N):

    S, I, R = y

    dSdt = -beta \* S \* I / N + sigma \* R / N

    dIdt = beta \* S \* I / N - gamma \* I

    dRdt = gamma \* I - sigma \* R / N

    return [dSdt, dIdt, dRdt]

def SEIR\_model(y, t, beta, gamma, tau, N):

    S, E, I, R = y

    dSdt = -beta \* S \* I / N

    dEdt = beta \* S \* I / N - E / tau

    dIdt = E / tau - gamma \* I

    dRdt = gamma \* I

    return [dSdt, dEdt, dIdt, dRdt]

# 模拟函数

def simulate\_model(model, initial\_conditions, params, t\_max, dt):

    t = np.arange(0, t\_max, dt)

    solution = odeint(model, initial\_conditions, t, args=params)

    return t, solution

# 可视化函数

def plot\_model(t, solution, model\_name, labels, save\_path=None):

    plt.figure(figsize=(10, 6))

    for i, label in enumerate(labels):

        plt.plot(t, solution[:, i], label=label)

    plt.title(f'{model\_name} Model')

    plt.xlabel('Time (days)')

    plt.ylabel('Population')

    plt.legend()

    plt.grid(True)

    # 保存为 SVG 文件

    if save\_path:

        plt.savefig(f"{save\_path}\_{model\_name}.svg", format="svg")

    plt.show()

# 参数设置

N = 1000  # 总人口数

beta = 0.3  # 传染率

gamma = 0.1  # 恢复率

sigma = 0.05  # 失去免疫力率

tau = 5  # 潜伏期

# 初始条件

S0 = 999  # 初始易感者数量

I0 = 1  # 初始感染者数量

R0 = 0  # 初始移除者数量

E0 = 0  # 初始潜伏者数量

# 时间设置

t\_max = 100  # 模拟时间

dt = 0.1  # 时间步长

# 保存路径

save\_path = "disease\_model"

# 模拟和可视化 SI 模型

t, solution = simulate\_model(SI\_model, [S0, I0], (beta, N), t\_max, dt)

plot\_model(t, solution, 'SI', ['Susceptible', 'Infected'], save\_path)

# 模拟和可视化 SIS 模型

t, solution = simulate\_model(SIS\_model, [S0, I0], (beta, gamma, N), t\_max, dt)

plot\_model(t, solution, 'SIS', ['Susceptible', 'Infected'], save\_path)

# 模拟和可视化 SIR 模型

t, solution = simulate\_model(SIR\_model, [S0, I0, R0], (beta, gamma, N), t\_max, dt)

plot\_model(t, solution, 'SIR', ['Susceptible', 'Infected', 'Recovered'], save\_path)

# 模拟和可视化 SIRS 模型

t, solution = simulate\_model(SIRS\_model, [S0, I0, R0], (beta, gamma, sigma, N), t\_max, dt)

plot\_model(t, solution, 'SIRS', ['Susceptible', 'Infected', 'Recovered'], save\_path)

# 模拟和可视化 SEIR 模型

t, solution = simulate\_model(SEIR\_model, [S0, E0, I0, R0], (beta, gamma, tau, N), t\_max, dt)

plot\_model(t, solution, 'SEIR', ['Susceptible', 'Exposed', 'Infected', 'Recovered'], save\_path)

#### 4. 实验设计与实现步骤

1. **模型选择**：选择SIR模型来描述基本的疾病传播过程，设置初始条件和模型参数。
2. **微分方程求解**：使用odeint函数进行数值求解，得出不同时间点上的易感者、感染者和恢复者的比例。
3. **参数调整**：通过修改β和γ的值，模拟不同的疾病传播速度和恢复速率，观察结果变化。
4. **数据可视化**：使用matplotlib绘制易感者、感染者和恢复者的变化曲线，直观展示疾病传播过程。

#### 5. 实验结果

通过数值求解，我们得到了疾病传播的不同阶段表现。以下是典型实验结果的图示：

* **S曲线**：随着时间的推移，易感者数量逐渐减少。
* **I曲线**：感染者数量在某个时刻达到最大值后逐渐下降。
* **R曲线**：恢复者数量随着感染者的康复而增加，最终趋于稳定。

通过调整传染率β和恢复率γ，我们可以观察到疾病传播速度的变化。例如，当β增大时，感染者人数会迅速增加，而恢复者人数也会随着时间增加。

#### 6. 参数分析与讨论

1. **传染率β**：增大β会导致疾病传播更加迅速，感染者在短时间内激增。
2. **恢复率γ**：提高γ（即加快恢复过程）能够缩短疫情的持续时间。
3. **初始感染者比例**：不同的初始感染者比例会影响最终的感染人数和传播速度。

实验表明，当传染率和初始感染者比例较高时，疾病传播会迅速达到峰值，恢复者人数也会相应增加。

#### 7. 结论

通过本实验的数值模拟，我们成功实现了基于SIR模型的疾病传播模拟，并通过调整不同参数观察了其对疾病传播速度和范围的影响。该模型可以为流行病学研究提供定量分析工具，并为公共卫生策略提供支持。

#### 8. 未来工作

* **更多复杂模型**：如SEIR模型或SIRS模型，加入暴露期或免疫周期，进行更精确的疾病传播模拟。
* **数据驱动**：结合实际流行病数据，调整模型参数以提高模型的预测准确性。
* **大规模模拟**：使用更多的计算资源进行大规模模拟，探讨不同地区、不同时间段的传播情况。

#### 9. 参考文献

* Kermack, W. O., & McKendrick, A. G. (1927). A Contribution to the Mathematical Theory of Epidemics. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 115(772), 700-721.
* Anderson, R. M., & May, R. M. (1992). *Infectious Diseases of Humans: Dynamics and Control*. Oxford University Press.