传染病模型

Example

问题的提出：医生们发现，在一个民族或地区，当某种传染病流传时，波及到的总人数大体上保持为一个常数。即既非所有人都会得病也非毫无规律，两次流行（同种疾病）的波及人数不会相差太大。如何解释这一现象呢？试用建模方法来加以证明。

指数模型

定义已感染人数为$i(t)$，假设每个病人单位时间有效接触（足以使人致病）的人数为λ，那么，在时间段Δt内，病人的增量可以用如下的公式进行计算：

$$i(t+\Delta t)-i(t)=\lambda i(t)\Delta t \tag{21}$$

将等式右侧的Δt除到等式左侧，并 取极限$\Delta t \to 0$

$$\lim_{\Delta t \to 0} \frac{i(t+\Delta t)-i(t)}{\Delta t}=\lambda i(t) \tag{22}$$

写成微分方程

$$\frac{\mathrm{d}i}{\mathrm{d}t}=\lambda t \tag{23}$$

记初始时刻的病人人数为$i(0)=i_0$，那么我们可以得到指数增长的传染病模型

$$\begin{cases} \dfrac{\mathrm{d}i}{\mathrm{d}t}=\lambda t \\ \\ i(0)=i_0 \end{cases} \tag{24}$$

指数模型

实际上，这与人口增长的指数模型是一致的，我们直接给出其解析解

$$i(t) = i_0 e^{\lambda t} \tag{25}$$

实际上，除了解析解之外，我们还可以使用欧拉前向差分（Forward Euler Method）的方法近似求解上述方程。将上述公式中的$i(t)$移到等式的右边，我们得到递推公式

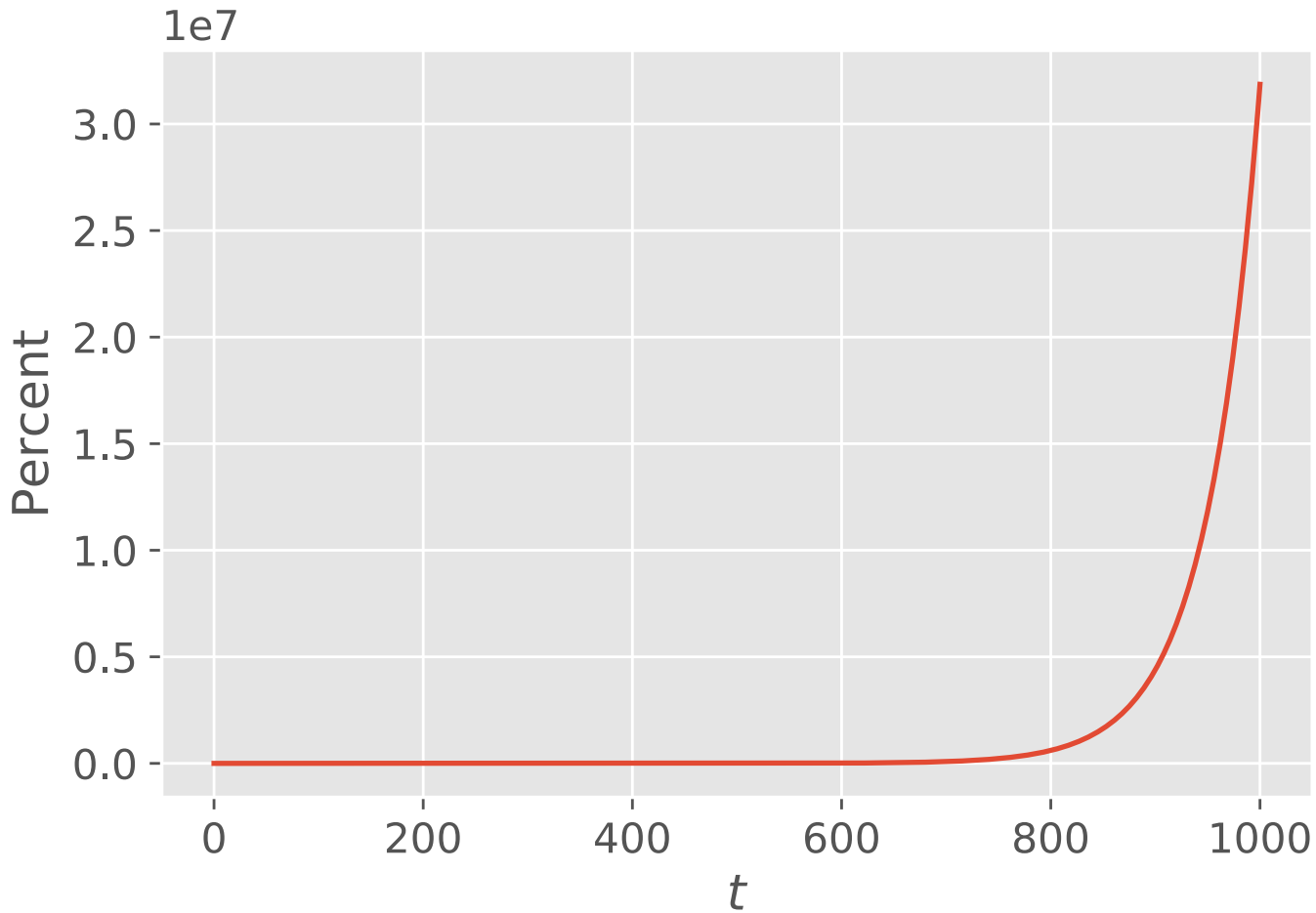
$$i(t + \Delta t) = i(t) + \lambda i(t) \Delta t \tag{26}$$

以上递推公式意味着，我们可以通过当前时刻的病人人数$i(t)$和致病参数λ，计算得到Δt时间后的病人人数$i(t + \Delta t)$.以上思想可以在Python中进行实现，代码如下。

指数模型

```
def exp():
    deltaT = 0.01
    lamb = 2
    i_list = []
    i0 = 0.08; # 初始有的人感染8%
    i_list.append(i0) # 输入i0
    Tot_Time = 10
    TotStep = int(Tot_Time/deltaT) # 表示取整int
    ## 递推地求解差分方程
    for i in range(TotStep):
        i_new = i_list[-1] + lamb * i_list[-1] *
            deltaT
        i_list.append(i_new) # 输入，进入循环i_new
    plt.plot(i_list)
    plt.xlabel(r"$t$") #加横坐标label
    plt.ylabel('Percent') #加纵坐标label
exp()
plt.show()
```

指数模型



指数模型

思考

从中我们可以看到病人的增长是指数级的，在短短十天后，已经有3000万人患病！这显然不符合实际情况的，那么问题出在哪里了呢?

实际上，若病人~~解除~~接触的是病人，并不能够使病人再次患病，实际上以上算法导致了**重复计数现象**的发生。

解决办法：**必须区分已感染者和未感染者。**

SI模型

现在我们将人群分成两个群体：**已感染者（病人，Infected）和未感染者（健康者，Suspect）**，该模型称为SI模型。

模型假设：

- 在研究时间内，不考虑死亡率和出生率，即总人数N不变，病人和健康人的比例分别为$i(t)$和$s(t)$.
- 每个病人在单位时间内有效接触并致病的人数为λ，且只有接触健康人才会致病，称λ为日接触率。

SI模型

仿照指数模型里面的建模方法，在时间段Δt内，病人的增量可以用如下的公式进行计算:

$$N[i(t+\Delta t)-i(t)]=[\lambda s(t)]Ni(t)\Delta t \tag{27}$$

其中的$\lambda s(t)$项可以理解为是打折扣以后的传播系数。随着健康人比例的下降，这个系数也会相应下降。 取极限$\Delta t \to 0$,

$$\frac{\mathrm{d}i}{\mathrm{d}t}=\lambda si \tag{28}$$

因为$i(t)+s(t)=1$,所以上式可以变形为

$$\frac{\mathrm{d}i}{\mathrm{d}t}=\lambda i(1-i) \tag{29}$$

SI模型

结合初始条件，我们得到了SI模型的微分方程

$$\begin{cases} \dfrac{\mathrm{d}i}{\mathrm{d}t} = \lambda i(1-i) \\ \\ i(0) = i_0 \end{cases} \tag{30}$$

思考

实际上，这里的SI模型就是我们之前讲到过的人口增长的logistic模型，请你思考如何证明？

$$\frac{\mathrm{d}x}{\mathrm{d}t} = rx(1 - \frac{x}{x_m}),$$

SI模型

这里我们重点考虑用差分方法求解这个方程。在上述公式中，消去N，再将$i(t)$移到等式的右边，我们得到如下的欧拉前向法递推公式：

$$i(t+\Delta t) = i(t) + \lambda i(t) s(t) \Delta t \tag{31}$$

同样地，我们可以通过当前时刻的病人人数和致病参数λ，计算得到Δt时间后的病人人数，将以上思想在Python中进行实现，代码如下。

SI模型

```
def SI():
    # 输入计算参数
    deltaT = 0.01
    lamb = 2
    i_list = []
    s_list = []
    i0 = 0.08; # 初始有的人感染8%
    i_list.append(i0)
    s_list.append(1 - i0)
    Tot_Time =5
    TotStep = int(Tot_Time/deltaT)
```

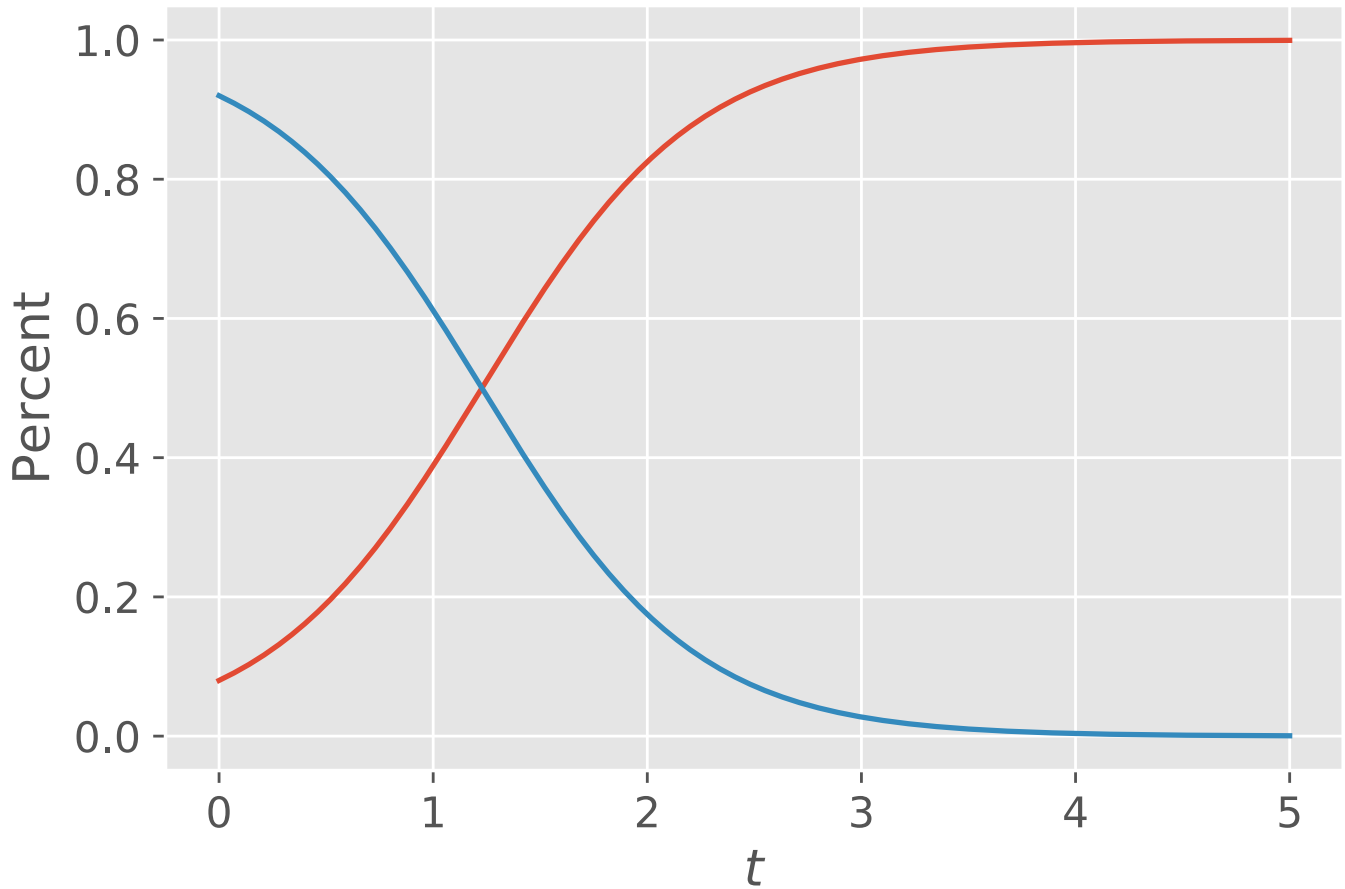
SI模型

$$i(t+\Delta t) = i(t) + \lambda i(t) s(t) \Delta t$$

```
## 递推地求解差分方程
    for i in range(TotStep):
        i_new = i_list[-1] + lamb * i_list[-1] *
            deltaT * s_list[-1]
        i_list.append(i_new)
        s_list.append(1- i_new)
    Time = [i * deltaT for i in range(TotStep + 1)]

    ## 可视化传染过程
    plt.plot(Time,i_list,label = 'i(t)') # 作i(t)图像
    plt.plot(Time,s_list,label = 's(t)') # 作s(t)图像
    plt.xlabel(r"$t$") # 加横坐标label
    plt.ylabel('Percent') # 加纵坐标label
    plt.legend() # 绘图
```

SI模型



SI模型

从SI模型我们可以看到，病人比例不再会出现**指数爆炸**的情况，在$t \to \infty$时最大患病比例为1。在SI模型中，病人数量的增长曲线是一个典型的S型曲线，又称为Logistic曲线，该曲线在生物学上经常被用来描述物种的增长模式。

思考

然而，SI模型的结论告诉我们，**无论λ多么小，最终人群都会患病**，这显然也是不符合实际情况的。这主要是因为病人患病后无法被治愈。

SIS模型

现在我们继续将人群分成两个群体：已感染者（病人，**Infected**）和未感染者（健康者，**Suspect**），但是病人可以被治愈，该模型称为SIS模型。

模型假设：

- 在研究时间内，不考虑死亡率和出生率，即总人数N不变，病人和健康人的比例分别为$i(t)$和$s(t)$.
- 每个病人在单位时间内有效接触并致病的人数为λ，且只有接触健康人才会致病，称λ为日接触率。
- **病人每天治愈的比例为μ，称为日治愈率。**

SIS模型

SI: $N[i(t+\Delta t)-i(t)]=[\lambda s(t)]Ni(t)\Delta t$

在时间段Δt内，病人的增量可以用如下的公式进行计算

$$\text{SIS: } N[i(t+\Delta t)-i(t)]=[\lambda s(t)]Ni(t)\Delta t-\mu Ni(t)\Delta t \tag{32}$$

其中新加入的 $\mu Ni(t)\Delta t$代表 Δt时间内治愈的病人数。
取极限$\Delta t\to 0$

$$\frac{\mathrm{d}i}{\mathrm{d}t}=\lambda i(1-i)-\mu i \tag{33}$$

结合初始条件，我们得到了SIS模型的微分方程

$$\begin{cases}\dfrac{\mathrm{d}i}{\mathrm{d}t}=\lambda i(1-i)-\mu i\\ \\ i(0)=i_0\end{cases} \tag{34}$$

SIS模型

这个模型难以求出解析解，我们继续采用差分近似的方法求解。在上述公式中消去N，再将$i(t)$移到等式的右边，我们得到如下的递推公式

$$i(t+\Delta t) = i(t) + \lambda i(t)s(t)\Delta t - \mu i(t)\Delta t \tag{35}$$

同样地，我们可以通过当前时刻的病人人数和致病参数λ,以及治愈参数μ，计算得到Δt时间后的病人人数，将以上思想在Python中进行实现，代码如下：

SIS模型

```
import matplotlib.pyplot as plt
def SIS():
    i0 = 0.01; # 初始有的人感染1%
    lamb = 2;
    s0 =  1- i0; # 初始未感染人数比例
    mu = 1.2;
    i_list = []
    s_list = []
    i_list.append(i0) # 输入i0
    s_list.append(s0) # 输入s0
    TotTime = 100
    TimeStep = 0.01
    TotStep = int(TotTime/TimeStep) # 表示取整数int
```

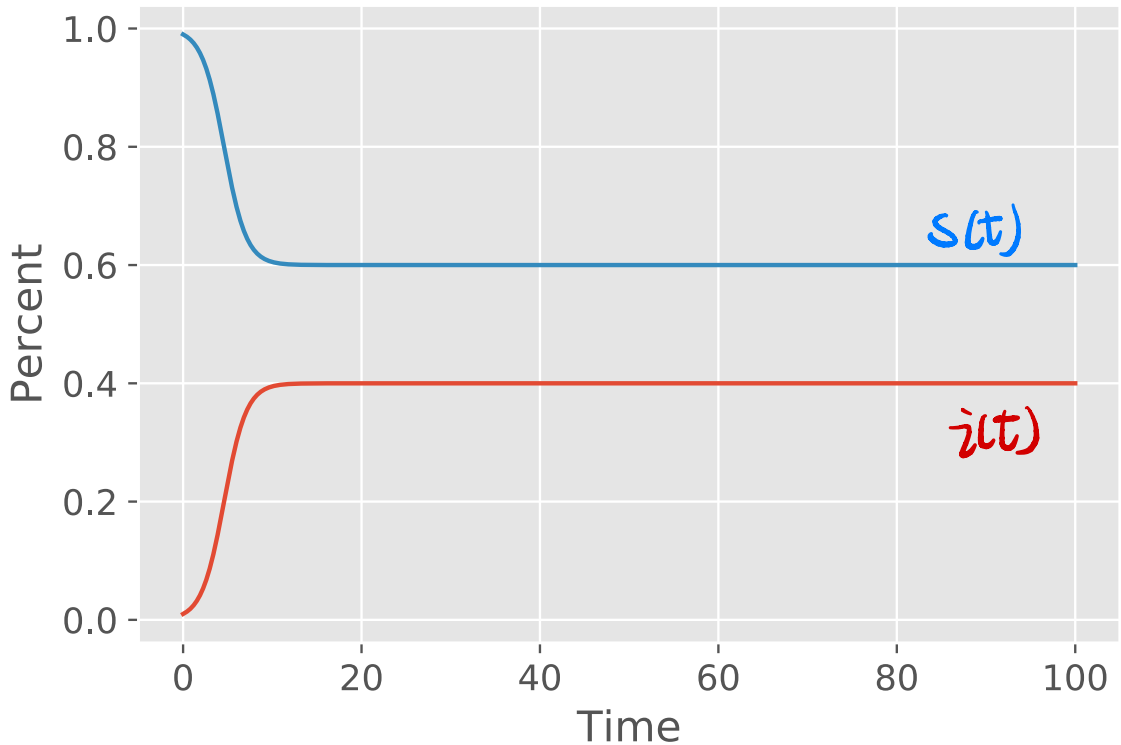
SIS模型

$$i(t+\Delta t) = i(t) + \lambda i(t) s(t) \Delta t - \mu i(t) \Delta t$$

```
## 递推地求解差分方程
    for i in range(TotStep):
        i_new = i_list[-1] + lamb * s_list[-1] *
            i_list[-1]*TimeStep - mu * i_list[-1]*
            TimeStep
        i_list.append(i_new) # 输入，继续循环i_new
        s_list.append(1-i_new) # 输入(1-i_new)，继续
            循环
    Time = [TimeStep * i for i in range(TotStep+1)]

    ## 可视化传染过程
    plt.plot(Time,i_list) # i(t)
    plt.plot(Time,s_list) # s(t)
    plt.xlabel("Time") # 加横坐标label
    plt.ylabel("Percent") # 加纵坐标label
    plt.legend(["i(t)","s(t)"])
SIS()
```

SIS模型



可以看到，病人的人数和健康人的人数会稳定在一个固定值，并非所有的人都会患病，这个稳定值与 λ和μ的取值有关。
我们也可以把SI模型看做是SIS模型在$\mu = 0$时候的特例。

SIR模型

有的传染病具有免疫性，病人治愈后即移出系统，称为移出者。
我们将人群分成三个群体：已感染者（病人，**Infected**）、未感染者（健康者，**Suspect**）和免疫者（**Removed**），病人被治愈后永久免疫。 SIS: $N[i(t+\Delta t)-i(t)]=[\lambda s(t)]Ni(t)\Delta t-\mu Ni(t)\Delta t$
在时间段Δt内，病人的增量可以用如下的公式进行计算

$$\text{SIR: } N[i(t+\Delta t)-i(t)]=[\lambda s(t)]Ni(t)\Delta t-\mu Ni(t)\Delta t \tag{36}$$

健康者的增量为 SIS: $N[s(t+\Delta t)-s(t)]=-N[i(t+\Delta t)-i(t)]$

$$\text{SIR: } N[s(t+\Delta t)-s(t)]=-[\lambda s(t)]Ni(t)\Delta t \tag{37}$$

取极限$\Delta t\to 0$，得到如下的微分方程

$$\begin{cases}\dfrac{\mathrm{d}i}{\mathrm{d}t}=\lambda si-\mu i\\[2ex] \dfrac{\mathrm{d}s}{\mathrm{d}t}=-\lambda si\\[2ex] i(0)=i_0,s(0)=s_0\end{cases} \tag{38}$$

SIR模型

```
def SIR():
    i0 = 0.01; # 初始有的人感染1%
    r0 = 0; # 初始没有人免疫
    lamb = 1.8 # 参数
    s0 =  1- i0; # 初始未感染人数比例
    mu = 0.7; # 参数
    i_list = []
    s_list = []
    r_list = []
    i_list.append(i0) # 输入i0
    s_list.append(s0) # 输入s0
    r_list.append(r0) # 输入r0
    TotTime = 20
    TimeStep = 0.01
    TotStep = int(TotTime/TimeStep) # 表示取整数int
```

SIR模型

$$N[i(t+\Delta t)-i(t)]=[\lambda s(t)]Ni(t)\Delta t-\mu Ni(t)\Delta t$$

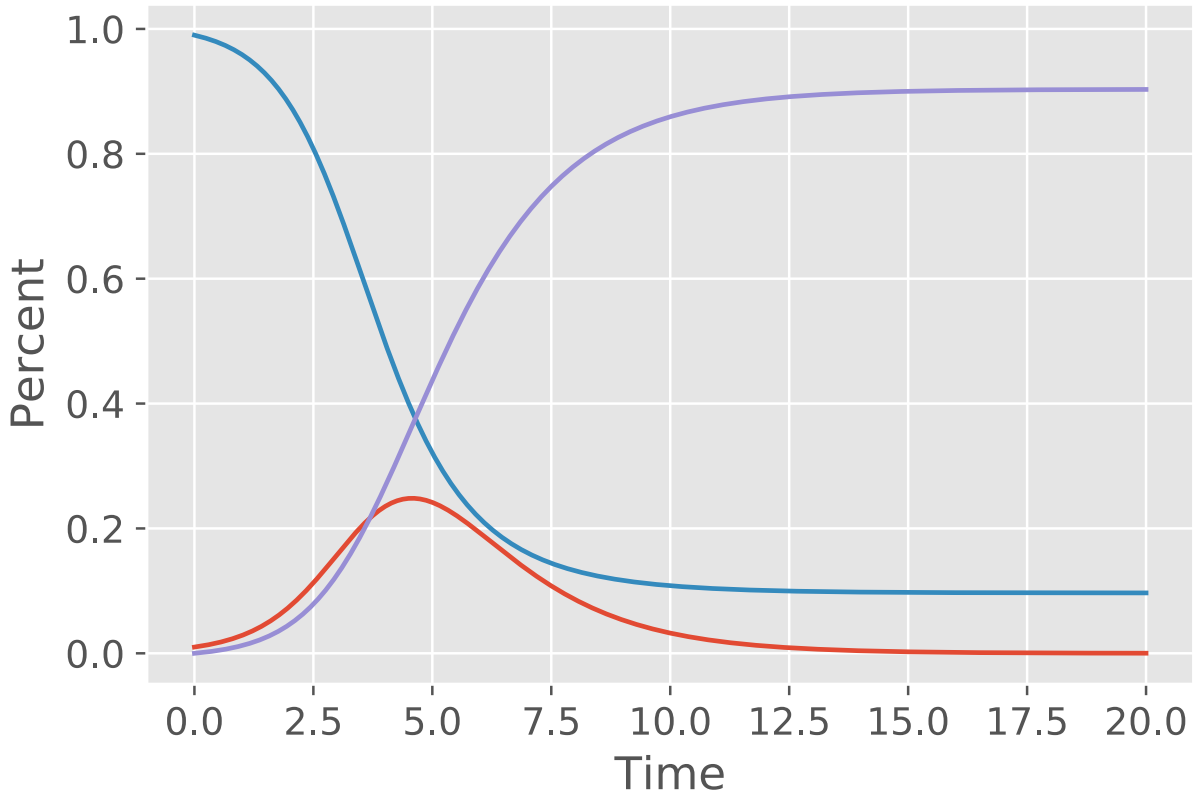
```
## 递推地求解差分方程
   for i in range(TotStep):
      i_new = i_list[-1] + lamb * s_list[-1] *
          i_list[-1]*TimeStep - mu * i_list[-1]*
          TimeStep
       s_new = s_list[-1] - lamb * s_list[-1] *
          i_list[-1]*TimeStep
       i_list.append(i_new) # 输入，继续循环i_new
       s_list.append(s_new) # 输入，继续循环s_new
       r_list.append(1-i_new-s_new) # 输入1-i_new-，
          继续循环s_new
   Time = [TimeStep * i for i in range(TotStep+1)]
```

$$N[s(t+\Delta t)-s(t)]=-[\lambda s(t)]Ni(t)\Delta t$$

SIR模型

```
## 可视化传染过程
    plt.plot(Time,i_list) # i(t)
    plt.plot(Time,s_list) # s(t)
    plt.plot(Time,r_list) # r(t)
    plt.xlabel("Time") # 加横坐标label
    plt.ylabel("Percent") # 加纵坐标label
    plt.legend(["i(t)","s(t)","r(t)"]) # 绘图
SIR()
```

SIR模型

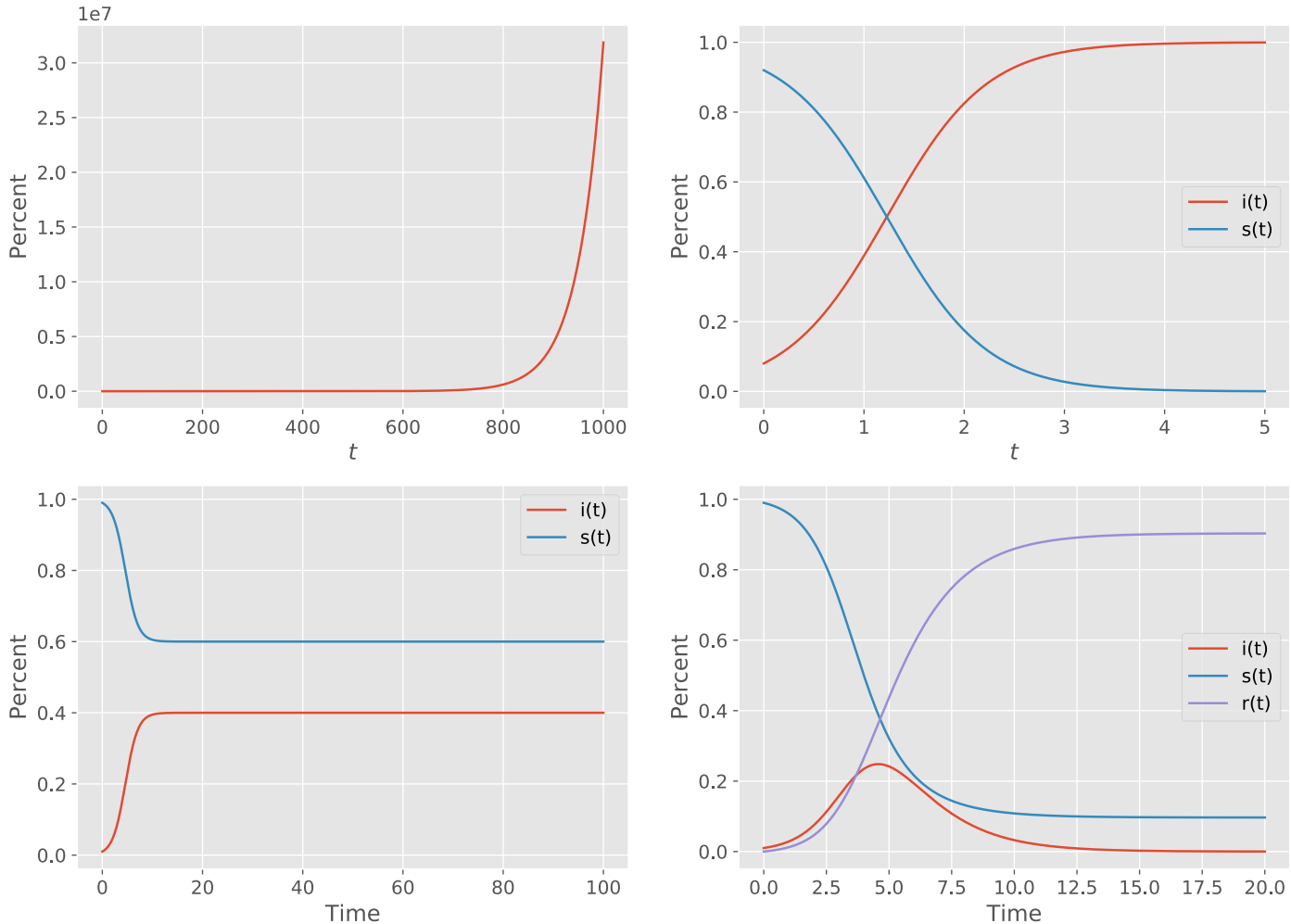


引入免疫者以后，感染人数会先增后减，有一个峰值，传染病最终后消失。

计算结果的对比

```
plt.figure(figsize=(14,10))
plt.subplot(2,2,1)
exp()
plt.subplot(2,2,2)
SI()
plt.subplot(2,2,3)
SIS()
plt.subplot(2,2,4)
SIR()
```

计算结果的对比


1e7
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Percent
0
200
400
600
800
1000
t
1.0
0.8
0.6
0.4
0.2
0.0
Percent
i(t)
s(t)
0
1
2
3
4
5
t
1.0
0.8
0.6
0.4
0.2
0.0
Percent
i(t)
s(t)
0
20
40
60
80
100
Time
1.0
0.8
0.6
0.4
0.2
0.0
Percent
i(t)
s(t)
r(t)
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Time

Volterra 模型[1]

猎物 - 捕食者模型

一战期间，人们捕获的鲨鱼比例大幅上升，~~可以按照尝试~~，由于战争，人们停止捕捞，应该普通的鱼类和鲨鱼数量都会上升，为什么单单鲨鱼数量上升如此明显呢？ 为了解释这一问题，生物学家D. Ancona 向数学家 Volterra 求助， Volterra 借用微分方程理论，成功地解释了这个现象。

猎物

~~食饵（食用鱼）~~和捕食者（鲨鱼）在时刻t的数量分别记为$x(t), y(t)$，方便起见，你也可以用“大鱼吃小鱼”的比喻来理解这个模型。

Figure: Caption

[1]本模型作为进阶内容，仅作了解即可

Volterra 模型

因为大海中资源丰富，假设当食饵独立生存时以指数规律增长，增长率为r,于是

$$\frac{\mathrm{d}x}{\mathrm{d}t} = rx \tag{39}$$

但是捕食者的存在，使得食饵的增长率减小，设减少率与捕食者的数量成正比，于是$x(t)$满足方程

$$\frac{\mathrm{d}x}{\mathrm{d}t} = x(r - ay) = rx - axy \tag{40}$$

比例系数a反映捕食者捕食食饵的能力。

捕食者离开食饵无法生存，设它独自存在时死亡率为d,即

$$\frac{\mathrm{d}y}{\mathrm{d}t} = -dy \tag{41}$$

而食饵的存在为捕食者提供了食物，相当于使捕食者的死亡率降低，且促使其增长。设增长率与食饵数量成正比，于是$y(t)$满足

$$\frac{\mathrm{d}y}{\mathrm{d}t} = y(-d + bx) = -dy + bxy \tag{42}$$

比例系数b反映食饵对捕食者的供养能力

Volterra 模型

将以上推导的结果写在一起，得到

$$\begin{cases} \dfrac{\mathrm{d}x}{\mathrm{d}t} = x(r - ay) = rx - axy \\ \dfrac{\mathrm{d}y}{\mathrm{d}t} = y(-d + bx) = -dy + bxy \end{cases} \tag{43}$$

这就是自然环境中食饵和捕食者之间的依存和制约的关系。这里没有考虑种群自身的阻滞增长作用，是Volterra提出的最简单的模型。
数值递推公式为

$$\begin{cases} x(t + \Delta t) = x(t) + \Delta t\,[rx(t) - ax(t)y(t)] \\ y(t + \Delta t) = y(t) + \Delta t\,[-dy(t) + bx(t)y(t)] \end{cases} \tag{44}$$

Volterra 模型

```
## Volterra 模型的数值求解示例
# 输入初始参数
r = 1
d = 0.5
a = 0.1
b = 0.02
x0 = 5
y0 = 12
deltaT = 0.01
TotTime = 30
timeStep = TotTime/deltaT
xlist = []
ylist = []
xlist.append(x0) # 输入x0
ylist.append(y0) # 输入y0
TimeList = [i*deltaT for i in range(int(timeStep))]
```

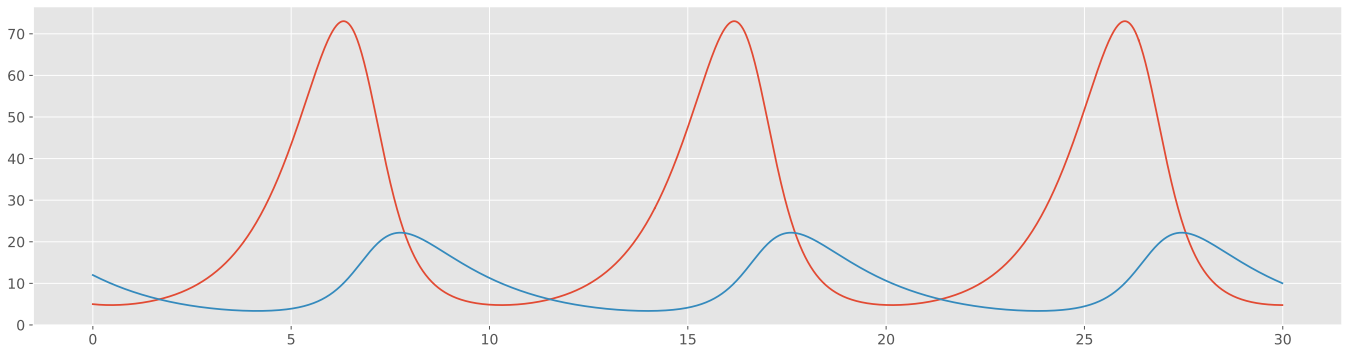
Volterra 模型

$$\begin{cases} x(t+\Delta t) = x(t) + \Delta t\left[rx(t) - ax(t)y(t)\right] \\ y(t+\Delta t) = y(t) + \Delta t\left[-dy(t) + bx(t)y(t)\right] \end{cases}$$

```
# 递推地求解差分方程
for time in TimeList:
    xlist.append(xlist[-1] +deltaT * (r *xlist[-1] -
        a*xlist[-1]*ylist[-1] ) )
    ylist.append(ylist[-1] +deltaT * (-d *ylist[-1]
        + b*xlist[-1]*ylist[-1] ) )

# 变化过程可视化
plt.figure(figsize = (20,5))
plt.plot(TimeList,xlist[:-1],label = 'Small Fish')
plt.plot(TimeList,ylist[:-1],label = 'Large Fish')
plt.legend()
```

Volterra 模型



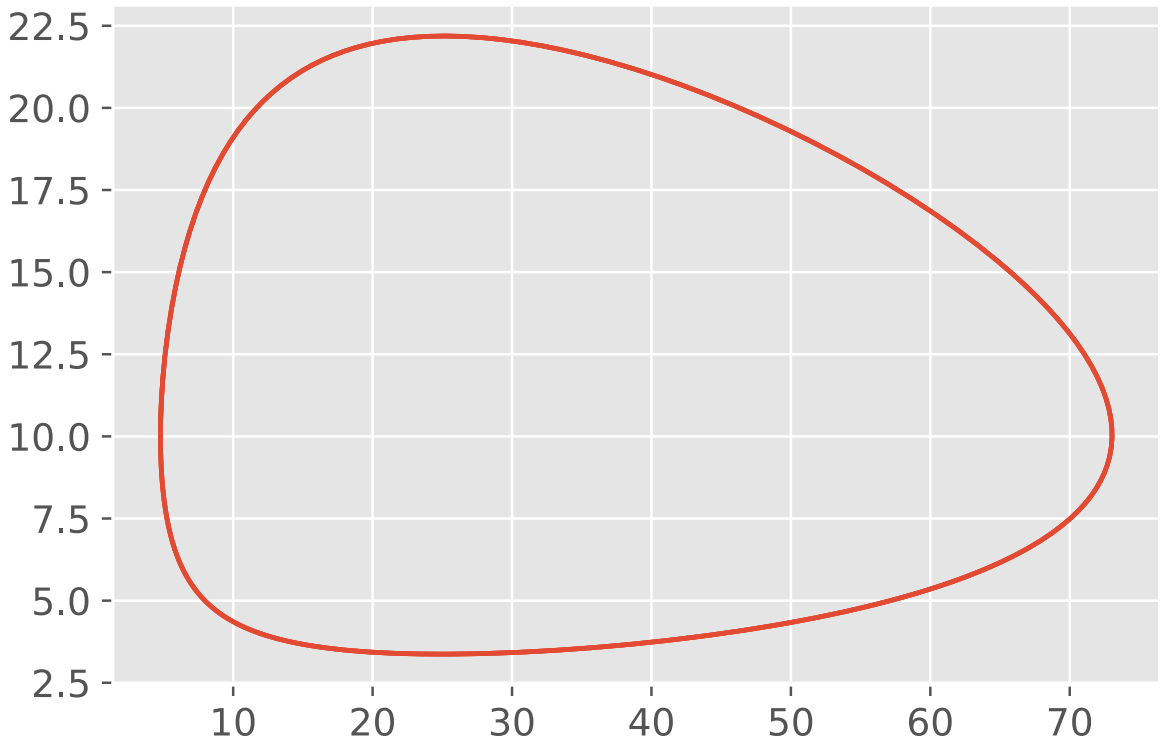
如何求大鱼和小鱼各自的平均值？

Volterra 模型

可以猜测，$x(t), y(t)$是周期函数，与此相应地，相轨线$y(x)$是封闭曲线。

```
# 绘制相轨线
plt.plot(xlist,ylist)
```

Volterra 模型



Volterra 模型

在数值解中我们看到，$x(t), y(t)$一个周期的平均值为$\overline{x} = 25, \overline{y} = 10$, 这个数值与稳定平衡点刚好相等。

$$\overline{x} = x_0 = \frac{d}{b}, \quad \overline{y} = y_0 = \frac{r}{a} \tag{45}$$

当然这也是能够通过解析方法来证明的，

$$x(t) = \frac{1}{b}\left(\frac{\dot{y}}{y} + d\right) \tag{46}$$

$$\bar{x} = \frac{1}{T}\int_0^T x(t)\mathrm{d}t = \frac{1}{T}\left[\frac{\ln y(T) - \ln y(0)}{b} + \frac{dT}{b}\right] = \frac{d}{b} \tag{47}$$

类似的，我们有：

$$\bar{y} = \frac{r}{a} \tag{48}$$

Volterra 模型

$$\overline{x} = x_0 = \frac{d}{b}, \quad \overline{y} = y_0 = \frac{r}{a}$$

注意到，r, d, a, b在生态学上的意义，上述结果表明：

- **捕食者的数量**(用一个周期内的平均值$\overline{y}$代替)与食饵增长率r成正比，与他掠食食饵的能力a成反比；
- **食饵的数量**（用一个周期内的平均值$\overline{x}$代替）与捕食者死亡率d成正比，与他供养捕食者的能力b成反比。

Volterra 模型

Volterra用这个模型来解释生物学家D'Ancona提出的问题：
我们在上面的模型中引入人为捕捞的影响，引入捕捞量系数e，相当于食饵增长率由r下降为$r-e$，而捕食者死亡率由d上升为$d+e$,用$\overline{x_1}, \overline{y_1}$表示这种情况下食用鱼（食饵）和鲨鱼（捕食者）的平均数量，则套用上面的公式可知

$$\overline{x_1} = \frac{d+e}{b} \quad \overline{y_1} = \frac{r-e}{a} \tag{49}$$

显然，$\overline{x_1} > x_1, y_1 < \overline{y_1}$.
战争期间捕获量下降，即捕获系数变为$e' < e$，于是食用鱼和鲨鱼的数量变为

$$\overline{x_2} = \frac{d+e'}{b} \quad \overline{y_2} = \frac{r-e'}{a} \tag{50}$$

显然，$\overline{x_2} < \overline{x_1}, \overline{y_2} > \overline{y_1}$，这正说明战争期间鲨鱼的比例会有明显的增加。

加入logistic项的Volterra模型

尽管Volterra模型可以解释一些现象，但是它作为近似反映现实对象的一个数学模型，必然存在不少局限性. 比如，许多生态学家指出，多数食饵－捕食者系统都观察不到Volterra模型 显示的那种周期震荡，而是趋向某种平衡状态，即系统存在稳定平衡点. 实际上， 只要在Volterra模型中加入考虑自身阻滞作用的logistic项，就可以模拟这一现象。

$$\begin{aligned}\dot{x}_1(t) &= r_1 x_1\left(1-\frac{x_1}{N_1}-\sigma_1\frac{x_2}{N_2}\right)\\ \dot{x}_2(t) &= r_2 x_2\left(-1+\sigma_2\frac{x_1}{N_1}-\frac{x_2}{N_2}\right)\end{aligned} \tag{51}$$

捕食者对猎物的影响

猎物对捕食者的影响

加入logistic项的Volterra模型

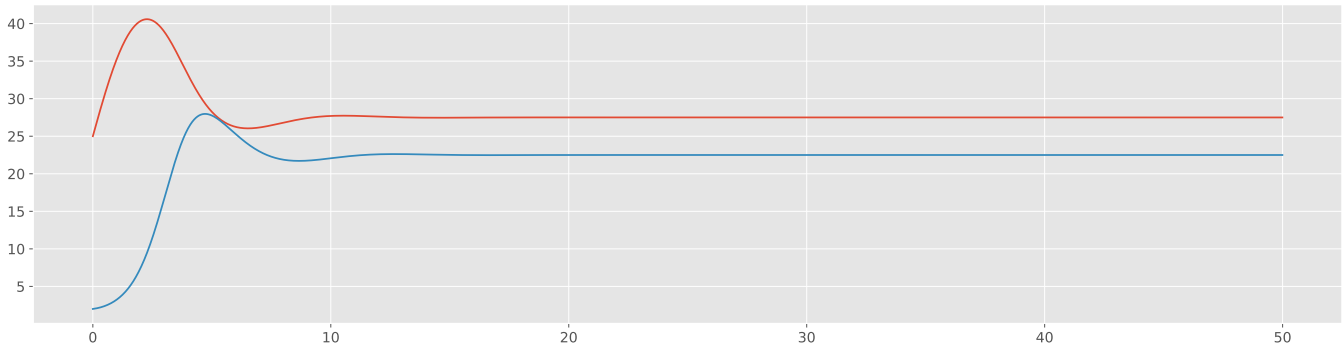
```
# 输入初始参数
r1 = 0.2
r2 = 0.1
N1 = 50
N2 = 60
sigma1 = 1.2
sigma2 = 2.5
x0 = 25
y0 = 2
deltaT = 0.01
TotTime = 50
timeStep = TotTime/deltaT
xlist = []
ylist = []
xlist.append(x0) # 输入x0
ylist.append(y0) # 输入y0
TimeList = [i*deltaT for i in range(int(timeStep))]
```

加入logistic项的Volterra模型

```
# 递推地求解差分方程
for time in TimeList:
    xlist.append(xlist[-1] +deltaT * (r *xlist[-1])
        *(1 - xlist[-1]/N1 - sigma1 * ylist[-1]/N2))
    ylist.append(ylist[-1] +deltaT * (r *ylist[-1])
        *(-1 +sigma2* xlist[-1]/N1 - ylist[-1]/N2))

# 结果可视化
plt.figure(figsize = (20,5))
plt.plot(TimeList,xlist[:-1])
plt.plot(TimeList,ylist[:-1])
```

加入logistic项的Volterra模型



课后作业

作业

考虑种群竞争模型

$$\begin{cases} \dfrac{\mathrm{d}x_1}{\mathrm{d}t} = r_1 x_1 (1 - \dfrac{x_1}{N_1} - \sigma_1 \dfrac{x_2}{N_2}) \\ \dfrac{\mathrm{d}x_2}{\mathrm{d}t} = r_2 x_2 (1 - \sigma_2 \dfrac{x_1}{N_1} - \dfrac{x_2}{N_2}) \end{cases} \tag{52}$$

取$r_1 = 0.2, r_2 = 0.3, \sigma_1 = 1.2, \sigma_2 = 0.5, N_1 = 100, N_2 = 70, x_1(0) = 30, x_2(0) = 40$,使用本节课程学到的数值方法研究两个种群的发展模式。

Thank You

· 微分方程模型