大学生数学建模培训

2024年美国大学生数学建模竞赛培训课程

机理分析类(微分方程，数值模拟）算法基本原理及应用（一）（二）

主讲人：王老师

本节目录 | 主要内容

MATH 数模加油站

目录
CONTENTS

1 基本概念

2 Python求解微分方程

3 马尔萨斯人口模型

4 阻滞增长模型

5 地中海鲨鱼问题

6 种群相互竞争模型

微分方程 | 基本概念

微分方程的定义：含导数或微分的方程称为微分方程，其一般形式为 $f\left(x,y,y',y'',\cdots,y^{(n)}\right)=0$。

微分方程的阶数：微分方程中所含导数或微分的最高阶数称为微分方程的阶数。例如：$y'''+2y''-2x=0$ 就是三阶微分方程。

微分方程的解：使得微分方程成立的函数称为微分方程的解。例如x^2和x^2+1都是一阶微分方程$y'-2x=0$ 的解。

微分方程 | 基本概念

微分方程的通解和特解：不含任意常数的解称为微分方程的特解；若解中所含的相互独立的任意常数的个数 与微分方程的阶数相等，称此解为方程的通解。例如一阶微分方程$y' - e^x = 0$ 的特解为e^x, 其通解为$e^x + C$。

初值条件：能够确定通解中任意常数的条件称为微分方程的初值条件.例如：若一阶微分方程满足$y' - e^x = 0$ 且$y(0) = 1$,则通解$e^x + C$ 中的C可以确定为 0.

SciPy库中的scipy.integrate模块提供了多种微分方程求解器，允许我们求解常微分方程（ODEs）。我们将重点介绍如何利用这些求解器，并解释相关函数的参数。

函数	问题类型	精确度	说明
ode45	非刚性	中等	采用算法为 4-5 阶 Runge-Kutta 法，大多数情况下首选的函数。
ode23	非刚性	低	基于 Bogacki-Shampine2-3 阶 Runge-Kutta 公式，在精度要求不高的场合，以及对于轻度刚性方程，ode23 的效率可能好于 ode45。
ode113	非刚性	低到高	基于变阶次 Adams-Bashforth-MoutlonPECE 算法。在对误差要求严格的场合或者函数本身计算量很大情况下比 ode45 效率高。ode113 可以看成一个多步解算器，因为它会利用前几次时间节点上的解计算当前时间节点的解。因此它不适应于非连续系统。
ode15s	刚性	低到中	基于数值差分公式（后向差分公式，BDFs 也叫 Gear 方法），因此效率不是很高。同 ode113一样，ode15s 也是一个多步计算器。当 ode45 求解失败，或者非常慢，并且怀疑问题是刚性的，或者求解微分代数问题时可以考虑用 ode15s。
ode23s	刚性	低	基于修正的二阶 Rosenbrock 公式。由于是单步解算器，当精度要求不高时，它效率可能会高于 ode15s。它可以解决一些 ode15s 求解起来效率不太高的刚性问题。
ode23t	适度刚性	低	对于仅仅是刚度适中的问题，使用该函数效果较好。
ode23tb	刚性	低	当方程是刚性的，并且求解要求精度不高时使用。

表中提到了刚性问题和非刚性问题，这是什么意思呢？大体上说，有的微分方程，未知函数随时间(自变量)的变化比较缓慢；而有的微分方程中未知函数在某些时间段内会发生剧烈的变化，前者是非刚性问题，后者是刚性问题。

常用ODE求解器：odeint 函数

odeint 是SciPy中一个常用的函数，用于求解初值问题。它的接口相对简单，适合于快速求解简单的ODE问题。

参数解释

func: 微分方程函数或方程组。它必须是一个函数，该函数接受至少两个输入参数。第一个参数是一个数组，表示当前的因变量值；第二个参数是一个标量，表示当前的自变量值（通常是时间）。

y0: 数组类型，表示初始条件。

t: 数组类型，表示积分的时间点，第一个元素必须是初始时间。

```
from scipy.integrate import odeint
# 微分方程函数
def model(y, t):
    k = 0.3
    dydt = -k * y
    return dydt
# 初始条件
y0 = 5
# 时间点
t = np.linspace(0, 20, 100)
# 求解ODE
result = odeint(model, y0, t)
# 输出结果
print(result)
```

solve_ivp 函数

solve_ivp 是一个更强大的ODE求解器，提供了更多选项和灵活性。

参数解释

fun: 微分方程函数，与odeint中的func相似，但它的第一个参数是标量（当前的自变量值），第二个参数是数组（当前的因变量值）。

t_span: 二元组类型，表示积分的时间区间（起始时间和结束时间）。

y0: 数组类型，表示初始条件。

method: 字符串类型（可选），积分方法，例如'RK45'（默认），'RK23'，'DOP853'，'BDF'等。

```
from scipy.integrate import solve_ivp
# 微分方程函数
def model(t, y):
    k = 0.3
    dydt = -k * y
    return dydt
# 初始条件
y0 = [5]
# 时间区间
t_span = (0, 20)
# 求解ODE
sol = solve_ivp(model, t_span, y0, t_eval=np.linspace(0, 20, 100))
# 输出结果
print(sol.y)
```

(1)人口预测模型

1789年，英国神父Malthus在分析了一百多年间人口统计资料之后，提出了Malthus模型。

模型假设：

1.设$x(t)$表示t时刻的人口数，且$x(t)$连续可微。

2.人口的增长率r是常数(增长率=出生率-死亡率)。

3.人口数量的变化是封闭的，即人口数量的增加与减少只取决于人口中个体的生育和死亡，且每一个体都具有同样的生育能力与死亡率。

MATH 数模加油站

建模与求解：

由假设，t时刻到t+△t时刻人口的增量为 $x(t+\triangle t)-x(t)=rx(t)\triangle t$,

可得：

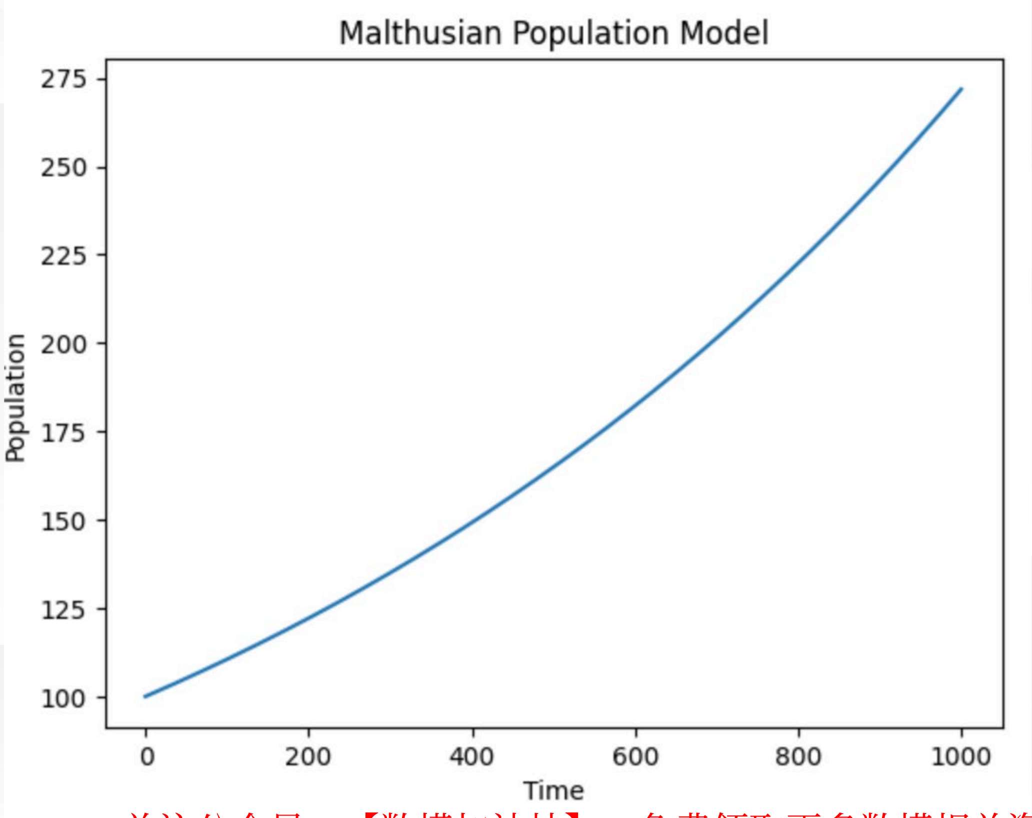
$$\begin{cases}\dfrac{\mathrm{d}x}{\mathrm{d}t}=rx,\\ x(0)=x_0,\end{cases}$$

其解为：$x(t)=x_0\mathrm{e}^{n}$

微分方程 | 马尔萨斯人口模型

```python
from scipy.integrate import odeint
import numpy as np
import matplotlib.pyplot as plt
# 定义马尔萨斯人口模型的微分方程
def malthusian_model(P, t, r):
    dPdt = r * P
    return dPdt
# 定义初始条件和参数
P0 = 100  # 初始人口数量
t = np.linspace(0, 1000, 1000)  # 时间范围
r = 0.001  # 人口增长率
# 求解微分方程
P = odeint(malthusian_model, P0, t, args=(r,))
# 绘制人口随时间变化的图像
plt.plot(t, P)
plt.xlabel('Time')
plt.ylabel('Population')
plt.title('Malthusian Population Model')
plt.show()
```

微分方程 | 马尔萨斯人口模型



模型评价：

考虑二百多年来人口增长的实际情况，1961年世界人口总数为3.06×10°,在1961年—1970年这段时间内，每年平均的人口自然增长率为2%,则公式可写为：

$$x(t) = 3.06 \times 10^{9} \cdot \mathrm{e}^{0.02(t-1961)}$$

根据1700—1961年间世界人口统计数据，发现这些数据与计算结果相当符合。因为在这期间全球人口大约每35年增加1倍，而用式(6.16)算出每34.6年增加1倍。

但是，利用公式对世界人口进行预测，也会得出令人惊异的结论，当t=2670年时，$x(t)=4.4\times10^{15}$,即4400万亿，这相当于地球上每平方米要容纳至少20人。

显然，用这一模型进行预测的结果远高于实际人口增长，误差的原因是对增长率r的估计过高。由此，可以对r是常数的假设提出疑问。

如何对增长率r进行修正呢?我们知道，地球上的资源是有限的，它只能提供一定数量的生命生存所需的条件。随着人口数量的增加，自然资源环境条件等对人口再增长的限制作用将越来越显著。如果在人口较少时可以把增长率r看成常数，那么当人口增加到一定数量之后，就应当视r为一个随着人口的增加而减小的量，即将增长率r表示为人口$x(t)$的函数$r(x)$,且$r(x)$为x的减函数。

模型假设:

1.设$r(x)$为x的线性函数， $r(x)=r-sx$(工程师原则，首先用线性)。

2.自然资源与环境条件所能容纳的最大人口数为x_m。即当$x=x_m$时，增长率$r(x_m)=0$。

建模与求解：

由假设1、2可得$r(x)=r\left(1-\frac{x}{x_m}\right)$，则有

$$\begin{cases}\dfrac{\mathrm{d}x}{\mathrm{d}t}=r\left(1-\dfrac{x}{x_m}\right)x, \\ x(t_0)=x_0\text{。}\end{cases}$$

上式是一个可分离变量的方程，其解为

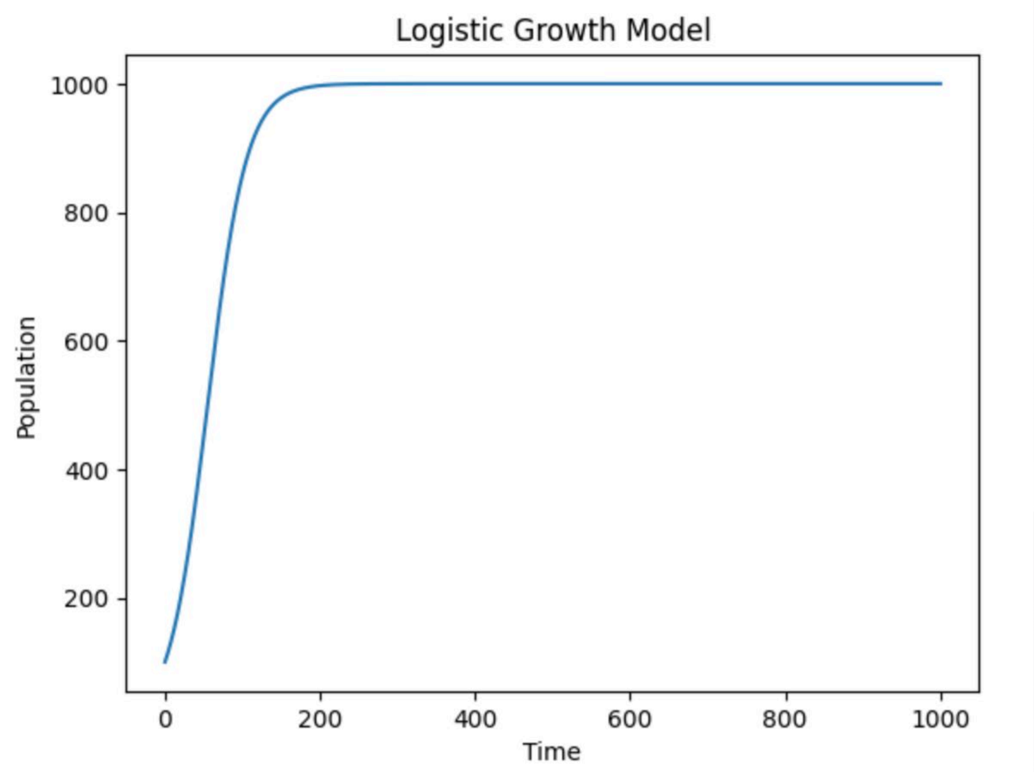
$$x(t)=\frac{x_m}{1+(\frac{x_m}{x_0}-1)\mathrm{e}^{-r(t-t_0)}}\text{。}$$

微分方程 | 阻滞增长模型

```
# 定义阻滞增长模型的微分方程
def logistic_growth_model(P, t, r, K):
    dPdt = r * P * (1 - P / K)
    return dPdt
# 定义初始条件和参数
P0 = 100  # 初始人口数量
t = np.linspace(0, 1000, 1000)  # 时间范围
r = 0.04  # 人口增长率
K = 1000  # 环境容量
```

```
# 求解微分方程
P = odeint(logistic_growth_model, P0, t, args=(r, K))
# 绘制人口随时间变化的图像
plt.plot(t, P)
plt.xlabel('Time')
plt.ylabel('Population')
plt.title('Logistic Growth Model')
plt.show()
```

微分方程 | 阻滞增长模型



Logistic Growth Model
1000
800
600
400
200
Population
0
200
400
600
800
1000
Time

对方程求解二阶导：

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2}=r^2(1-\frac{x}{x_m})(1-\frac{2x}{x_m})x。$$

1. $\lim_{x\to 0}x(t) = x_m$,即无论人口初值x_0 如何，人口总数都以x_m 为极限。

2.当$0<x<x_m$ 时，$\frac{\mathrm{d}x}{\mathrm{d}t}=r\left(1-\frac{x}{x_m}\right)x>0$,这说明$x(t)$ 是单调增加的。又由 式

知，当$x<\frac{x_m}{2}$时，$\frac{\mathrm{d}^2x}{\mathrm{d}t^2}>0, x=x(t)$ 为凹函数;当$x>\frac{x_m}{2}$时，$\frac{\mathrm{d}^2x}{\mathrm{d}t^2}<0, x=x(t)$ 为凸函数。

3. 人口变化率$\frac{\mathrm{d}x}{\mathrm{d}t}$在$x=\frac{x_m}{2}$时取到最大值，即人口总数达到极限值一半以前是加速生长时期，经过这一点之后，生长速率会逐渐变小，最终达到 0。

地中海鲨鱼问题

意大利生物学家D'Ancona曾致力于鱼类种群相互制约关系的研究。从第一次世界大战期间(1914年7月28日—1918年11月11日),地中海各港口捕获的几种鱼类捕获量百分比的资料中，发现鲨鱼等捕食者的比例有明显增加，而供其捕食的食用鱼的百分比却明显下降。显然战争使捕鱼量下降，食用鱼增加，鲨鱼等也随之增加，但为何鲨鱼的比例却有特别大幅的增加呢?

他无法解释这个现象，于是求助于著名的意大利数学家V.Volterra,希望能建立一个捕食者—猎物系统的数学模型，定量地回答这个问题。

Volterra模型(沃尔泰拉模型)

食饵(即猎物)和捕食者在时刻t的数量分别记作$x_1(t)$和$x_2(t)$.因为大海中资源丰富，可以假设如果出食饵独立生存则将以增长率r_1按指数规律增长，即有$\frac{dx_1}{dt}=r_1x_1$,捕食者的存在使得食饵的增长率降低，假设降低的程度正比于捕食者的数量，于是$x_1(t)$满足方程：

$$\frac{dx_1}{dt}=x_1(r_1-\lambda_1x_2)$$

其中：比例系数λ_1反映捕食者掠取食饵的能力

捕食者离开食饵无法生存，若假设它独自存在时的死亡率为r_2,即食饵为它提供食物的作用相当于使其死亡率降低。假设这个作用与食饵数量成正比，于是$x_2(t)$满足方程：

$$\frac{dx_2}{dt} = x_2(-r_2 + \lambda_2 x_1)$$

其中：比例系数λ_2反映食饵对捕食者的供养能力

方程(1)、(2)是在没有人工捕获情况下的自然环境中食饵与捕食者之间的制约关系，是数学家Volterra提出的最简单的模型。

在考虑人工捕获时，假设表示捕获能力或强度的系数为e,那么相当于食饵的自然增长率由r_1降为r_1-e,捕食者的死亡率由r_2增为r_2+e.方程变为：

$$\begin{cases} \dfrac{dx_1}{dt} = x_1[(r_1 - e) - \lambda_1 x_2] \\ \dfrac{dx_2}{dt} = x_2[-(r_2 + e) + \lambda_2 x_1] \end{cases}$$

MATH 数模加油站

现在我们取一组具体的数据进行分析：假设食饵和捕食者的初始数量分别 $x_1(0)=25, x_2(0)=2$,

假设战前的捕获强度为0.3,战后的捕获强度为0.1,另假设$r_1=1, \lambda_1=0.1, r_2=0.5, \lambda_2=0.02$,那么:

$$\begin{cases} \dfrac{dx_1}{dt} = x_1[(1-0.3)-0.1x_2] \\ \dfrac{dx_2}{dt} = x_2[-(0.5+0.3)+0.02x_i] \\ x_i(0)=25, x_2(0)=2 \end{cases}$$

$$\begin{cases} \dfrac{dx_1}{dt} = x_1[(1-0.1)-0.1x_2] \\ \dfrac{dx_2}{dt} = x_2[-(0.5+0.1)+0.02x_1] \\ x_i(0)=25, x_2(0)=2 \end{cases}$$

微分方程 | 地中海鲨鱼问题

MATH 数模加油站

定义Volterra模型的微分方程

def volterra_model(X, t, alpha, beta, delta, gamma, c):

    P, V = X

    dNdt=2(t/)

    dPdt = alpha * P - beta * P * V - c * P

    dVdt = delta * P * V - gamma * V

    return [dPdt, dVdt]

定义初始条件和参数

P0 = 100 # 初始猎物数量

V0 = 20 # 初始捕食者数量

t = np.linspace(0, 10, 100) # 时间范围

alpha = 1.0 # 猎物增长率

beta = 0.1 # 捕食者对猎物的影响系数

delta = 0.1 # 猎物对捕食者的影响系数

gamma = 1.0 # 捕食者死亡率

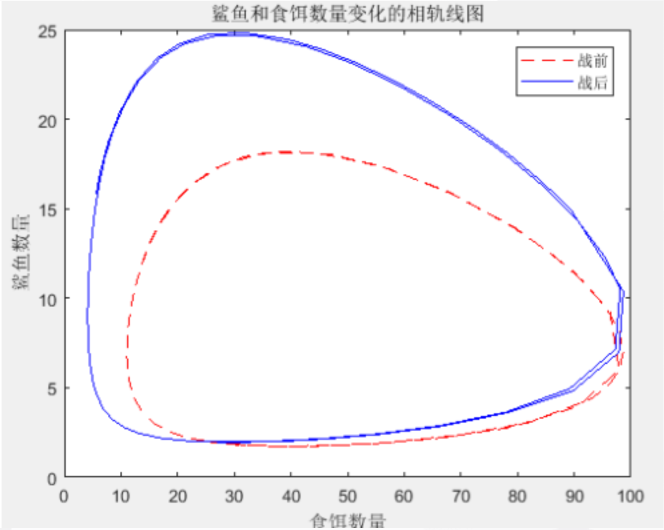
c = 0.05 # 捕获强度

微分方程 地中海鲨鱼问题

```
# 求解微分方程
X0 = [P0, V0]  # 初始状态
sol = odeint(volterra_model, X0, t,
args=(alpha, beta, delta, gamma, c))
P, V = sol[:, 0], sol[:, 1]
```

```
# 绘制猎物和捕食者数量随时间变化的图像
plt.plot(t, P, label='Prey')
plt.plot(t, V, label='Predator')
plt.xlabel('Time')
plt.ylabel('Population')
plt.title('Volterra Model with Capture Intensity')
plt.legend()
plt.show()
```

微分方程 | 地中海鲨鱼问题

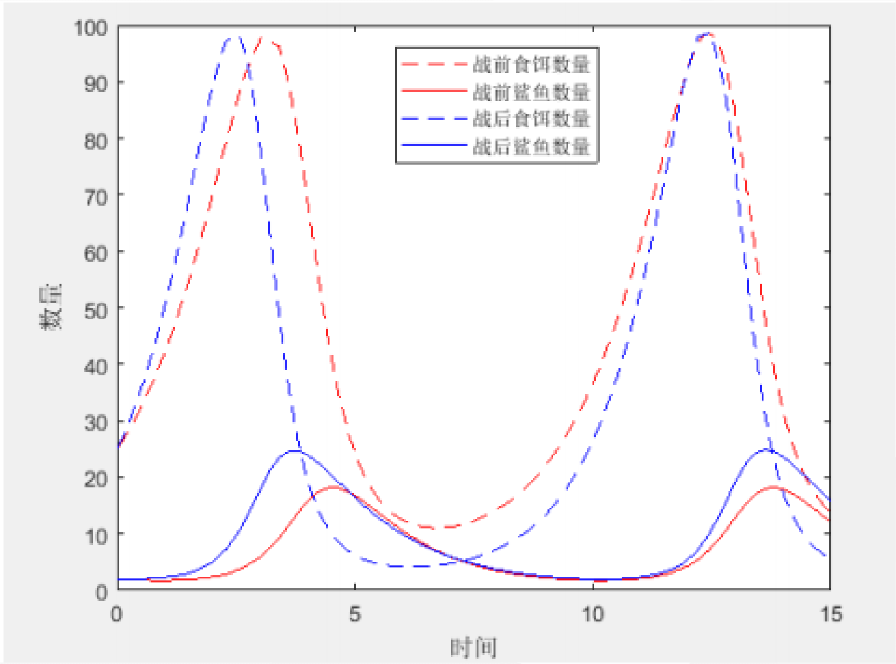


(1)从下方开始逆时针的去看上面这个相轨线图：随着食饵增多，则鲨鱼易于取食，所以鲨鱼数量增加；然而，由于鲨鱼数量的增多而需要食用更多的食饵，此时食饵的数量正在下降，直到食饵数量特别少时，鲨鱼进入了饥饿的状态而使得鲨鱼的数量急剧下降，这时部分食饵得以存活，食饵数量重新开始回升；食饵与鲨鱼的数量交替增减形成了生物圈中的动态平衡。

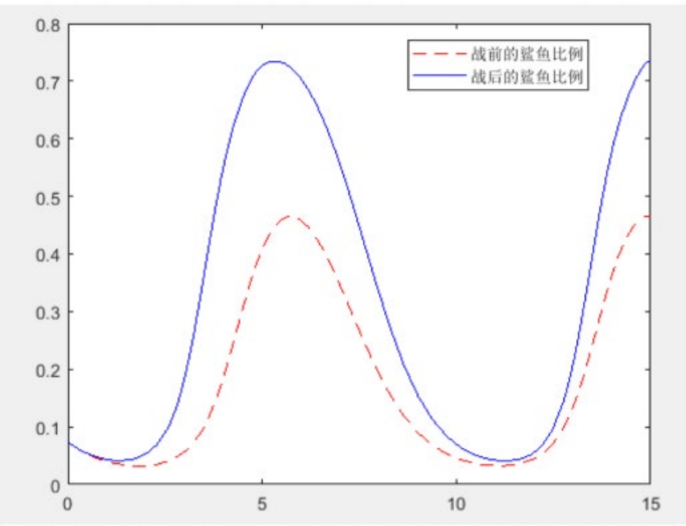
(2) 另外，蓝色的圈代表战后的数量变化，可以看出，蓝色的圈几乎把红色的圈包在里面，且蓝色圈的上方和红色圈的上方差距较大，这说明战后鲨鱼的增长速度要明显快于红圈，这导致了鲨鱼数量的大幅增长，和生物学家D'Ancona观察到的现象相一致。

微分方程 | 地中海鲨鱼问题

(3) 另外，我们从下面的这张图中也能直观的看出这种规律。



（4）为了更加明显知道鲨鱼的数量在战后较战前相比有多大的差异，我们分别计算出了鲨鱼在战后和战前的比例(鲨鱼比例 ＝ 鲨鱼数量/(鲨鱼数+食饵数))。从下图可以看出战后鲨鱼的比例要高于战前，这也印证了意大利生物学家D'Ancona的观察。



如果一个自然环境中存在两个或两个以上的种群，它们之间的关系大致可分为以下几种：相互竞争，相互依存，弱肉强食(食饵与捕食者),也可能毫无关系。弱肉强食我们已经研究过，最后一种情形我们不用研究。所以还剩下两种关系，这部分我们先来研究两个种群的相互竞争模型，在下个部分我们再来研究两个种群的相互依存模型。

考虑单个种群在自然资源有限的环境下生存时，我们常用阻滞增长模型来描述它的数量的演变过程，即：$\frac{dx(t)}{dt} = rx\left(1 - \frac{x}{N}\right)$，

其中r是增长率，N是该环境下能容纳的最大数量。

假设甲乙两个种群是相互竞争的关系，我们定义以下符号：$x_1(t),x_2(t)$分别是甲乙两个种群的数量；

$r1,r_2$分别是甲乙两个种群的固有增长率；N_1,N_2分别是甲乙两个种群的最大容量；$x_1(0),x_2(0)$表示甲乙两个的种群的初始数量。

如果不考虑乙的影响，甲种群的数量变化服从阻滞增长模型：

$$\frac{dx_1(t)}{dt}=r_1x_1(1-\frac{x_1}{N_1})$$

注意：这里的因子$1-\frac{x_1}{N_1}$,反映了甲对有限资源的消耗导致其对自身增长的阻滞作用，

可理解为相对于N_1而言，数量为x_1时供养甲的食物量(假设仅仅考虑对于食物的竞争，食物的总量为1)。

如果考虑乙的竞争作用.那么乙消耗同一资源会对甲的增长造成影响,我们在$-1-\frac{x_1}{N_1}$，中再减去一项，该项和种群乙的数量x_2(相对于N_2而言)成正比，于是对于种群甲有：

$$\frac{dx_1(t)}{dt}=r_1x_1(1-\frac{x_1}{N_1}-\sigma_1\frac{x_2}{N_2})$$

这里多了一个系数σ_1,它表示的意义是：单位数量的乙种群(相对于N_2)消耗的供养甲的食物量为单位数量的甲(相对于N_1)消耗的供养甲的食物量的σ_1倍。

注意：如果$\sigma_1<1$则意味着在对供养甲的资源的竞争中乙弱于甲。例如：$\sigma_1=0.5$则意味着：乙对甲的食物的消耗强度只有甲的一半，也可以反过来理解为甲对资源的占有能力更强一点。

类似地，对于种群乙有：

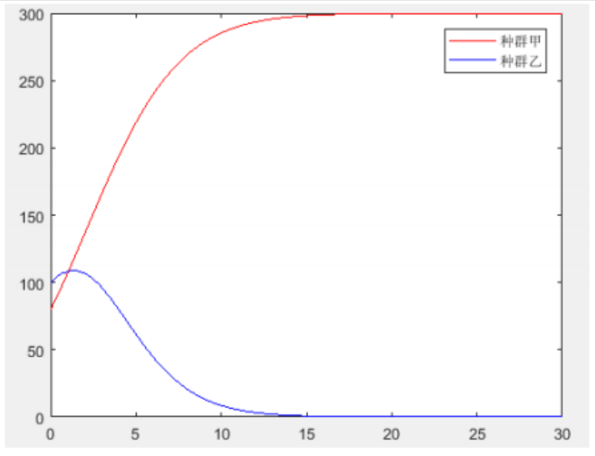
$$\frac{dx_2(t)}{dt} = r_2 x_2 (1 - \frac{x_2}{N_2} - \sigma_2 \frac{x_1}{N_1})$$

这里的系数σ_2表示的意义是：单位数量的甲种群(相对于N_1)消耗的供养乙的食物量为单位数量的乙(相对于N_2)消耗的供养乙的食物量的σ_2倍。

注意：σ_1和σ_2一般是相互独立的。只有在甲乙两个种群的食物选择以及对食物的偏好完全相同时，我们有$\sigma_1\sigma_1=1$。例如牛和羊都只吃草，那么假设$\sigma_1\sigma_1=1$没问题；猎豹和灰熊都吃小动物，构成了竞争关系，但是灰熊还会吃青草坚果等植物，此时$\sigma_1\sigma_1$就很有可能不等于1。

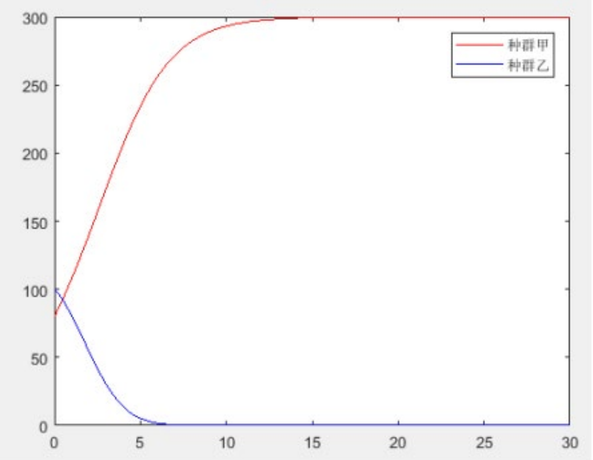
下面我们就使用Python,来生成一些数据进行仿真：

参数	r_1	r_2	N_1	N_2	σ_1	σ_2	$x_1(0)$	$x_2(0)$
模拟数值	0.5	0.5	300	500	0.5	2	80	100



$\sigma_1<1$意味着在对供养甲的资源的竞争中乙弱于甲，$\sigma_2>1$意味着在对供养乙的资源的竞争中甲强于乙，于是乙终将灭绝，种群甲将趋于最大容量。

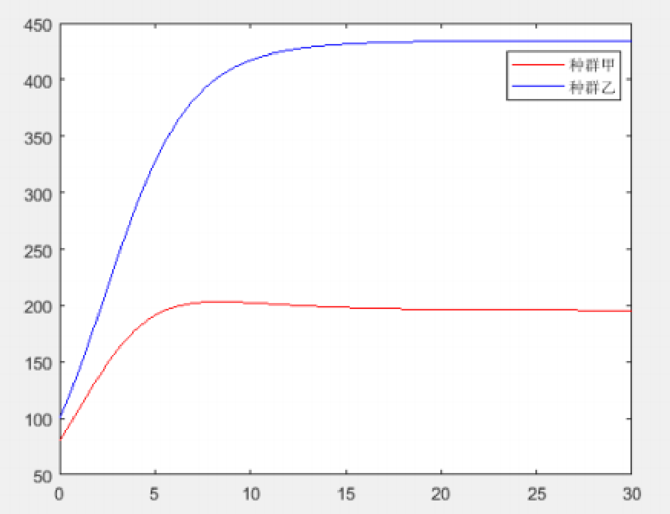
参数	r_1	r_2	N_1	N_2	σ_1	σ_2	$x_1(0)$	$x_2(0)$
模拟数值	0.5	0.5	300	500	0.5	4	80	100



将σ_2从2增加到4,此时在对供养乙的资源的竞争中甲的竞争力增加了，和上一张图相比可以发现，种群乙灭绝的时间提前了。

微分方程 | 种群相互竞争模型

参数	r_1	r_2	N_1	N_2	σ_1	σ_2	$x_1(0)$	$x_2(0)$
模拟数值	0.5	0.5	300	500	0.4	0.2	80	100



$\sigma_1<1$意味着在对供养甲的资源的竞争中乙弱于甲，$\sigma_2<1$意味着在对供养乙的资源的竞争中甲弱于乙，于是甲乙两个种群的数量最终会在竞争中趋于平衡。

历史上曾经爆发过多次大范围的传染病，例如天花、流感、SARA、猪流感等。2020年上半年爆发的新冠肺炎具有发病急、传播快、病情重等特点，给人们的健康带来极大的危害，对社会的繁荣安定带来严重影响。因此，搞清楚传染病的传播过程、分析其传播特征、传播规律，以便避免传染病的暴发、控制传染病的蔓延、最终消灭传染病是一件非常重要的事。我们可以建立数学模型描述传染病的传播过程、分析其传播规律、为预测和控制传染病提供信息和支持。要深入区分不同传染病的不同传播特点，需要很多病理学知识。这里我们仅从传染病的一般传播机理出发建立几种最常用的传染病模型。

(1)易感者(S,Susceptible):潜在的可感染人群;

(2)潜伏者(E,Exposed):已被传染但没表现出来的人群;

(3)感染者(I,Infected):确诊感染的人;

(4)康复者(R,Recovered):已痊愈的感染者，体内含抗体。

注意：以上分类的命名不唯一，例如有的文献将R解释为移出状态(removed),表示脱离系统不再受到传染病影响的人(痊愈、死亡或被有效隔离的人)。

微分方程 | 传染病模型

- 首先考虑最基本的模型——SI 模型。这里 S 是指易感染者 (Susceptible) 也就是健康人，I 指已感染者 (Infective) 也就是患者。在该模型中我们把人群分为易感染者和已感染者两大类 (即健康人和患者)。

- 假设我们不考虑人口的迁移和生死，设总人数为 N, 那么在任意时刻 N=S+I. 接下来我们需要考虑传染病的传播机理，这里有两种不同的角度： (1) 假设单位时间内，易感染者与已感染者接触且被传染的强度为β, 且单位时间内，由易 感染者 S 转换为已感染者 I 的人数为：

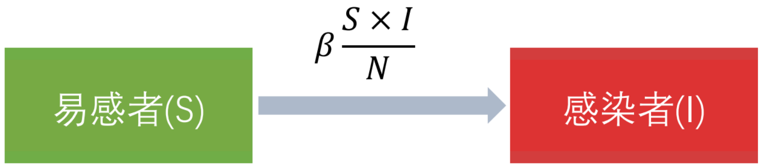
- $\beta\frac{S}{N}\times\frac{I}{N}\times N=\beta\frac{SI}{N}$ 。那么有$S(t+\Delta t)-S(t)=-\beta\frac{S(t)\times I(t)}{N}\Delta t$, $I(t+\Delta t)-I(t)=\beta\frac{S(t)\times I(t)}{N}\Delta t$ 。这里我们可以借用高数的知识：

- $\frac{S(t+\Delta t)-S(t)}{\Delta t}=-\beta\frac{S(t)\times I(t)}{N}$,则$\frac{\mathrm{dS}}{\mathrm{dt}}=\lim_{\Delta t\to 0}\frac{S(t+\Delta t)-S(t)}{\Delta t}=-\beta\frac{\mathrm{S}\times\mathrm{I}}{\mathrm{N}}$,

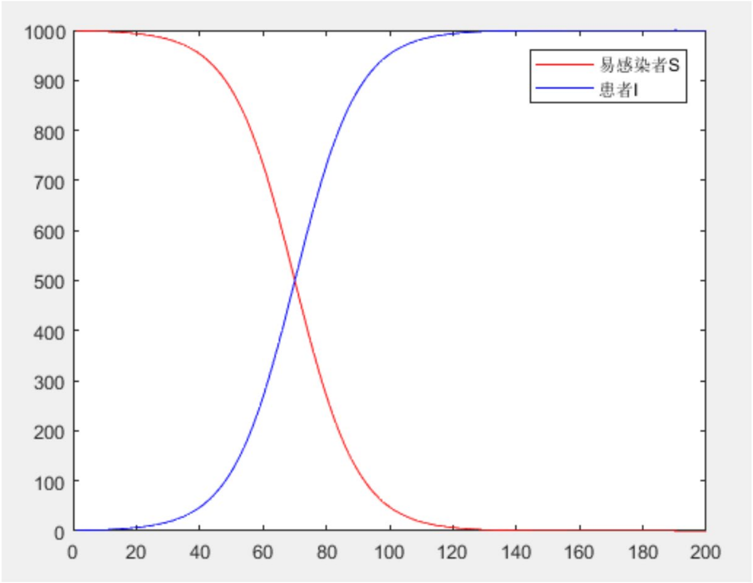
- 接下来我们即可得到对应的微分方程为：

$$\begin{cases} \dfrac{\mathrm{dS}}{\mathrm{dt}} = -\beta\dfrac{\mathrm{S}\times\mathrm{I}}{\mathrm{N}} \\ \dfrac{\mathrm{dI}}{\mathrm{dt}} = \beta\dfrac{\mathrm{S}\times\mathrm{I}}{\mathrm{N}} \end{cases}$$

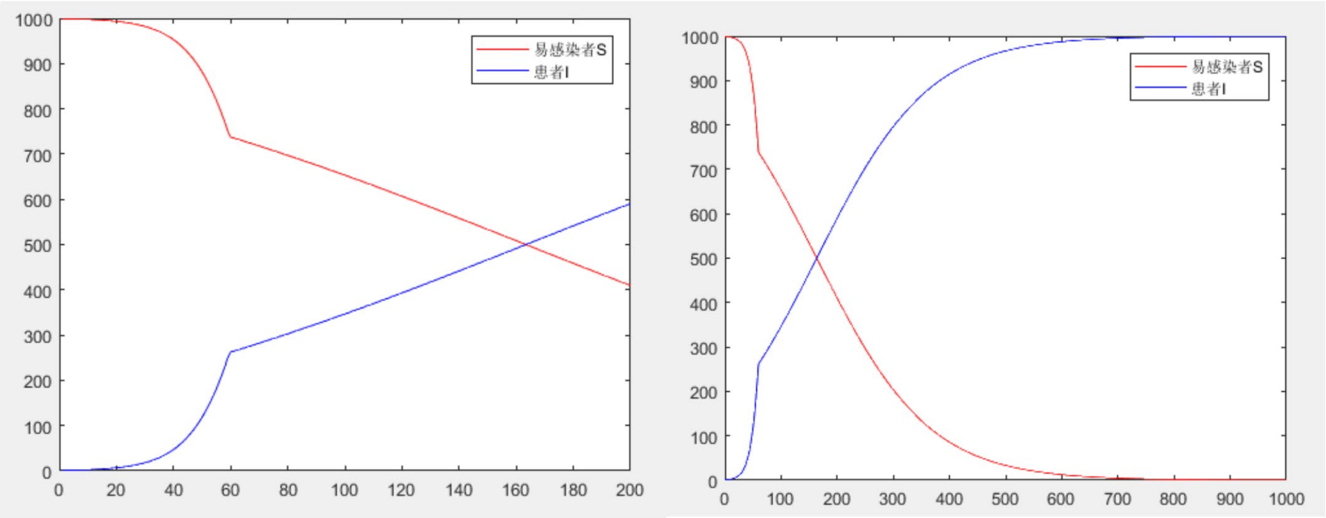
- 状态转移图形如下：



假设总人数为1000人，最开始有1个人感染，易感染者与已感染者接触且被传染的强度β取0.1,我们可以用Python做出模拟200期的图像：

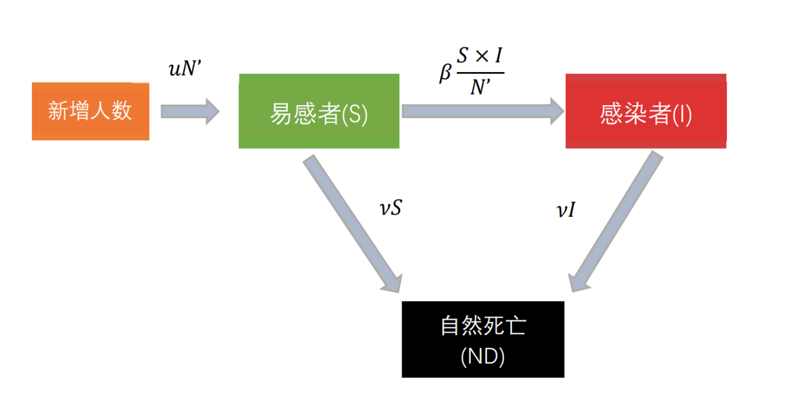


对于SI模型的拓展：
(1):考虑某种使得参数β降低的因素(例如禁止大规模聚会、采取隔离措施等)
例如：第60期后禁止大规模聚会，使得传染强度beta缩小为原来的10倍
我们下面画了两张图，分别是模拟200期的图像和模拟1000期的图像，可以看出，第60期后，疾病的传染速度有了明显的下降。



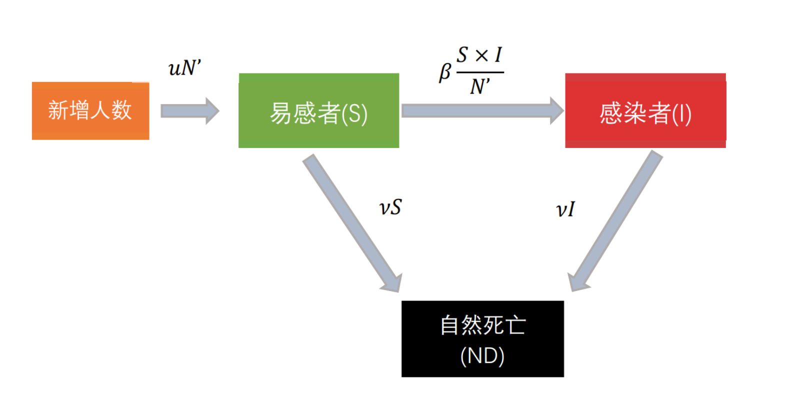
(2): 增加人口自然出生率和死亡率，但不考虑疾病的死亡率 假定初始总人口数为$N(=S+I)$, 疾病流行期间，人口出生率和自然死亡率分别为μ和ν, 不考虑因病死亡，新增人都是易感染者，初始时单位时间内感染人数为$\beta\frac{SI}{N}$。

注意，由于总人数 N 不再固定，因此我们在后期不断对总人数 N 进行更新，新的总人口N'的计算公式仍然为 S+I。其状态转移图形为：

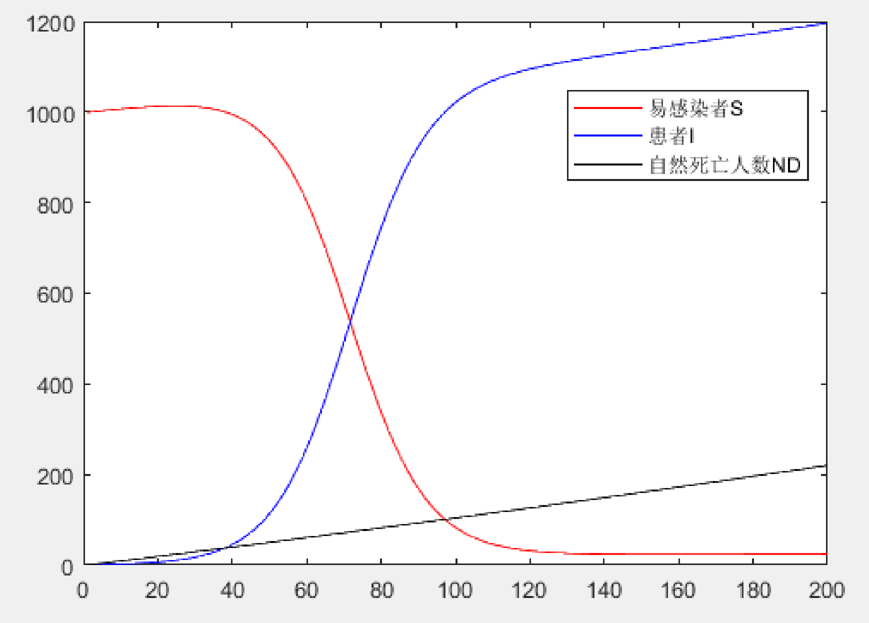


对应的微分方程组为:

$$\begin{cases} \dfrac{dS}{dt} = -\beta\dfrac{S\times I}{N'} + uN' - vS \\ \dfrac{dI}{dt} = \beta\dfrac{S\times I}{N'} - vI \\ \dfrac{dND}{dt} = vS + vI \end{cases}，且N'=S+I$$



取β=0.1 u=0.002 v=0.001,利用Python可以得到:



微分方程 | 传染病模型

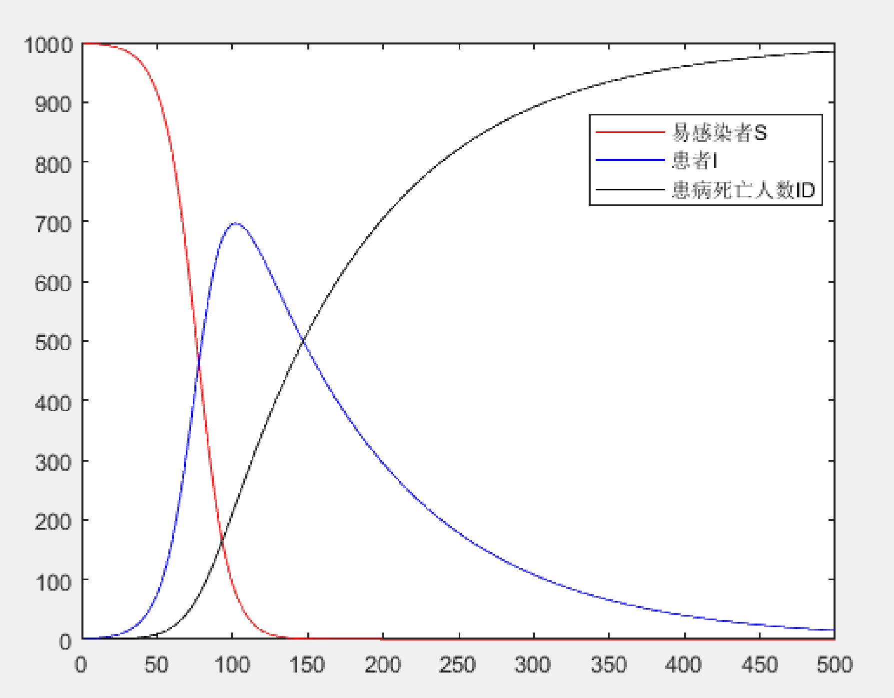
(3):不考虑人口自然出生率和死亡率，只考虑疾病的死亡率假定因病的死亡率为d,不考虑人口自然出生率和死亡率。其状态转移图形为：

对应的微分方程组为：

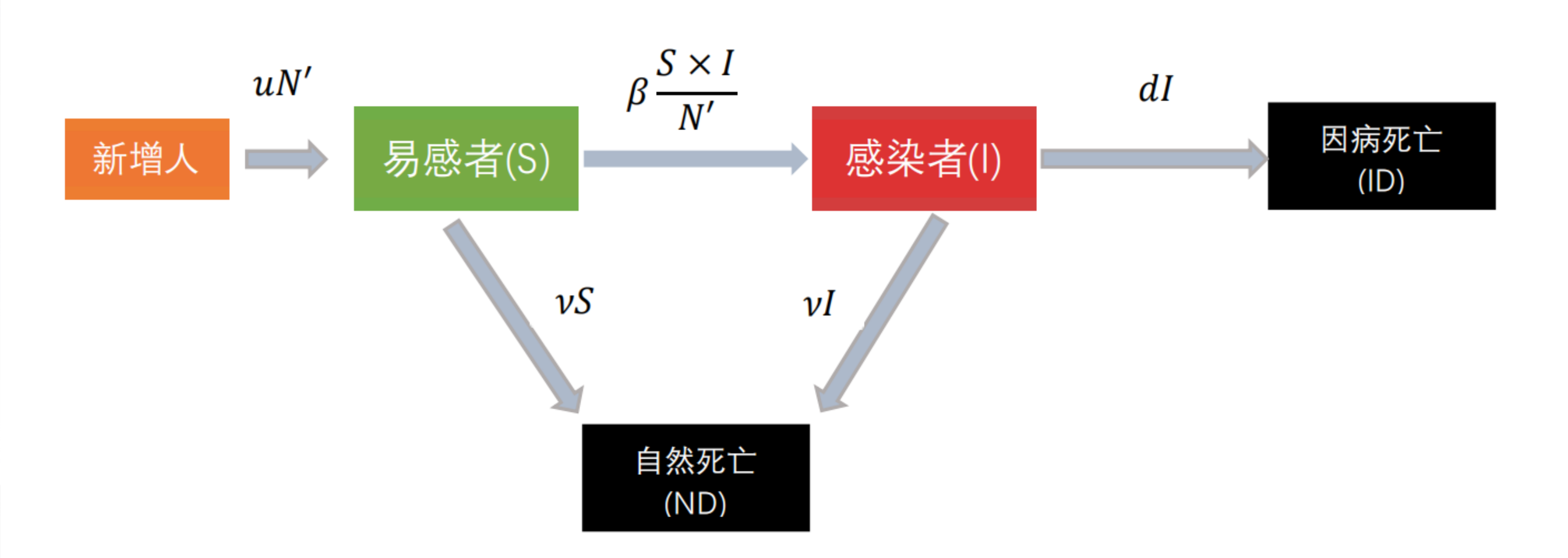
$$\begin{cases}\frac{dS}{dt}=-\beta\frac{S\times I}{N'}\\ \frac{dI}{dt}=\beta\frac{S\times I}{N'}-dI\ ,\ 且N'=S+I\\ \frac{dID}{dt}=dI\end{cases}$$

微分方程 | 传染病模型

取β=0.1 d=0.01,利用python可以得到：



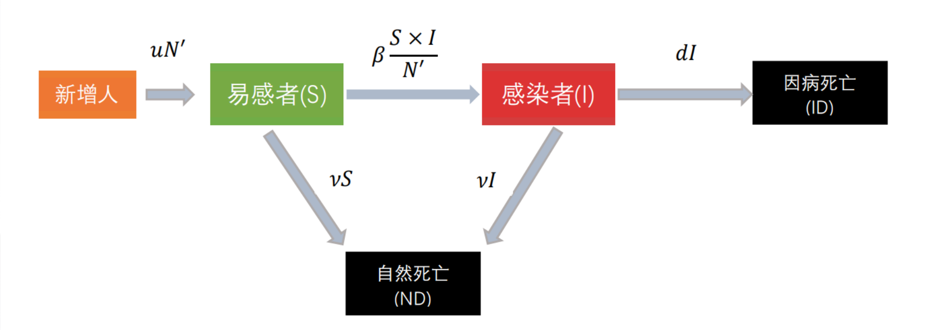
4）:同时考虑人口自然出生率和死亡率和疾病的死亡率疾病流行期间，人口出生率和自然死亡率分别为μ和v, 因病的死亡率为d, 其状态转移图形为：



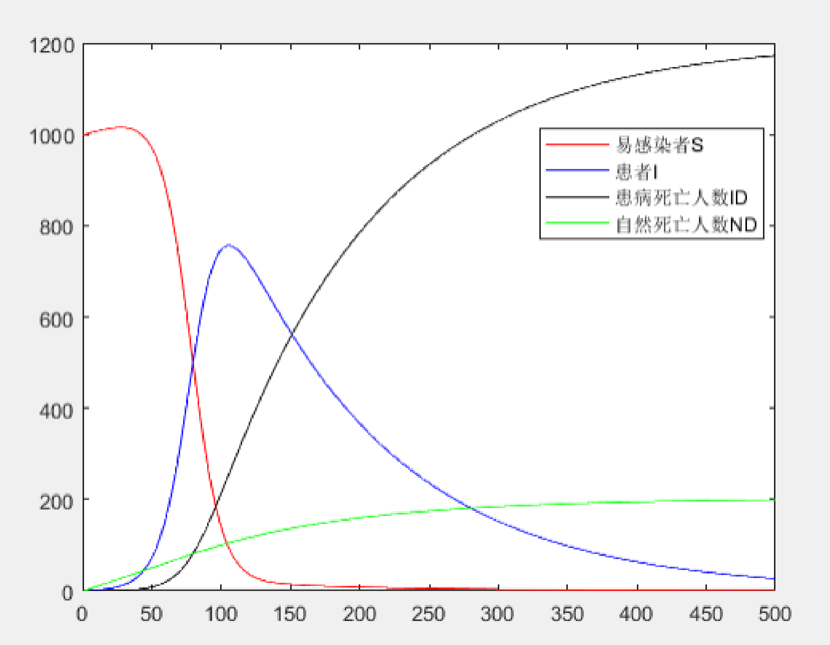
对应的微分方程组为:

$$\begin{cases} \dfrac{dS}{dt} = -\beta \dfrac{S \times I}{N'} + uN' - vS \\ \dfrac{dI}{dt} = \beta \dfrac{S \times I}{N'} - dI - vI \\ \dfrac{dID}{dt} = dI \\ \dfrac{dND}{dt} = vS + vI \end{cases}$$

，且$N'=S+I$

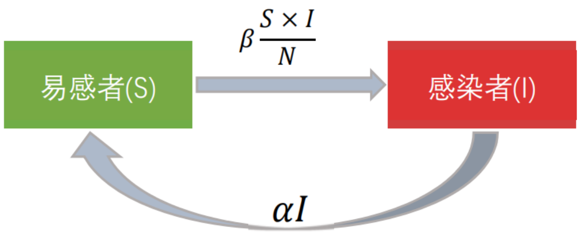


取β=0.1 d=0.01 u=0.002 v=0.001,利用Python可以得到:



微分方程 | 传染病模型

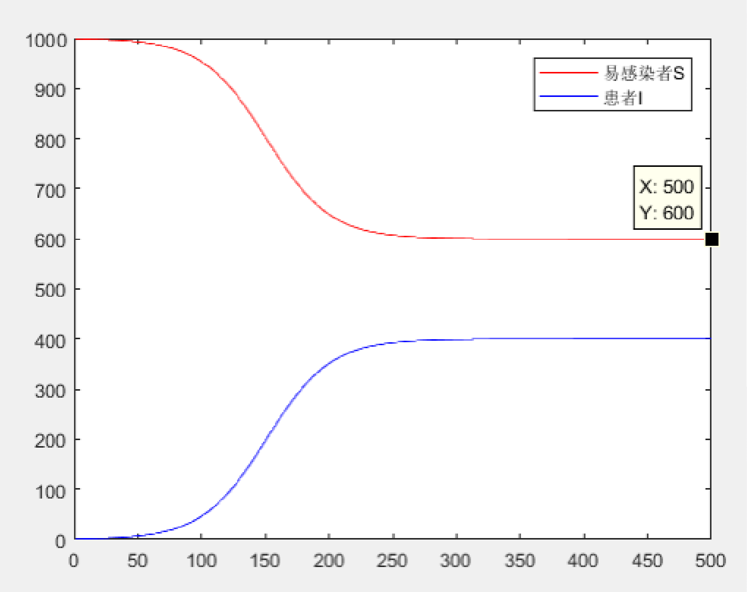
某些疾病容易被治愈，但是却容易发生变异，例如流感病毒。假设从某种疾病恢复后仍不能产生抗体，即未来仍可能患病，即我们可能会经历：感染，恢复为易感者，再感染，不断循环下去的过程。我们可以建立SIS模型来描述这一过程，假定总人口数为N,且不考虑因病死亡和自然出生死亡，单位时间内感染人数为βSI/N,由感染状态I恢复为易感者状态S的恢复率为a.其状态转移图形为：



对应的微分方程组为：
$$\begin{cases} \dfrac{dS}{dt} = \alpha I - \beta\dfrac{S\times I}{N} \\ \dfrac{dI}{dt} = \beta\dfrac{S\times I}{N} - \alpha I \end{cases}$$

微分方程 | 传染病模型

假设总人数为1000人，最开始有1个人感染，易感染者与已感染者接触且被传染的强度β取0.1,恢复率a取0.06,我们可以用Python做出模拟500期的图像：



欢迎关注数模加油站

THANKS