HD·EDUCATION

HD EDUCATION

MATH3076

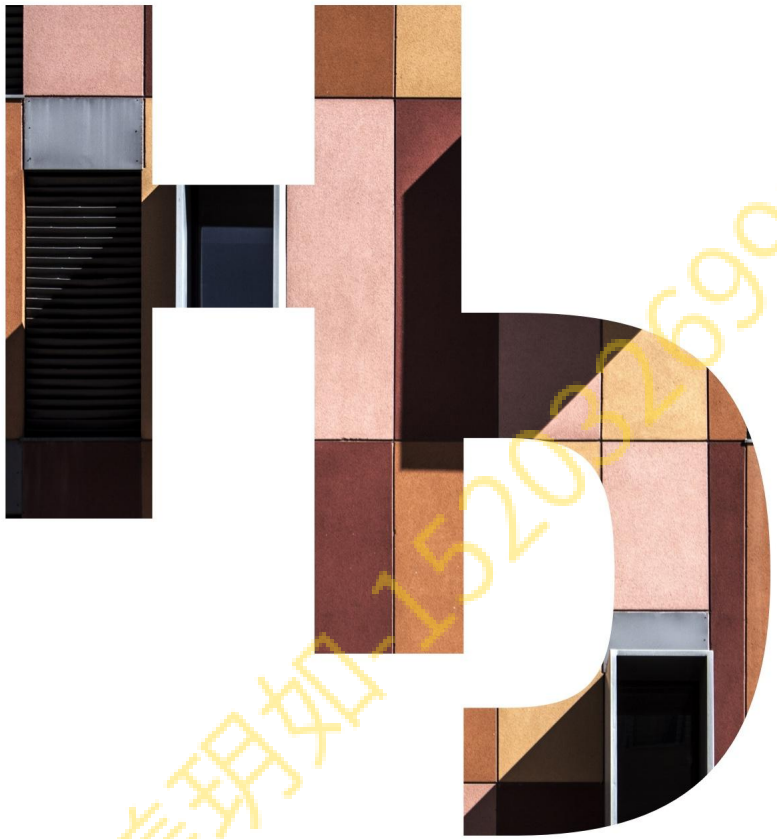
课程拓展课3

TUTOR:Doris

HD·EDUCATION

全球累计服务用户超十万

· 让海外学习更轻松 ·



关于
HD EDUCATION

HD · EDUCATION（简称HD·EDU）成立于2018年1月，拥有学业辅导和职业规划两大核心业务。从创办伊始就秉承着 “让年轻人成为知识的生产者、传播者、受惠者” 的使命，坚持从留学生的角度出发，为他们量身制定属于他们的课程。“成为最受年轻人喜爱的教育品牌“一直是我们的不懈追求。

截止2020年，我们的Tutor人数已达1300人，业务范围涵盖了澳大利亚、新西兰、美国、英国4个国家的40多所高等校，为15万留学生提供了优质的学习辅导服务，成为澳大利亚华人留学生覆盖人数最多的在线教育学习平台。

HD·EDU的成长有你陪伴

课后，如果您有任何建议和意见，我们都非常欢迎您联系小助手分享您的想法，给予我们改进和提高的机会！

感谢您参与HD Education的辅导课程！

TUTOR
Self-Introduction
自我介绍

#

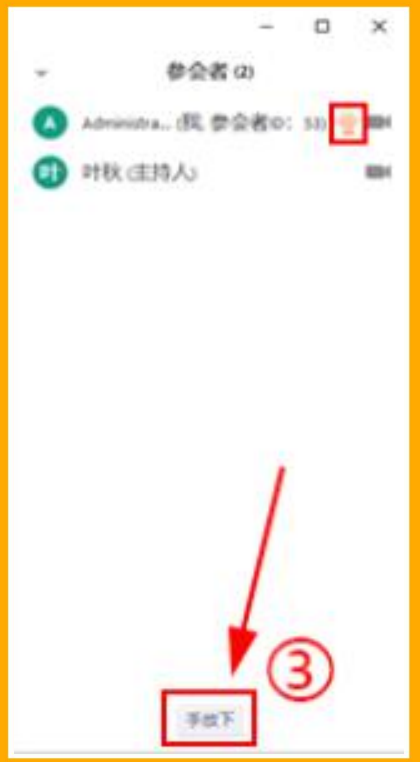
TUTOR:Doris

1. 阶段：目前毕业于港中文的master，phd申请ing；
2. 背景：本硕专业均为数学系，擅长计算机和几何相关课程；
3. 优势：具有较好的数学优势与几何思维；
4. 教学风格：依照较轻松的方式教学，喜欢并鼓励学生创造性地解决问题；
5. 兴趣爱好：学习其他方向的知识、画画；
6. 工作经历：在港中文和港大做了一年多的研究助理，目前在香港理工大学参加计算机视觉应用项目研究。

HD·EDUCATION

同学们有问题怎么办?

方法一:举手



1.点【参会者】
2.点【举手】即可与老师实时互动
3.问题被解答了还可以【手放下】

HD·EDUCATION

同学们有问题怎么办?

方法二:
文字提问

红圈处输入问题提问

HD·EDUCATION

同学们有问题怎么办?

直播平台

互动方法

直播平台：举手+聊天室提问

点【参会者】再点【举手】，即可与老师实时互动！ 在此输入你想问的问题

问题被解答了还可以【手放下】

HD·EDUCATION

CONTENT

课程目录

HD·EDUCATION

作业简介

考试介绍

作业标题：Assignment 2: Epidemic modelling

DDL：11:59PM, 6 June 2021 (SUNDAY)

题目数量：3 tasks in total，task 1 contains 3 subtasks

提交形式：（Online- Turnitin）

作业权重：15%

HD·EDUCATION

知识点精讲

---background knowledge

考点1．知识点精讲

SIR模型是一个结合了全球真实流行病数据、气象因素和隔离措施的改良流行病SIR模型。假定在暴发期间不同地区的总人口保持不变；COVID-19只是通过人与人之间的传染扩散；个体之间没有免疫力差异。每个国家的总人口被分成三种类型：易感人群S，感染人群I，治愈和死亡人群R。

重要程度：⭐⭐⭐
难易程度：⭐⭐⭐⭐⭐

Background

It so happens that we are (still) living through one of the more extraordinary times in the past century. I thought it would be especially relevant for everyone to know something about the phenomena that got us here in the first place.

It turns out there is an upside to everything. In normal years, this assignment at least partially considers numerically solving ordinary differential equations (ODEs). So let's use this opportunity to discuss solving ODEs with one of the most important systems of equations in the world right now (maybe it's *the* most important set of equations in the world right now). This was all a bit more true a year ago. But it's still pretty true now.

Susceptible-Infected-Recovered (SIR) models:

We start with the simplest possible model of an epidemic; the famous "SIR" model. This kind of modelling started in the 1930s, when A.G. McKendrick (a Scottish military doctor) and W.O. Kermack (a biochemist) produced their mathematical model. They originally considered all kinds of things like birth/death, different age groups, and migration. We will simplify things as much as possible to get an idea of how things work.

The assumptions are the following:

假设样本量足够大

- We suppose there is a closed population with a large number $N \to \infty$ of individuals.
- Within this population, we suppose the presence of an "infection". This could be something bad, like a disease. Or it could be something good, like deciding to get a vaccine.
- We split the population into three groups.

  Susceptible: people who can catch the infection.

  Infected: people who currently have the infection.

  "Recovered"/"Removed": people who are no longer contagious.

人群分类

It's possible to consider more subtle groups of people within the population. Or to consider several interacting populations. But for now, this is good enough. It reflects a common-sense notion of an epidemic.

痊愈和死亡人群为同一分类

It's also essential to point out that "recovered" may not be a good thing. People may be "recovered" because they are dead. It's better to think of "removed" individuals. A more nuanced model could consider people who are alive and recovered versus people who die from the infection. The former group may (for example) be able to help others without getting sick again. Another subtlety is that it might take a different amount of time to die than to recover completely. But again, for now, this is only to get you thinking. We'll have plenty to explore with the simplest model. Also, in the case of COVID-19, the **case fatality rate** is large enough to cause the whole world to shut down for multiple months. But it's still small enough that it doesn't affect the overall population level. Of course, this is not always reasonable. The Black Plague killed roughly 50% of Europe between 1346-1353.

病死率

考点1．知识点精讲

总共占比为1

With each group, we consider the ***relative fraction*** of the population in each group:

$$s(t) + i(t) + r(t) = 1$$

Considering the population as a fraction (or a percentage) is useful because it eliminates the explicit dependence on the total population size, N. Because of this relation, we don't need to model the recovered population explicitly. We know that

痊愈比率

$$r(t) = 1 - s(t) - i(t).$$

From now on, I'll focus on the susceptible fraction, $s(t)$, and infected fraction, $i(t)$.

Another parameter that might be relevant is

康复需要的时间

$$\tau \approx \text{time needed to ``recover'' from the infection.}$$

It turns out that this is not quite as important as you might originally think. But it's still useful to discuss.

For COVID-19 it seems that,

$$\tau \approx \text{2-4 weeks},$$

This is typical of many common respiratory viral infection (like the flu and the common cold). This number sets the overall timescale for how long the overall epidemic lasts. But we can solve everything without knowing τ in advance.

The fraction of susceptible and infected people follow a pair of coupled nonlinear ODEs:

$$\tau \frac{ds(t)}{dt} = -R_0\, s(t)\, i(t), \qquad \tau \frac{di(t)}{dt} = R_0\, s(t)\, i(t) - i(t).$$

易感和感染人群服从非线性微分方程

Notice a few features of this system:

- The term $R_0\, s(t)\, i(t)$ represents the action of the infection. It transfers the population from susceptible to infected.
- The transfer is symmetric. A decrease of one susceptible person leads directly to exactly one infected person.
- The transfer is multiplicative in both the susceptible and infected fraction. You need both to make things happen.

We can add the two populations together, and the transfer cancels

两种比例随时间变化

$$\frac{d(s(t) + i(t))}{dt} = -\frac{i(t)}{\tau}.$$

线性下降

This is the recovery. It says that the total susceptible and infected fractions linearly decrease into the recovered fraction. This is a very important part of the dynamic. If there were no recovery, then after some time, close to the whole population would be infected. The whole system would simplify into a single equation called ***logistic growth***:

综合公式：（逻辑增长）

$$\frac{di(t)}{dt} = \frac{R_0}{\tau}(1 - i(t))\, i(t), \quad \text{as} \quad \tau \to \infty.$$

This might be a good model for some "epidemics". Literacy, for example, might behave this way. You don't #unlearn reading. With epidemics that last for very long periods of time, you have to take many other things into account, like the natural birth rate. While people don't forget how to read, a society can fail to "infect" the next generation. But this is all very complicated to analyse.

We want to think about an epidemic that can start and end in a finite amount of time. We want an epidemic that is fast enough that normal population dynamics don't change very much over the course of the epidemic. COVID-19 is almost perfect from the modelling standpoint.

考点1．知识点精讲

重要程度：★★★
难易程度：★★★★★

The basic reproduction number, R_0

We now see the appearance of the famous and much talked about "R-naught" parameter:

R_0 = Average number of people an infected person infects (early in the epidemic).

早期的患者传播人数（假设患者周围都未感染

The ***basic reproduction number*** is both important and poorly understood. It's not just a parameter. It's a complicated function of all kinds of factors involving biology, human behaviour, and probability.

When you hear about R_0 in the news, it's actually about the value *at the beginning* of the epidemic. This is a well-defined quantity.

- COVID-19: $R_0 \approx 3$-6.
- Influenza: $R_0 \approx 0.9$-2.
- Measles: $R_0 \approx 12$-18.
- HIV: $R_0 \approx 2$-5.
- Ebola: $R_0 \approx 1.5$-2.

不同传染病R0不同

Clearly, R_0 is not the whole story by far when it comes to how serious an epidemic is. But it is essential for understanding how much time you have to respond to an outbreak.

儿童期疾病：传染度很高

There is one other thing worth mentioning about R_0. Infections that often called "childhood diseases" are anything with a very high $R_0 \approx 10$, or greater. Why? Absent a vaccine, it is just so likely that every susceptible person in a population will get it. In a population where Measles has existed for some time, children are the only susceptible people. This also further explains why diseases like Measles had such a disastrous effect on indigenous populations after the arrival of European colonists. High-R_0 infections happen so fast that you also need to take normal population dynamics into account, just as for infections that take very long periods to recover from, like HIV.

In the case of COVID-19, the world started out with only a tiny number of people infected (maybe only one). Everyone else was susceptible. Therefore,

初期假设全部未感染，除了i0个带头人，剩下全是易感s

$$i(t=0) = i_0, \quad s(t=0) = 1 - i_0$$

We can also assume i_0 is *tiny*. Within a place like Australia, probably $i_0 \approx 10^{-6}$. The precise value doesn't matter much.

The reason that R_0 is important is that it's vital information *early* in the epidemic when,

R0早期对i(t)变化影响较大

$$\log \frac{i(t)}{i_0} \approx (R_0 - 1)\frac{t}{\tau} \quad \text{(early on)}.$$

若传播人数R0大于1，感染率i将以指数级增长 i(t)=10^(...)

This is the exponential trend that we all watched closely in the news around the world in early 2020. The big deal is that nothing takes off unless $R_0 > 1$. The average person must pass along the infection to at least one other person. This seems obvious, but it wasn't understood until mathematical modelling pointed it out.

考点1. 知识点精讲

重要程度：⭐⭐⭐
难易程度：⭐⭐⭐⭐⭐

A more careful aproach looks at the follwoing relation that follows from the original equations:

由前面方程得到i的变化率：

$$\frac{di}{dt} = -\left(1 - \frac{1}{R_0\, s}\right)\frac{ds}{dt}.$$

Because the susceptible fraction can't increase ($ds/dt \leq 0$), and $0 < s < 1$,

易感率下降，感染率增加，R0*s-->病人传播数量

$$\frac{di}{dt} \geq 0 \iff R_0\, s \geq 1.$$

This also shows the infection will eventually go away; because s is always decreasing.

What is R_0, really?

A more detailed look shows that:

P：当患者遇到易感人群时，传播的概率。

$$R_0 = P_{\text{transmission}} \times N_{\text{encounter}}$$

N：在时间τ中，每两人相遇次数

where

$P_{\text{transmission}}$ = Probability a sick person will infect a susceptible person if they meet.

$N_{\text{encounter}}$ = Expected number of contancts between any two people within time $= \tau$

You can see that these parameters are clearly a function of biology and our behaviour. This is why social distancing works. And so does hand washing. Vaccines work by lowering the total number of susceptible people.

For a serious epidemic, it makes sense to consider

$$R_0 = R_0(s, i).$$

严重的流感，R0是由s，i决定的函数

We know very well that society responds if things are bad enough.

In general, this is a function of *both* the susceptible and infected fractions. For simplicity, we will temporarily pretend that it's some general function of s alone. This makes the math easier. It seems from watching the news that society mostly responds to the number of infected people. Clearly, without infected people, no one would change their behaviour.

But it's not quite that simple.

早期只有i和s，晚期i=0，疾病被基本消灭

- First, early in the epidemic, $s \approx 1 - i$, so the two are linked closely.
- At the end of the epidemic, $i \approx 0$, and s is the only variable for things to depend on.

So it is clear that R_0 does depend on s at the very least. Also,

- Very roughly $i \propto (1 - s)s$ throughout the epidemic. At least as far as approximating $R_0(s)$.
- Because s is monotonic, it is in one-to-one correspondence with time. Any function of time can be remapped to a function of s.

- R0依赖于s，i和(1-s)s成正比。
- 因为s是单调的，所以与时间一一对应。 任何时间函数都可以重新映射为 s 的函数。

考点1．知识点精讲

重要程度：⭐⭐⭐
难易程度：⭐⭐⭐⭐⭐

General properties.

一般对策以控制R0为手段，由于方法多样，所以R0无法以常数做假设，通常设为方程。

It is clear from our current situation that societies respond in complex ways to epidemics. This can happen because a disease spreads rapidly, is very deadly, last a very long time, or because immunity doesn't last forever (like the flu). Most of all, the responses occur by altering R_0 in complicated and nonlinear ways, e.g., vaccines, social distancing, education. Therefore, we want to make *computational* methods that can cope with a wide range of situations. It's not enough to assume R_0 is constant. We want to be able to experiment with all kinds of R_0 functions. Done well, this can help influence policy farther down the line.

Before just solving the ODEs, we can analyse some of the general mathematical properties of the system.

We can integrate the model over the susceptible fraction, giving a very general relation:

对s进行积分，得到r（康复）函数，康复人数是过去某个时间最终感染的所有易感人群的总和，与时间无关。

$$r(s) = \int_s^1 \frac{ds'}{s' R_0(s')}.$$

You might call this the ***recovery function***. It might look like it came from the sky. But this equation says the number of recovered people is the sum of all the past susceptible people who ended up infected at some time. It doesn't matter how long it took.

Of course recall,

$$i(s) = 1 - s - r(s)$$

高峰期感染率，结束后易感率较重要

We are really interested in the number of infected people at the peak and the number of susceptible people at the end of the epidemic.

`import`

You are allowed to use the following imported libraries. In the case, of integrating and time stepping differential equations, it's better to use something supplied by `scipy` than `DIY`.

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 %matplotlib inline
4
5 # You should use these:
6 from scipy.integrate import simps
7 from scipy.integrate import odeint
```

simps辛普森法则，用于积分
odeint()函数是scipy库中一个数值求解微分方程的函数

考点1．知识点精讲

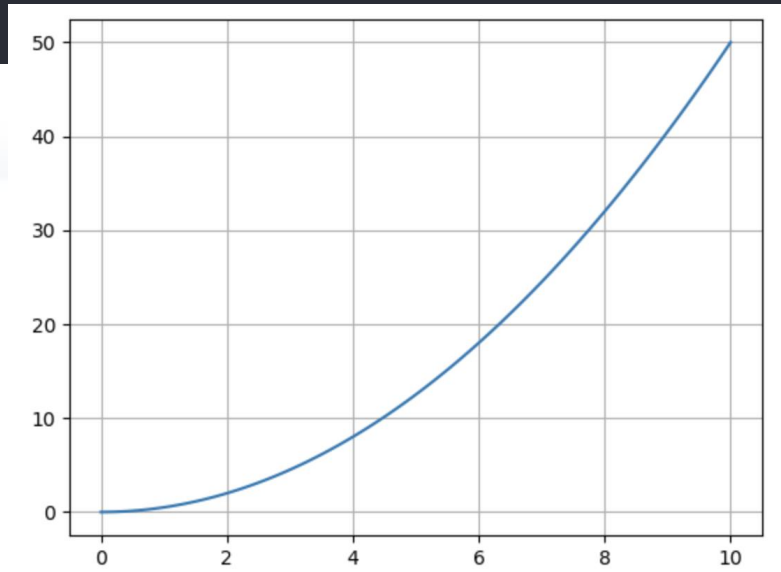
重要程度：★★★
难易程度：★★★★★

simps辛普森法则，例如对numpy数组np.arange(-9,10)进行积分：
y就是我们要积分的数组，x则是每一个y对应的坐标。

```
from scipy.integrate import simps

y = np.arange(-9,10)
print(simps(y))
x = np.arange(y.size)
print(simps(y,x))

x = x ** 2
print(simps(y,x))
```

odeint()函数是scipy库中一个数值求解微分方程的函数，需要至少**三个变量**，第一个是**微分方程函数**，第二个是微分方程**初值**，第三个是微分的**自变量**。
一个一阶微分方程例子：

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

def diff(y, x):
    return np.array(x)
    # 上面定义的函数在odeint里面体现的就是dy/dx = x
x = np.linspace(0, 10, 100)  # 给出x范围
y = odeint(diff, 0, x)  # 设初值为0 此时y为一个数组，元素为不同x对应的y值
# 也可以直接y = odeint(lambda y, x: x, 0, x)
plt.plot(x, y[:, 0])  # y数组（矩阵）的第一列，（因为维度相同，plt.plot(x, y)效果相同）
plt.grid()
plt.show()
```



HD·EDUCATION

作业内容解读

作业内容解读--task 0

TASK 0.0: `recovery` 求康复率

Make a Python function called `recovery` that computes $r(s)$. Your function should have the following input:

- The susceptible fraction, `s`. 需要输入s作为参数
- A Python *function* called `R0` that computes $R_0(s)$. 定义传播人数R0的方程
- A default integer argument, `n=101`, giving the number of internal points to use for the integration. 设定默认数量n，作为积分内部分割点数

You should consider the following guidelines:

- If you give your function a `float` for `s`, you should return a `float` type.
- If you give an `np.ndarray` input type for `s`, your function should return the results in an array; i.e., your function should be *vectorised*.
- You should use `scipy.integrate.simp` to carry out your internal integration. This will mean your accuracy will only depend on `n`.

n：分割数。将会影响积分结果准确度。

For example:

```
In [0]: def R0(s):
            return 4 + 0*s

        s = 0.2

        recovery(s,R0,n=101)
```

例如将R定义为常数

```
Out[0]: 0.40235949932702725

In [1]: def R0(s): 将R定义为关于感染数的方程
            4*(1 - 9*(1 - s)*s**4)

        s = np.array([0.2,0.3,0.4])

        recovery(s,R0,n=1001)

Out[1]: array([0.66912522, 0.56496397, 0.48604337])
```

可以使用函数嵌套

Remember, you can pass *functions* to other functions. This is useful for something like an integration scheme. You might want to pass in different functions to be integrated and just get the answer back. How everything happens behind the scenes is not something you want to worry about each time.

作业内容解读--task 0

For example:

```
1  ### SKIP
2  ### Demo
3
4  def square(x):  平方
5      return x*x
6
7  def increment(x):  增加1
8      return x+1
9
10 def wrapper(x,f):  函数嵌套
11     print(f'Calling: {f.__name__}({x})')
12     y = f(x)  可以直接输入变量、函数名取得结果
13     print(f' Result: {y}','\n')
14     return y
15
16 result = wrapper(3,square)
17 result = wrapper(4,increment)
```

```
Calling: square(3)
 Result: 9

Calling: increment(4)
 Result: 5
```

```
1  ### TEST FUNCTION: test_recovery
2  ## DO NOT ERASE ANYTHING INITIALLY IN THIS CELL ##
3
4  def recovery(s,R0,n=101):
5
6      ### INPUT YOUR CODE BELOW THIS LIKE ###
7
8      return
```

```
1  ### SKIP
2  ### TEST YOUR CODE HERE ###
3
4
```

作业内容解读--task 0

TASK 0.1: `susceptible_final` 流感结束后的易感率

For a given $R_0(s)$, make a function called `susceptible_final` that computes the value of the susceptible fraction after the epidemic is over. This is defined as s when the infection fraction is zero. We know from the recovery function this is when

$$i(s) = 1 - s - r(s) = 0$$ i=0时结束流感

Solve this using Newton's method. Make a function called `s_inf` according to the following guidelines: 用牛顿法求i结束时刻对应的s值

The function should take:

- a Python *function* `R0`.
- `s_guess` for the initial Newton's method guess. s_guess给出s的初值

The function should also take the following optional arguments:

- `iterations = 30` for the number of Newton steps to use internally.
- `n = 101` for the number of interanl grid points to use compuiting `r(s,R)`.

默认迭代次数为30，计算康复率（要用到辛普森积分）时默认分割101段

For example:

```
In [0]: def R0(s):
            retrurn 4+0*s

        susceptible_final(R0,0.01)

Out[0]: 0.019856516268401604
```

```
### TEST FUNCTION: test_susceptible_final
## DO NOT ERASE ANYTHING INITIALLY IN THIS CELL ##

def susceptible_final(R0,s_guess,iterations=30,n=101):

    ### INPUT YOUR CODE BELOW THIS LIKE ###


    return
```

作业内容解读--task 0

TASK 0.2: `infection_max`

Make a function called `infection_max` that computes the maximum infected fraction for a given $R_0(s)$. This happens at the value of s where

计算最大感染率（当导数为0）

$$i'(s) = -1 + \frac{1}{s\,R_0(s)} = 0$$

Therefore, you have to solve

整理方程

$$s\,R_0(s) - 1 = 0,$$

and then substitute that value back into $i(s)$.

对变量s用牛顿法求方程的根

- You should use Newton's method on the function $s\,R_0(s) - 1$
- This will require computing the derivative $R_0'(s)$.
- Not knowing in advance the form of $R_0(s)$, approximate the derivative using the second-order centred finite-difference formula

由于不知道R0的公式，其导数用二阶中心有限差分计算

$$R_0'(s) \approx \frac{R_0(s+ds) - R_0(s-ds)}{2ds} \quad \text{for some small value of } ds.$$

Your function should take the following input:

- A Python function `R0`.
- The initial guess for the critical susceptible fraction `s_guess`.

s_guess给出初值

You should also have the following default arguments:

- `ds=1e-3` for the finte-difference spacing. 误差
- `iterations=30` for the number of Newton iterations.

Your function should output *both* of the following:

- The value of `s` where the peak occurs.
- The value of `i` at the peak.

需要同时输出s的最高值和i的值

The order of the output should be `s,i`.

For example:

```
In [0]: def R0(s):
            return 4 + 0*s

        infection_max(R0,0.5)

Out[0]: (0.25, 0.40342640301079824)
```

易感率0.24，感染率0.4

- This is 40% of the population at the same time!

作业内容解读--task 0

```
### TEST FUNCTION: test_infection_max
## DO NOT ERASE ANYTHING INITIALLY IN THIS CELL ##

def infection_max(R0,s_guess,ds=1e-3,iterations=50):

    ### INPUT YOUR CODE BELOW THIS LIKE ###

    return
```

作业内容解读--task 1

TASK 1: `SIRModel` 写一个 SIRModel 的函数，求解一般 $R0(s,i)$ 的变化方程。

Write a function called `SIRModel` that solves the dynamical equations for a general $R_0(s,i)$.

$$\frac{ds}{dt} = -R_0(s,i)\,s\,i, \qquad \frac{di}{dt} = R_0(s,i)\,s\,i - i$$

You can assume that $\tau = 1$. This means that the timescale is in units of the recovery time. Meaning your time axis should have units of about a month if you think about COVID-19.

可以假设 τ=1 。时间刻度以恢复时间为单位。

You should follow the design specifications:

- Your function should take an input `np.array` `t` for which you want the solution.
- Your function should take a Python function `R0`, which is assumed to depend only on `s`, `i` in that order.

函数 R0，假定该函数仅依赖 s,i。

- You should include a default initial-condition `i0 = 1e-6`. Assume `s0=1-i0`.
- Your function should output a tuple of two arrays, the susceptible and infected fraction `s`, `i`, in that order.
- You should define the right-hand sides of the equations inside your function. I want this to be a "black box".

在函数内定义方程的右侧。

- You should use `scipy.integrate.odeint` to time step your system. Look up online how to use it.

用odeint来执行

```
For example:

In [0]: t = np.linspace(0,30,200)

        def R0(s,i,t):
            return 4 + 0*s

        s,i = SIRModel(t,R0)

        s[-1]

Out[0]: 0.01982739605220006
```

```
1  ### TEST FUNCTION: test_SIRModel
2  ## DO NOT ERASE ANYTHING INITIALLY IN THIS CELL ##
3
4  def SIRModel(t,R0,i0=1e-6):
5
6      ### INPUT YOUR CODE BELOW THIS LIKE ###
7
8      return
```

作业内容解读--task 2

TASK 2: A multiple-peak epidemic 多个峰值情况

When we look at the news from around the world, we see a few general trends happening.

- If an epidemic is left to naturally "run its course", then infection fraction will comprise a single large bump (e.g. with the seasonal flu). 让流行病自然发展，那么感染人数将有一个大爆发
- In the case of a severe epidemic, we now know that societies will respond and attempt to decrease R_0.
- The simplest response is "flattening the curve"; the peak is lower, the epidemic last longer, and perhaps fewer people overall get sick.
- Some societies, however, decide to reverse social-distancing policies during the epidemic.
- If "opening up" happens early enough, it means that the epidemic behaves as it would with little or no intervention. You still get a single bump.
- If "opening up" happens late enough, then you get a smaller single bump.
- If however, a country goes back to normal within a narrow window, then what looks like a single bump can turn into an epidemic with two peaks.

在R0函数中找到两个峰值点

I want you to find an $R_0(s,i)$ that produces a double bump. Inside the function you can have it depend on s and i any way you want with the following conditions:

- **Zeroth**: the function must be positive for any well-defined input:

条件：

$$0 \;<\; R_0(s,i), \qquad \text{for} \quad 0 \le s,i \le 1.$$

- **First**, the value if no one is susceptible should be the same if everyone is and there is no infection.

$$R_0(s=1,i=0) \;=\; R_0(s=0,i)$$

- **Second**: the function should not become larger than its initial value at any point during the epidemic.

$$R_0(s,i) \;\le\; R_0(s=1,i=0) \qquad \text{for} \quad 0 < s,i < 1.$$

```
### TEST FUNCTION: test_multi_bump_R0
## DO NOT ERASE ANYTHING INITIALLY IN THIS CELL ##

def multi_bump_R0(s,i):

    ### INPUT YOUR CODE BELOW THIS LIKE ###

    return
```

作业难点总结

作业重难点总结 & 建议

教案范例

1. 包含了一些初级的概率论知识。
2. 需要用到ODE常微分方程的求解内容。
3. 两个模块的使用方法。
4. 如何搭建整体的模型框架是一个难点，建议先理清公式和思路。

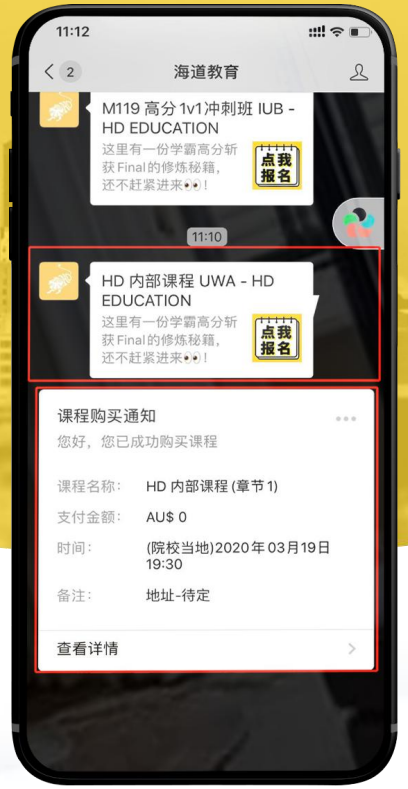
课程结束后，如果您对课程或者服务的任何建议和意见

请给予我们提高和改进的机会，感谢您对 HD · EDUCATION 课程和服务的信任！

· 填写问卷操作流程 ·

第一步

关注【海道教育】服务号



第二步

点击【购买通知】或【上课提醒】

第三步

【填写问卷】