Лабораторная работа №6. Модель эпидемии

c/б 1032186063 | НФИбд-01-18

Доборщук Владимир Владимирович

20 марта 2021

Содержание

# Цели и задачи

**Цель:** изучить простейшую модель эпидемии, а также реализовать её программно.

**Задачи:**

* изучить теорию о модели эпидемии
* построить модель для случаев:

# Теоретическая справка

Рассмотрим простейшую модель эпидемии. Предположим, что некая популяция, состоящая из особей, (считаем, что популяция изолирована) подразделяется на три группы. Первая группа - это восприимчивые к болезни, но пока здоровые особи, обозначим их через . Вторая группа – это число инфицированных особей, которые также при этом являются распространителями инфекции, обозначим их . А третья группа, обозначающаяся через – это здоровые особи с иммунитетом к болезни.

До того, как число заболевших не превышает критического значения , считаем, что все больные изолированы и не заражают здоровых. Когда , тогда инфицирование способны заражать восприимчивых к болезни особей. Для этого случая:

$\frac{dS}{dt} = -\alpha{S}\\$

$\frac{dI}{dt} = \alpha{S}-\beta{I}\\$

Для случая, когда :

$\frac{dS}{dt} = 0\\$

$\frac{dI}{dt} = -\beta{I}\\$

# Программная реализация

## Подготовка к моделированию

Все данные соответствуют варианту 14 = .

**Инициализация библиотек**

import numpy as np  
import matplotlib.pyplot as plt  
from scipy.integrate import odeint  
  
from jupyterthemes import jtplot  
jtplot.style(context='notebook', fscale=1.2, spines=False, gridlines='--')

**Начальные данные и необходимые функции**

N = 19500  
t0 = 0  
I0 = 88  
R0 = 25  
S0 = N - R0 - I0  
  
alpha = 0.01  
beta = 0.02  
  
x0 = np.array([S0, I0, R0])  
t = np.arange(0, 200, 0.01)

Объявим необходимые функции, исходя из данной нам информации в теоретической справке.

def dx\_less(x,t):  
 dS\_dt = 0.0  
 dI\_dt = -beta\*x[1]  
 dR\_dt = beta\*x[1]  
 return [dS\_dt, dI\_dt, dR\_dt]  
   
def dx\_greater(x,t):  
 dS\_dt = -alpha\*x[0]  
 dI\_dt = alpha\*x[0] - beta\*x[1]  
 dR\_dt = beta\*x[1]  
 return [dS\_dt, dI\_dt, dR\_dt]

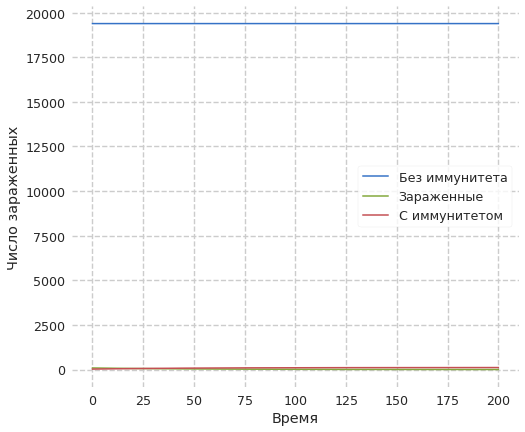
Заложим в переменные решения для наших СДУ с помощью функции odeint модуля scipy.integrate.

y\_less = odeint(dx\_less, x0, t)  
y\_greater = odeint(dx\_greater, x0, t)

## Построение графиков для модели

### Модель при

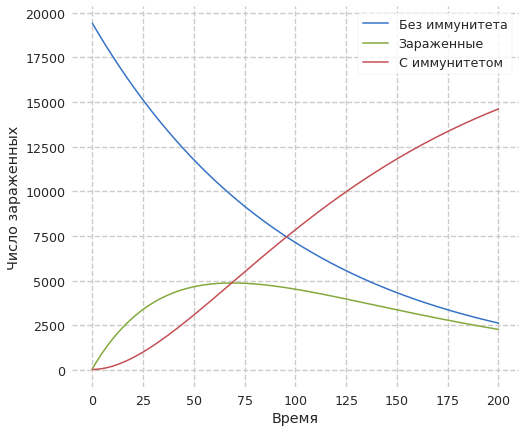
S = [e for e in y\_less[:,0]]  
I = [e for e in y\_less[:,1]]  
R = [e for e in y\_less[:,2]]  
  
plt.plot(t, S, label='Без иммунитета')  
plt.plot(t, I, label='Зараженные')  
plt.plot(t, R, label='С иммунитетом')  
plt.ylabel('Число зараженных')  
plt.xlabel('Время')  
plt.legend(loc='best')  
plt.show()



Модель при

### Модель при

S = [e for e in y\_greater[:,0]]  
I = [e for e in y\_greater[:,1]]  
R = [e for e in y\_greater[:,2]]  
  
plt.plot(t, S, label='Без иммунитета')  
plt.plot(t, I, label='Зараженные')  
plt.plot(t, R, label='С иммунитетом')  
plt.legend(loc='best')  
plt.ylabel('Число зараженных')  
plt.xlabel('Время')  
plt.show()



Модель при

# Выводы

Мы изучили простейшую модель эпидемии, после чего успешно реализовали её с помощью языка Python и дополняющих его модулей.