

Групповой проект номер №5

Дисциплина “Научное программирование”

Супонина Анастасия Павловна, Лобов Михаил Сергеевич, Нирдоши Всеволод Раджендер

02 Ноября 2024

Кафедра теории вероятностей и кибербезопасности

Российский университет дружбы народов имени Патриса Лумумбы, Москва, Россия

Задача о эпидемии — это математическая модель, описывающая распространение инфекционных заболеваний в популяции.

Моделирование эпидемий важно для:

- прогнозирования инфекций
- разработки стратегий вакцинации
- контроля распространения болезней

Модель динамики эпидемии базируется на системе дифференциальных уравнений, которая описывает скорость изменения численности этих трёх групп.

Для исследования этой задачи используют разные модели. Мы будем использовать модель SIR.



Рис. 1 Диаграмма переходов между состояниями

В нашей задаче дополнительным условием является значение `crit_I`, которое отображает критическое количество зараженных, при котором если $I > \text{crit_I}$, то эпидемия распространяется.

Получаем дифференциальные уравнения:

Скорость изменение числа восприимчивых к болезни здоровых особей ($S(t)$):

$$\frac{dS}{dt} = \begin{cases} -\alpha S, & \text{если } I(t) > I^* \\ 0, & \text{если } I(t) \leq I^* \end{cases}$$

Скорость изменение числа инфицированных особей, являющихся распространителями ($I(t)$):

$$\frac{dI}{dt} = \begin{cases} \alpha S - \beta I, & \text{если } I(t) > I^* \\ -\beta I, & \text{если } I(t) \leq I^* \end{cases}$$

Скорость изменение числа особей с иммунитетом (переболевших) ($R(t)$):

$$\frac{dR}{dt} = \beta I$$

где α и β - коэффициенты заболевания и выздоровления соответственно.

Julia предоставляет мощную библиотеку для решения обыкновенных дифференциальных уравнений (ODE) с помощью `DifferentialEquations.jl`. С помощью функции `ODEProblem` можно задать дифференциальное уравнение, начальные условия и временной промежуток для решения. Использование этой библиотеки позволят нам избежать того, что мы решаем не просто дифференциальные уравнения, а алгебродифференциальные уравнения, таким образом при решении задачи таким методом, число людей в нашей ограниченной популяции всегда остается одинаковым.

```
import Pkg
Pkg.add("DifferentialEquations")
```

```
prob_combo = ODEProblem(combo_case!, u0, tspan, p_combo)
```

Для построения графиков используем библиотеку **Plots**. Решение можно легко визуализировать на графике, чтобы увидеть, как изменяются популяции во времени (например, восприимчивые, зараженные, выздоровевшие).

```
import Pkg
```

```
Pkg.add("Plots")
```

```
sol_combo = solve(prob_combo)
```

```
plot(sol_combo, title="Комбинированная фаза: с учетом критического  
порога", label=["S" "I" "R"])
```

Параметры задачи

```
N = 7777 # Общее количество особей в популяции
# Она делится на 3 группы (по обозначениям SIR) :
I = 700   # Количество инфицированных особей
R = 0     # Количество выздоровевших особей
S = N - I - R # Количество здоровых, восприимчивых к инфекции особей
# В дифференциальных уравнениях модели у нас появляются два коэффициента:
a = 0.007 # Коэффициент заболевания
b = 0.003 # Коэффициент выздоровления
# Если  $I < \text{crit\_I}$ , то все больные изолированы.
crit_I = 90 # Критическое значение инфицированных
# Начальные условия
u0 = [S, I, R]
tspan = (0.0, 2500.0)
```


Всех заболевших получилось изолировать ($I(t) \leq I^*$).

Определение первой фазы, где происходит только выздоровление

```
function first_case!(du, u, p, t)
```

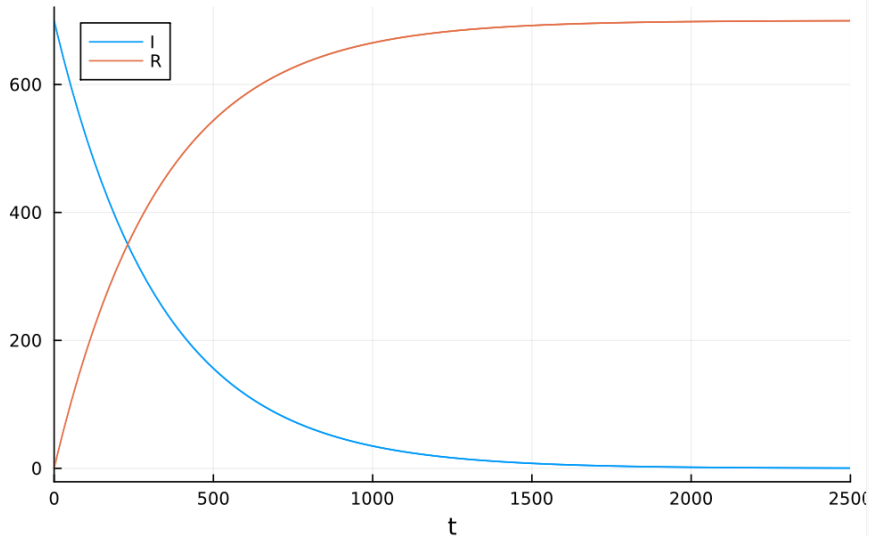
```
    _, b = p
```

```
    du[1] = -u[1] * b  # Уменьшается за счет выздоровления
```

```
    du[2] = u[1] * b   # Увеличивается
```

```
end
```

Первая фаза: только выздоровление



Второй случай

Всех заболевших НЕ получилось изолировать ($I(t) > I^*$).

Определение второй фазы, где происходят как заражения, так и выздоровления

```
function second_case!(du, u, p, t)
```

```
    a, b = p
```

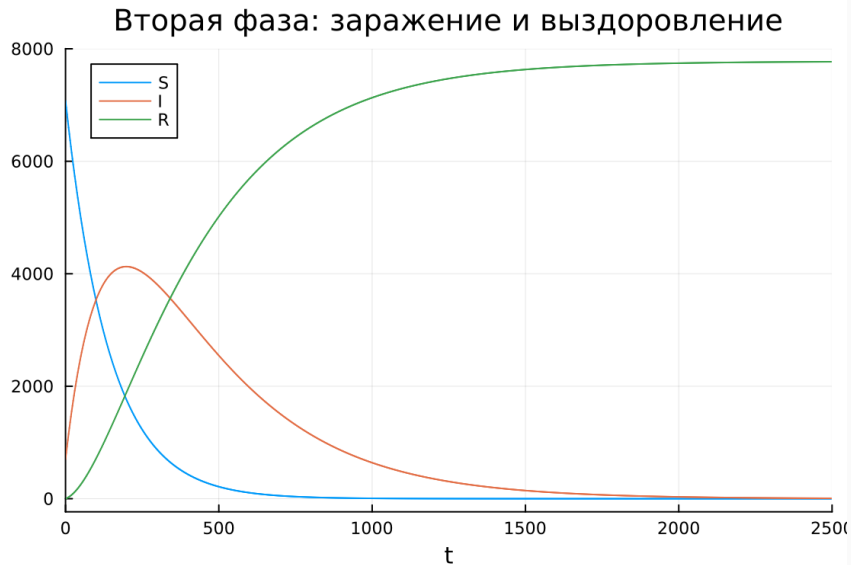
```
    du[1] = -a * u[1]                # Уменьшается за счет заражений
```

```
    du[2] = a * u[1] - u[2] * b      # Изменяется из-за заражений и
```

```
    #выздоровлений
```

```
    du[3] = u[2] * b                # Увеличивается
```

```
end
```



Определение комбинированной фазы с учетом критического порога

```
function combo_case!(du, u, p, t)
```

```
    a, b, crit_I = p
```

```
    if u[2] > crit_I
```

```
        du[1] = -a * u[1] # Уменьшается за счет заражений
```

```
        du[2] = a * u[1] - u[2] * b
```

```
        du[3] = u[2] * b
```

```
    else
```

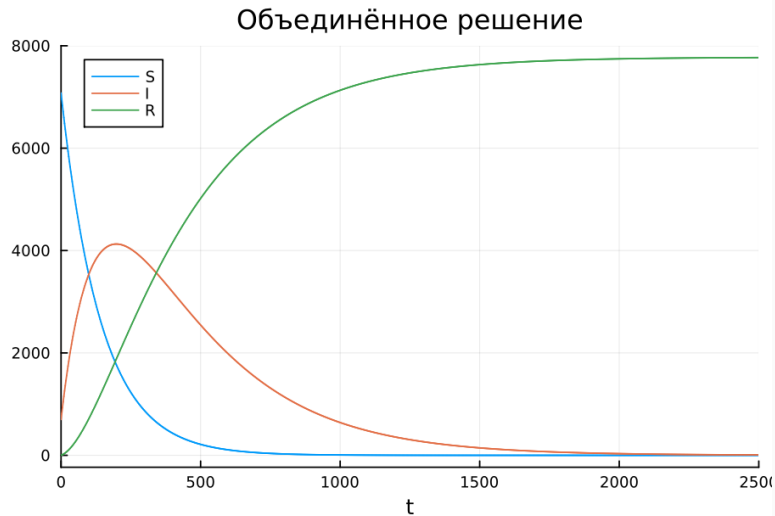
```
        du[1] = 0 # Остается неизменным
```

```
        du[2] = -u[2] * b
```

```
        du[3] = u[2] * b
```

```
    end
```

```
end
```



Спасибо за внимание!