Отчет по лабораторной работе №6

по дисциплине: Математическое моделирование

Ким Михаил Алексеевич

Содержание

Список иллюстраций

1 Цель работы

Продолжить знакомство с функционалом языка программирования Julia, дополнительных библиотек (DifferentialEquations, Plots), интерактивного блокнота Pluto, а также интерактивной командной строкой REPL. Продолжить ознакомление с языком моделирования Modelica и программным обеспечением OpenModelica. Используя эти средства, описать задачу об эпидемии (используя измененную математическую модель SIR).

2 Задание

На одном острове вспыхнула эпидемия. Известно, что из всех проживающих на острове ($N = 11000$) в момент начала эпидемии ($t = 0$) число заболевших людей (являющихся распространителями инфекции) $I(0) = 111$, А число здоровых людей с иммунитетом к болезни $R(0) = 11$. Таким образом, число людей восприимчивых к болезни, но пока здоровых, в начальный момент времени $S(0) = N - I(0) - R(0)$.

Постройте графики изменения числа особей в каждой из трех групп. Рассмотрите, как будет протекать эпидемия в случае:

1. Если $I(0) \leq I^*$
2. Если $I(0) > I^*$

3 Теоретическое введение

Задача текущей лабораторной работы сводится к построению математической модели, достаточно сильно похожей на модель SIR. Сначала будет дан материал о модели «Susceptible-Infectious-Recovered», а далее будут рассмотрены различия данной модели и модели, используемой при выполнении лабораторной работы.

3.1 Модель SIR

Модель SIR - это математическая модель, используемая для описания распространения инфекционных заболеваний в популяции. Аббревиатура SIR означает «Susceptible-Infectious-Recovered». Из расшифровки аббревиатуры следует, что модель разделяет популяцию на три группы: восприимчивые (susceptible), инфицированные (infectious) и выздоровевшие (recovered).

В модели SIR инфекционное заболевание передается от инфицированных к восприимчивым через непосредственный контакт. Когда восприимчивый контактирует с инфицированным, есть определенная вероятность заражения, которая зависит от свойств возбудителя и сопротивляемости организма. После того, как восприимчивый заразился, он становится инфицированным, и тем самым переходит в группу infectious.

Когда инфицированный выздоравливает, он переходит в группу recovered. В отличие от других моделей, таких как SEIR, модель SIR не учитывает длительности инкубационного периода или время восстановления, и считает, что инфицированные остаются в одном состоянии до тех пор, пока не выздоровеют [1].

Модель SIR представляется системой трех дифференциальных уравнений, которые описывают динамику численности каждой группы в зависимости от времени. Эти уравнения могут быть использованы для прогнозирования темпов распространения заболевания и оценки эффективности мер по его контролю.

1. Уравнение числа восприимчивых (S):

   $$\frac{dS}{dt} = -\frac{\beta IS}{N},$$

   где β — коэффициент интенсивности контактов индивидов с последующим инфицированием; $S(t)$ — численность восприимчивых индивидов в момент времени t; $I(t)$ — численность инфицированных индивидов в момент времени t; N — объем популяции.

   Первое уравнение описывает изменение численности восприимчивых с течением времени. Уравнение показывает, что изменение числа здоровых (и при этом восприимчивых к заболеванию) индивидуумов уменьшается со временем пропорционально числу контактов с инфицированными. После контакта происходит заражение, восприимчивый переходит в состояние инфицированного.

2. Уравнение числа инфицированных (I):

   $$\frac{dI}{dt} = \frac{\beta IS}{N} - \gamma I,$$

   где γ — коэффициент интенсивности выздоровления инфицированных индивидов.

   Второе уравнение описывает изменение числа инфицированных с течением времени. Уравнение показывает, что скорость увеличения числа заразившихся растет пропорционально числу контактов здоровых и инфицированных и уменьшается по мере выздоровления последних.

3. Уравнение числа выздоровевших (R):

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

где $R(t)$ — численность переболевших индивидов в момент времени t.

Третье уравнение демонстрирует, что число выздоровевших в единицу времени пропорционально числу инфицированных. Иначе говоря, каждый заболевший через некоторое время должен поправиться.

Стоит отметить, что сумма численностей трех групп всегда остается постоянной, т.е. $S + I + R = N$. Коэффициент $R_0 = \frac{\beta}{\gamma}$ называется **«базовым коэффициентом воспроизведения»** [2]. Для каждой болезни есть собственный коэффициент R_0. К примеру, у COVID-19 он находится в пределах $2.4 - 2.9$ [3].

Модель SIR может быть использована для прогнозирования темпов распространения заболевания и оценки эффективности мер по его контролю, таких как вакцинация, карантин, социальная дистанцирование и т.д. Также, в зависимости от начальных условий, коэффициента инфицирования, коэффициента выздоровления и других коэффициентов, модель может быть использована для исследования различных вариантов эпидемических сценариев.

Хотя модель SIR довольно проста, она оказывается достаточно полезной для анализа динамики распространения заболевания в популяции и понимания того, какие меры контроля наиболее эффективны в определенных ситуациях [4].

3.2 Задача об эпидемии

Отличия модели, предлагаемой для описания в лабораторной работы, от вышеуказанной модели SIR таковы:

1. Введен дополнительный параметр: I^* — критическое значение $I(t)$, после превышения которого инфицированные способны заражать воспри-

имчивых. До этого критического значения инфицированные не заражают восприимчивых.

2. Изменены стандартные символы, отождествляющие коэффициенты: $\beta\frac{I}{N} \to \alpha$ (коэффициент заболеваемости), $\gamma \to \beta$ (коэффициент выздоровления).
3. В соответствии с предыдущими пунктами изменена система уравнений [5]:

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

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

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

4 Выполнение лабораторной работы

4.1 Pluto.jl

4.1.1 Задание №1

1. Пишем программу, воспроизводящую модель на языке программирования Julia с использованием интерактивного блокнота Pluto (рис. 4.1, 4.2, 4.3, 4.4, 4.5, 4.6).

```
begin
    import Pkg
    Pkg.activate()
    using DifferentialEquations
    using LaTeXStrings
    import Plots
end
```

```
begin
    const ⊠ = 0.75
    const ⊠ = 0.25
    @show ⊠ / ⊠

    const N = 11000
    const I⊠ = 111
    const R⊠ = 11
```

```
    const S⊠ = N - I⊠ - R⊠
    const I⊠ = 200
    @show S⊠

    "Начальные условия: u⊠[1] - S⊠, u⊠[1] - I⊠, u⊠[1] - R⊠"
    u⊠ = [S⊠, I⊠, R⊠]

    "Период времени"
    T = (0.0, 30.0)
end

"Правая часть нашей системы, p, t не используются. u[1] - S, u[2] - I, u[3]
function F!(du, u, p, t)
    if u[2] > I⊠
        println("I(t) > I⊠, I(t) = ", u[2])
        du[1] = - ⊠ * u[1]
        du[2] = ⊠ * u[1] - ⊠ * u[2]
    else
        println("I(t) ⩽ I⊠, I(t) = ", u[2])
        du[1] = 0
        du[2] = - ⊠ * u[2]
    end
    du[3] = ⊠ * u[2]
end

prob = ODEProblem(F!, u⊠, T)

sol = solve(prob, saveat=0.2)

begin
    const ss = []
    const ii = []
```

```
    const rr = []
    for u in sol.u
        s, i, r = u
        push!(ss, s)
        push!(ii, i)
        push!(rr, r)
    end
    time = sol.t
    time
end

begin
    fig = Plots.plot(
        dpi=150,
        grid=:xy,
        gridcolor=:black,
        gridwidth=1,
        size=(800, 400),
        legend=:outerbottom,
        plot_title="Измененная модель SIR"
    )

    Plots.plot!(
        fig[1],
        time,
        [ss, ii, rr],
        color=[:blue :red :green],
        xlabel="t",
        ylabel="S(t), I(t), R(t)",
        label=["S(t) — количество здоровых, но восприимчивых к болезни" "I(t
```

```
    )
end
```

```
begin
    import Pkg
    Pkg.activate()
    using DifferentialEquations
    using LaTeXStrings
    import Plots
end
```

```
Activating project at `~/.julia/environments/v1.8`
```

Рис. 4.1: Импорт библиотек

T

Период времени

```
begin
    const α = 0.75
    const β = 0.25
    @show α / β

    const N = 11000
    const I₀ = 111
    const R₀ = 11
    const S₀ = N - I₀ - R₀
    const I* = 200
    @show S₀

    "Начальные условия: u₀[1] - S₀, u₀[1] - I₀, u₀[1] - R₀"
    u₀ = [S₀, I₀, R₀]

    "Период времени"
    T = (0.0, 30.0)
end
```

```
α / β = 3.0
S₀ = 10878
```

Рис. 4.2: Задание коэффициентов, критического значения (I^*), начальных условий, периода времени

F!

Правая часть нашей системы, p, t не используются. u[1] - S, u[2] - I, u[3] - R

```
"Правая часть нашей системы, p, t не используются. u[1] - S, u[2] - I, u[3] - R"
function F!(du, u, p, t)
    if u[2] > I⁺
        println("I(t) > I⁺, I(t) = ", u[2])
        du[1] = - α * u[1]
        du[2] = α * u[1] - β * u[2]
    else
        println("I(t) ≤ I⁺, I(t) = ", u[2])
        du[1] = 0
        du[2] = - β * u[2]
    end
    du[3] = β * u[2]
end
```

Рис. 4.3: Запись системы уравнений в виде функции (соблюдены условия, связанные с критическим значением I)

```
prob = ODEProblem with uType Vector{Int64} and tType Float64. In-place: true
       timespan: (0.0, 30.0)
       u0: 3-element Vector{Int64}:
        10878
          111
           11
```

```
prob = ODEProblem(F!, u₀, T)
```

sol =

	timestamp	value1	value2	value3
1	0.0	10878.0	111.0	11.0
2	0.2	10878.0	105.586	16.4135
3	0.4	10878.0	100.437	21.563
4	0.6	10878.0	95.5386	26.4614
5	0.8	10878.0	90.8791	31.1209
6	1.0	10878.0	86.4469	35.5531
7	1.2	10878.0	82.2308	39.7692
8	1.4	10878.0	78.2204	43.7796
9	1.6	10878.0	74.4055	47.5945
10	1.8	10878.0	70.7767	51.2233
⋮ more				

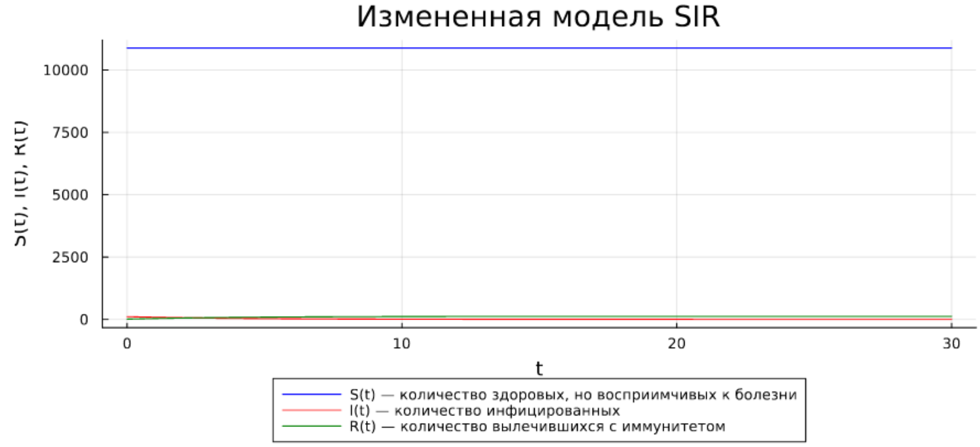
```
sol = solve(prob, saveat=0.2)
```

```
I(t) ≤ I*, I(t) = 111.0
I(t) ≤ I*, I(t) = 111.0
I(t) ≤ I*, I(t) = 110.81039235253624
I(t) ≤ I*, I(t) = 110.58599047656996
I(t) ≤ I*, I(t) = 110.16234116814493
I(t) ≤ I*, I(t) = 108.70931973370503
I(t) ≤ I*, I(t) = 108.50773827025584
I(t) ≤ I*, I(t) = 108.45749201854665
I(t) ≤ I*, I(t) = 108.45806984510264
I(t) ≤ I*, I(t) = 106.25953640119796
I(t) ≤ I*, I(t) = 104.08558920084458
I(t) ≤ I*, I(t) = 96.80931762611455
I(t) ≤ I*, I(t) = 95.81394851254295
I(t) ≤ I*, I(t) = 95.56786027890517
I(t) ≤ I*, I(t) = 95.62726362547826
I(t) ≤ I*, I(t) = 92.1086301919546
I(t) ≤ I*, I(t) = 88.75050129631919
I(t) ≤ I*, I(t) = 77.69981023311077
I(t) ≤ I*, I(t) = 76.16224744621954
I(t) ≤ I*, I(t) = 75.7863767400629
I(t) ≤ I*, I(t) = 76.08991785862682
I(t) ≤ I*, I(t) = 72.46456792993294
I(t) ≤ I*, I(t) = 69.08655643569882
```

Рис. 4.4: Решение задачи (сохранение происходит каждые 0.2 секунды)

```
▸[0.0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0, 2.2, 2.4, 2.6, 2.8, 3.0, 3.2, 3.4, 3
begin
    const ss = []
    const ii = []
    const rr = []
    for u in sol.u
        s, i, r = u
        push!(ss, s)
        push!(ii, i)
        push!(rr, r)
    end
    time = sol.t
    time
end
```

Рис. 4.5: Формирование четырех массивов, содержащих значения $S(t),\ I(t),\ R(t),\ t$




```
begin
    fig = Plots.plot(
        dpi=150,
        grid=:xy,
        gridcolor=:black,
        gridwidth=1,
        size=(800, 400),
        legend=:outerbottom,
        plot_title="Измененная модель SIR"
    )

    Plots.plot!(
        fig[1],
        time,
        [ss, ii, rr],
        color=[:blue :red :green],
        xlabel="t",
        ylabel="S(t), I(t), R(t)",
        label=["S(t) — количество здоровых, но восприимчивых к болезни" "I(t) — количество инфицированных" "R(t) — количество вылечившихся с иммунитетом"]
    )
end
```

Рис. 4.6: Отрисовка графика

4.1.2 Задание №2

1. Изменено значение I^*, которое теперь меньше $I(0)$. Остальные блоки кода оставляем без изменений. Любуемся результатом (рис. 4.7, 4.8).

```
begin
    const ⊠ = 0.75
    const ⊠ = 0.25
    @show ⊠ / ⊠
```

```
    const N = 11000
    const I⊠ = 111
    const R⊠ = 11
    const S⊠ = N - I⊠ - R⊠
    const I⊠ = 100
    @show S⊠

    "Начальные условия: u⊠[1] - S⊠, u⊠[1] - I⊠, u⊠[1] - R⊠"
    u⊠ = [S⊠, I⊠, R⊠]

    "Период времени"
    T = (0.0, 38.0)
end
```

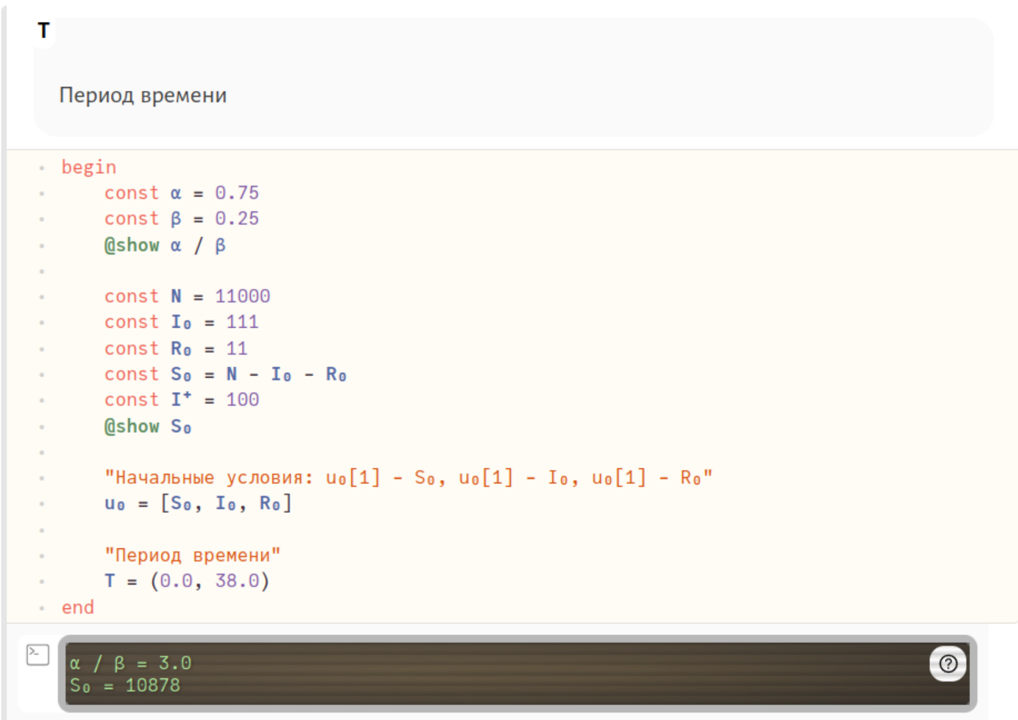


Рис. 4.7: Изменение значения I^*

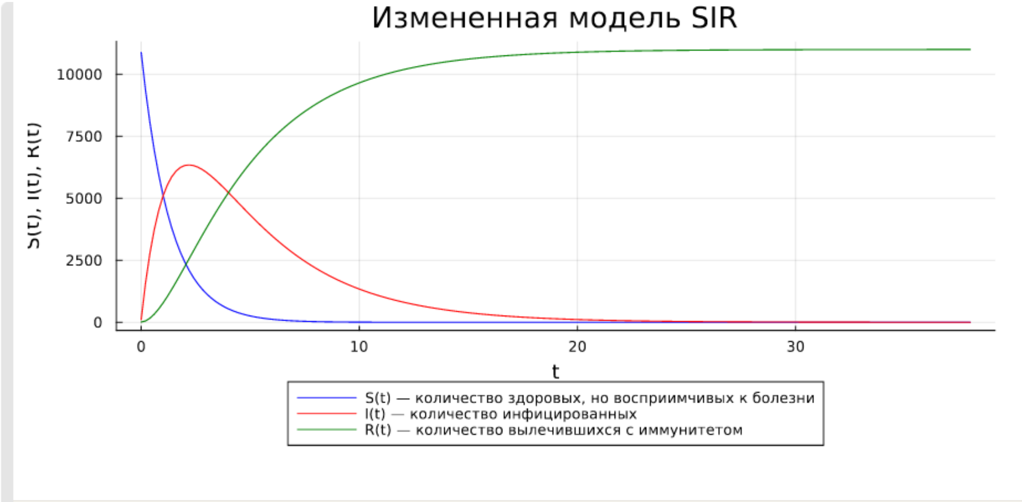


Рис. 4.8: Результат в виде графика

4.2 Julia

4.2.1 Задание №1

1. Код на Julia в файле аналогичен тому же, написанному с использованием Pluto (рис. 4.9, 4.10). Единственные различия:

    - блоки перенесены в файл в виде построчного алгоритма без повторяющихся ‘begin’ и ‘end’;
    - измененный синтаксис подключения библиотек;
    - выгрузка графиков в виде изображений при помощи метода в последней строчке кода.

```
using DifferentialEquations
using Plots



const ⊠ = 0.75
const ⊠ = 0.25
```

```
@show ⊠ / ⊠

const N = 11000
const I⊠ = 111
const R⊠ = 11
const S⊠ = N - I⊠ - R⊠
const I⊠ = 200
@show S⊠


"Начальные условия: u⊠[1] - S⊠, u⊠[1] - I⊠, u⊠[1] - R⊠"
u⊠ = [S⊠, I⊠, R⊠]

"Период времени"
T = (0.0, 30.0)

"Правая часть нашей системы, p, t не используются. u[1] - S, u[2] - I, u[3]
function F!(du, u, p, t)
    if u[2] > I⊠
        println("I(t) > I⊠, I(t) = ", u[2])
        du[1] = - ⊠ * u[1]
        du[2] = ⊠ * u[1] - ⊠ * u[2]
    else
        println("I(t) ⩽ I⊠, I(t) = ", u[2])
        du[1] = 0
        du[2] = - ⊠ * u[2]
    end
    du[3] = ⊠ * u[2]
end
```

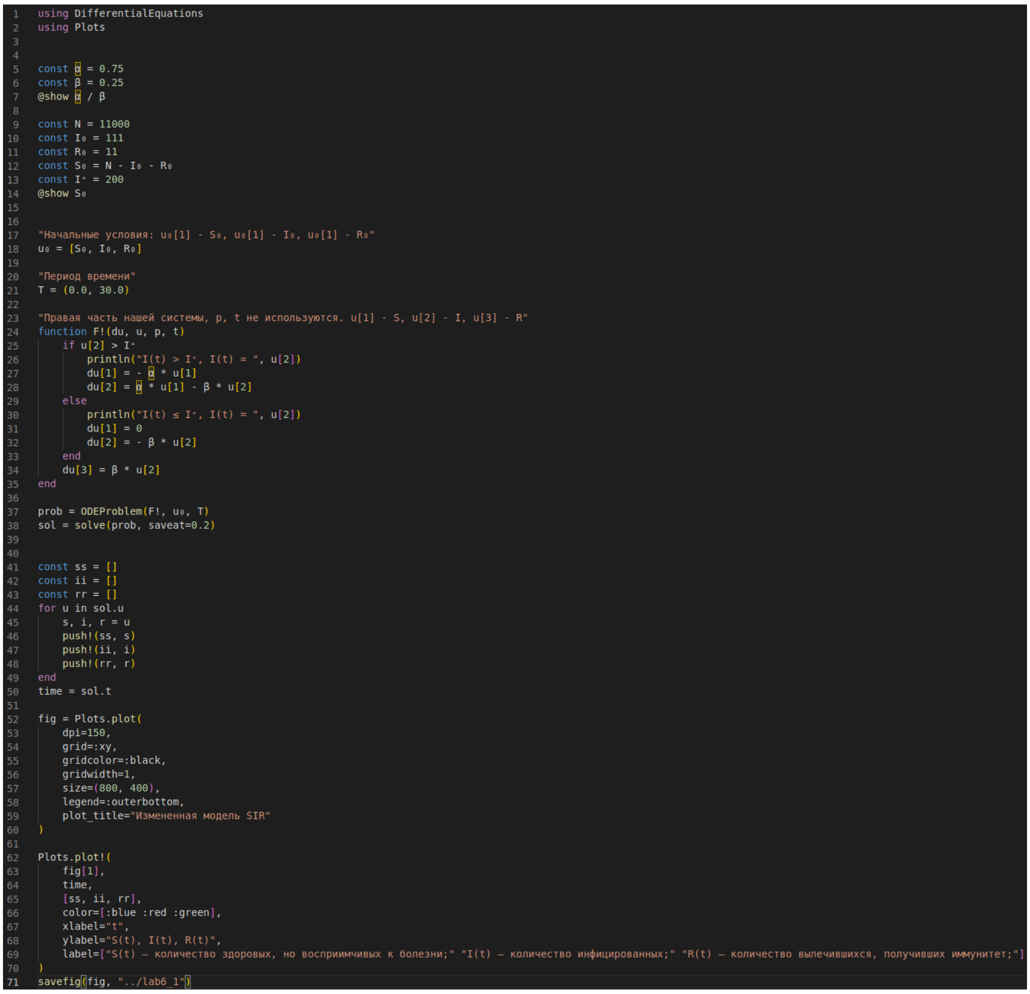
```
prob = ODEProblem(F!, u⊠, T)
sol = solve(prob, saveat=0.2)


const ss = []
const ii = []
const rr = []
for u in sol.u
    s, i, r = u
    push!(ss, s)
    push!(ii, i)
    push!(rr, r)
end
time = sol.t

fig = Plots.plot(
    dpi=150,
    grid=:xy,
    gridcolor=:black,
    gridwidth=1,
    size=(800, 400),
    legend=:outerbottom,
    plot_title="Измененная модель SIR"
)

Plots.plot!(
    fig[1],
    time,
```

```
    [ss, ii, rr],
    color=[:blue :red :green],
    xlabel="t",
    ylabel="S(t), I(t), R(t)",
    label=["S(t) — количество здоровых, но восприимчивых к болезни" "I(t) —
)
savefig(fig, "../lab6_1")
```



```
using DifferentialEquations
using Plots


const α = 0.75
const β = 0.25
@show α / β

const N = 11000
const I₀ = 111
const R₀ = 11
const S₀ = N - I₀ - R₀
const I⁺ = 200
@show S₀


"Начальные условия: u₀[1] - S₀, u₀[1] - I₀, u₀[1] - R₀"
u₀ = [S₀, I₀, R₀]

"Период времени"
T = (0.0, 30.0)

"Правая часть нашей системы, p, t не используются. u[1] - S, u[2] - I, u[3] - R"
function F!(du, u, p, t)
    if u[2] > I⁺
        println("I(t) > I⁺, I(t) = ", u[2])
        du[1] = - α * u[1]
        du[2] = α * u[1] - β * u[2]
    else
        println("I(t) ≤ I⁺, I(t) = ", u[2])
        du[1] = 0
        du[2] = - β * u[2]
    end
    du[3] = β * u[2]
end

prob = ODEProblem(F!, u₀, T)
sol = solve(prob, saveat=0.2)


const ss = []
const ii = []
const rr = []
for u in sol.u
    s, i, r = u
    push!(ss, s)
    push!(ii, i)
    push!(rr, r)
end
time = sol.t

fig = Plots.plot(
    dpi=150,
    grid=:xy,
    gridcolor=:black,
    gridwidth=1,
    size=(800, 400),
    legend=:outerbottom,
    plot_title="Измененная модель SIR"
)

Plots.plot!(
    fig[1],
    time,
    [ss, ii, rr],
    color=[:blue :red :green],
    xlabel="t",
    ylabel="S(t), I(t), R(t)",
    label=["S(t) — количество здоровых, но восприимчивых к болезни;" "I(t) — количество инфицированных;" "R(t) — количество вылечившихся, получивших иммунитет;"]
)
savefig(fig, "../lab6_1")
```

Рис. 4.9: Код программы на Julia. Аналогичен коду задания для Pluto.jl

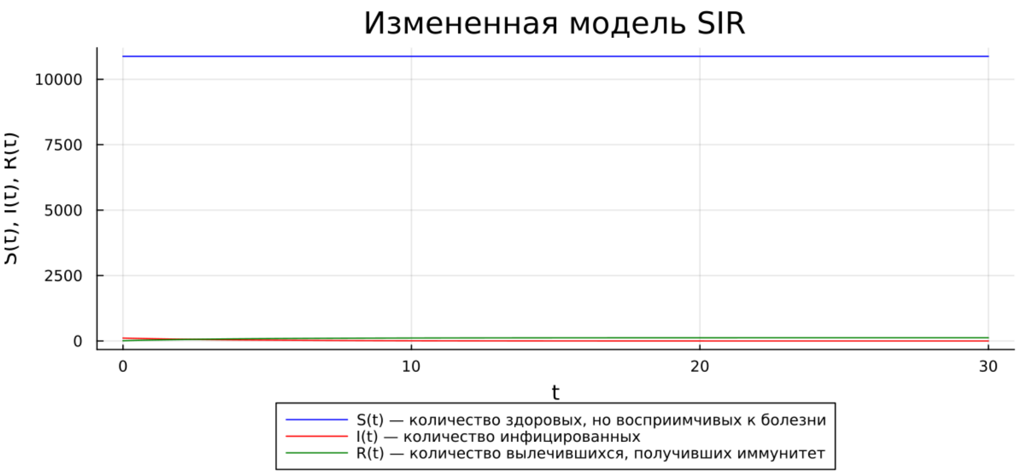


Рис. 4.10: Результат в виде графика

4.2.2 Задание №2

1. Изменяем значение I^*, которое теперь меньше $I(0)$, и любуемся результатом (подробное объяснение давалось в предыдущей главе) (рис. 4.11, 4.12).

```
const N = 11000
const I⊠ = 111
const R⊠ = 11
const S⊠ = N - I⊠ - R⊠
const I⊠ = 100
@show S⊠
```

```
const N = 11000
const I₀ = 111
const R₀ = 11
const S₀ = N - I₀ - R₀
const I⁺ = 100
@show S₀
```

Рис. 4.11: Измененная часть кода

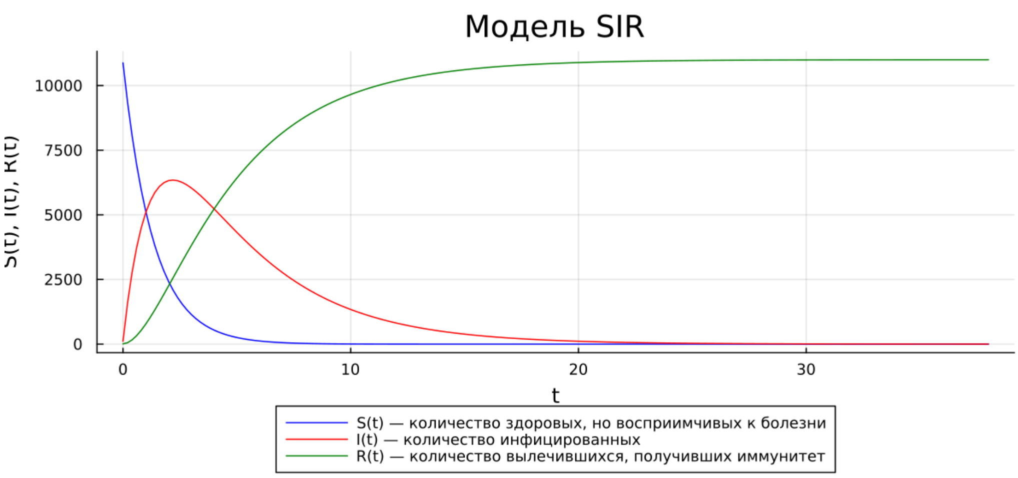


Рис. 4.12: Результат в виде графиков

4.3 Modelica

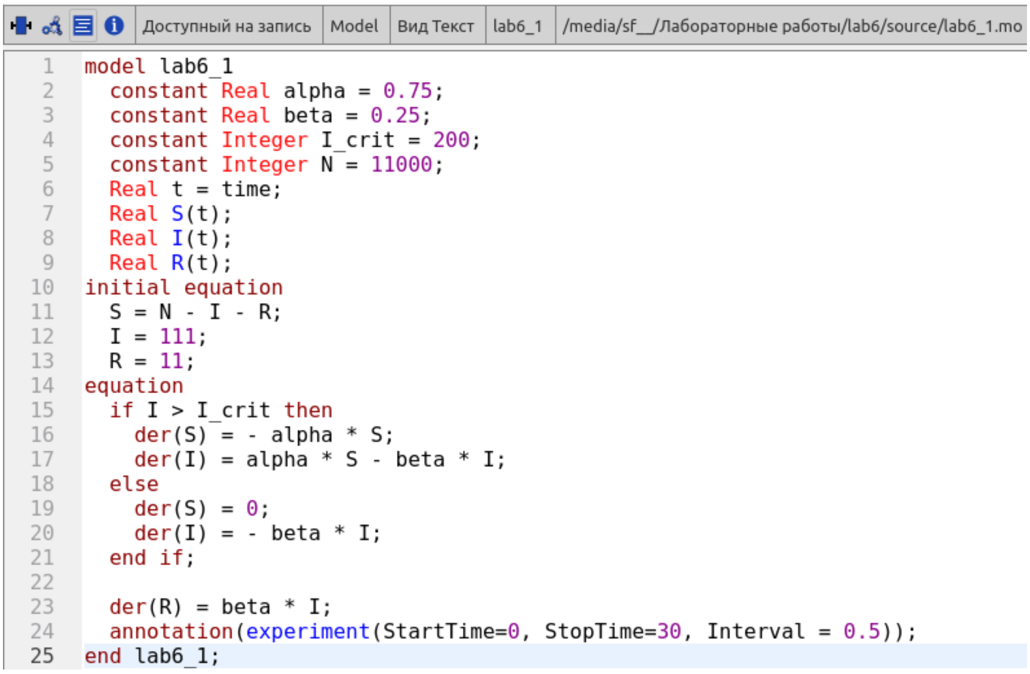
4.3.1 Задание №1

1. По аналогии с Julia пишем программу, воспроизводящую измененную модель SIR на языке моделирования Modelica с использованием ПО

OpenModelica. Любуемся результатами (рис. 4.13, 4.14, 4.15).

```
model lab6_1
  constant Real alpha = 0.75;
  constant Real beta = 0.25;
  constant Integer I_crit = 200;
  constant Integer N = 11000;
  Real t = time;
  Real S(t);
  Real I(t);
  Real R(t);
initial equation
  S = N - I - R;
  I = 111;
  R = 11;
equation
  if I > I_crit then
    der(S) = - alpha * S;
    der(I) = alpha * S - beta * I;
  else
    der(S) = 0;
    der(I) = - beta * I;
  end if;

  der(R) = beta * I;
  annotation(experiment(StartTime=0, StopTime=30, Interval = 0.5));
end lab6_1;
```




```
model lab6_1
  constant Real alpha = 0.75;
  constant Real beta = 0.25;
  constant Integer I_crit = 200;
  constant Integer N = 11000;
  Real t = time;
  Real S(t);
  Real I(t);
  Real R(t);
initial equation
  S = N - I - R;
  I = 111;
  R = 11;
equation
  if I > I_crit then
    der(S) = - alpha * S;
    der(I) = alpha * S - beta * I;
  else
    der(S) = 0;
    der(I) = - beta * I;
  end if;

  der(R) = beta * I;
  annotation(experiment(StartTime=0, StopTime=30, Interval = 0.5));
end lab6_1;
```

Рис. 4.13: Определяем коэффициенты, критическое значение I^*, переменные от времени, начальные условия, систему уравнений (согласуется с условиями I^*), а также начальное/конечное время и частоту разбиения при симуляции

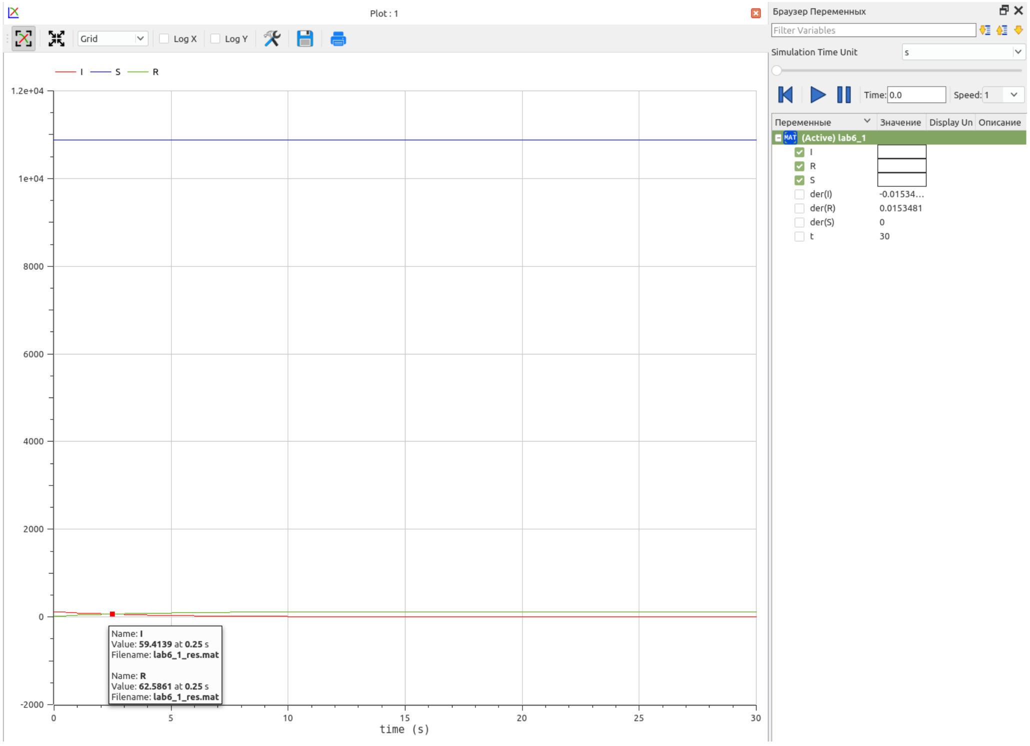


Рис. 4.14: Результат в виде графика зависимости $S, \ I, \ R$ от t

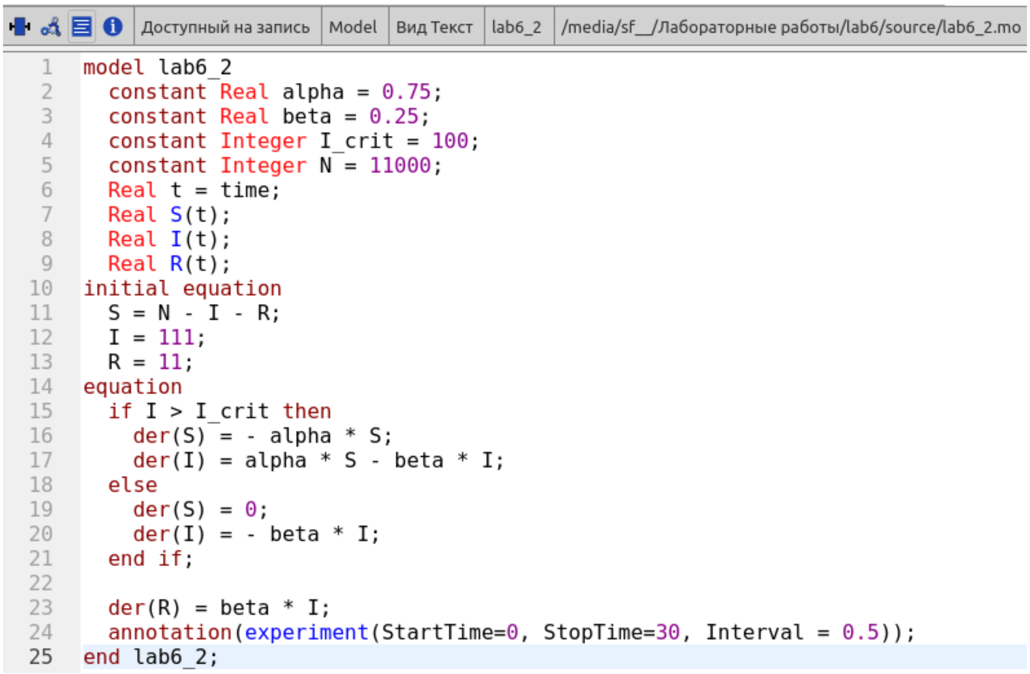
4.3.2 Задание №2

1. По аналогии с Julia пишем программу для второго случая. Любуемся результатами (рис. 4.15, 4.16).

```
model lab6_2
  constant Real alpha = 0.75;
  constant Real beta = 0.25;
  constant Integer I_crit = 100;
  constant Integer N = 11000;
  Real t = time;
  Real S(t);
  Real I(t);
  Real R(t);
```

```
initial equation
  S = N - I - R;
  I = 111;
  R = 11;
equation
  if I > I_crit then
    der(S) = - alpha * S;
    der(I) = alpha * S - beta * I;
  else
    der(S) = 0;
    der(I) = - beta * I;
  end if;

  der(R) = beta * I;
  annotation(experiment(StartTime=0, StopTime=30, Interval = 0.5));
end lab6_2;
```




```
model lab6_2
  constant Real alpha = 0.75;
  constant Real beta = 0.25;
  constant Integer I_crit = 100;
  constant Integer N = 11000;
  Real t = time;
  Real S(t);
  Real I(t);
  Real R(t);
initial equation
  S = N - I - R;
  I = 111;
  R = 11;
equation
  if I > I_crit then
    der(S) = - alpha * S;
    der(I) = alpha * S - beta * I;
  else
    der(S) = 0;
    der(I) = - beta * I;
  end if;

  der(R) = beta * I;
  annotation(experiment(StartTime=0, StopTime=30, Interval = 0.5));
end lab6_2;
```

Рис. 4.15: По сравнению с предыдущим случаем изменяется только значение I^*

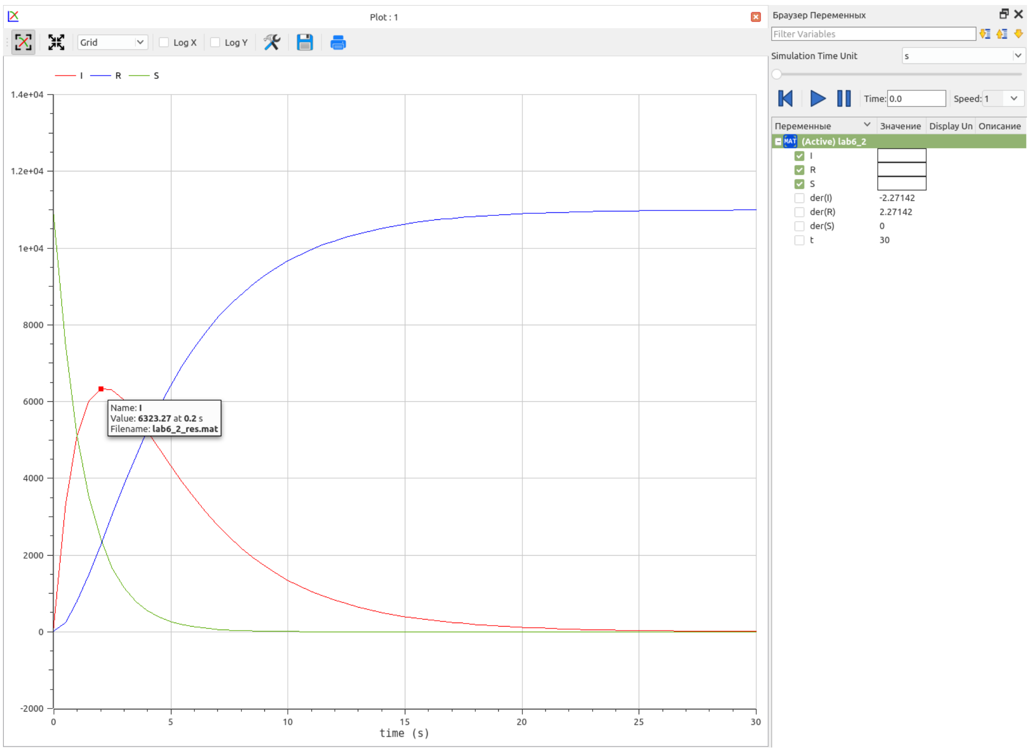


Рис. 4.16: Результат в виде графика зависимости S, I, R от t

5 Анализ результатов

На текущем примере построения математической модели, схожей с моделью SIR, мы можем продолжить сравнивать язык программирования Julia и язык моделирования Modelica. Говоря честно, по сравнению с анализом результатов при выполнении предыдущей лабораторной работы мало что изменилось: тенденция к сглаживанию негативных моментов при выполнении лабораторной работы на языке программирования Julia продолжается. Со временем и с новыми заданиями, решаемыми при помощи библиотеки DifferentialEquations, скорость написания программ на Julia почти сравнялась с таковой скоростью при использовании Modelica.

На обоих языках одинаково просто добавляются условия в уравнения, как в текущем случае. Однако, хочется заметить, что в Modelica в разы удобнее составлять уравнения, т.к. все переменные, зависящие от времени, подписываются заданными ранее символами в отличие от Julia, где каждой переменной соответствует элемент массива. Такая реализация может запутать, особенно при условии наличие трех и более переменных, зависящих от времени и используемых в системе. Это может привести к ошибкам, связанными с усидчивостью, при написании системы.

6 Выводы

Продолжил знакомство с функционалом языка программирования Julia, дополнительных библиотек (DifferentialEquations, Plots), интерактивного блокнота Pluto, а также интерактивной командной строкой REPL. Продолжил ознакомление с языком моделирования Modelica и программным обеспечением OpenModelica. Используя эти средства, описал математическую модель, схожую с моделью SIR.

Список литературы

1. SIR и разновидности: модели COVID-эпидемии в России [Электронный ресурс]. The AnyLogic Company, 2020. URL: https://www.anylogic.ru/blog/sir-i-raznovidnosti-modeli-covid-epidemii-v-rossii/.
2. Зараза, гостья наша [Электронный ресурс]. N + 1 Интернет-издание, 2019. URL: https://nplus1.ru/material/2019/12/26/epidemic-math.
3. Basic reproduction number [Электронный ресурс]. Wikimedia Foundation, Inc., 2023. URL: https://en.wikipedia.org/wiki/Basic_reproduction_number.
4. Compartmental models in epidemiology [Электронный ресурс]. Wikimedia Foundation, Inc., 2023. URL: https://en.wikipedia.org/wiki/Compartmental_models_in_epidemiology.
5. Задача об эпидемии [Электронный ресурс]. RUDN, 2023. URL: https://esystem.rudn.ru/mod/resource/view.php?id=967249.