Лабораторная работа №6

Задача об эпидемии

Ильин Андрей Владимирович

Содержание

# 1 Цель работы

Рассмотреть модель эпидемии. Построить вышеуказанную модель средствами OpenModellica и Julia.

# 2 Задачи

На одном острове вспыхнула эпидемия. Известно, что из всех проживающих на острове () в момент начала эпидемии () число заболевших людей (являющихся распространителями инфекции) , А число здоровых людей с иммунитетом к болезни . Таким образом, число людей восприимчивых к болезни, но пока здоровых, в начальный момент времени .

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

1. если
2. если

# 3 Среда

* Julia – это открытый свободный высокопроизводительный динамический язык высокого уровня, созданный специально для технических (математических) вычислений. Его синтаксис близок к синтаксису других сред технических вычислений, таких как Matlab и Octave. [1]
* OpenModelica — свободное открытое программное обеспечение для моделирования, симуляции, оптимизации и анализа сложных динамических систем. Основано на языке Modelica. [2]

# 4 Теоретическое введение

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. [3]

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.

Таким образом, скорость изменения числа меняется по следующему закону:

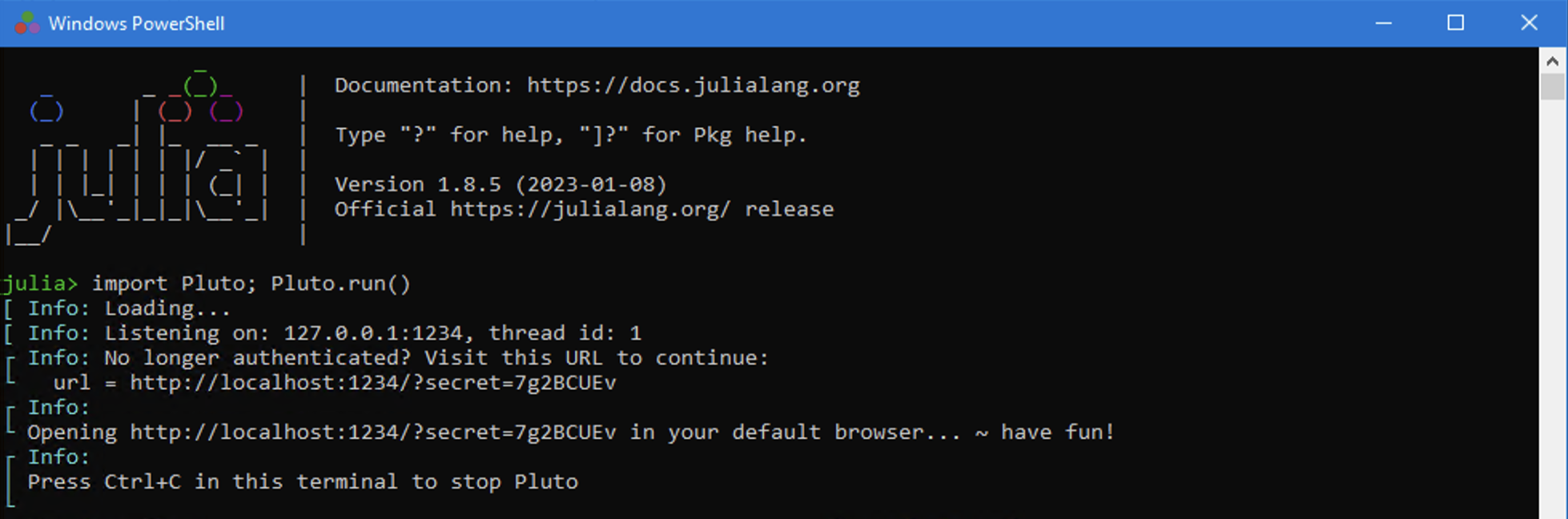
Поскольку каждая восприимчивая к болезни особь, которая, в конце концов, заболевает, сама становится инфекционной, то скорость изменения числа инфекционных особей представляет разность за единицу времени между заразившимися и теми, кто уже болеет и лечится, т.е.:

А скорость изменения выздоравливающих особей (при этом приобретающие иммунитет к болезни)

Постоянные пропорциональности , - это коэффициенты заболеваемости и выздоровления соответственно.

# 5 Выполнение лабораторной работы

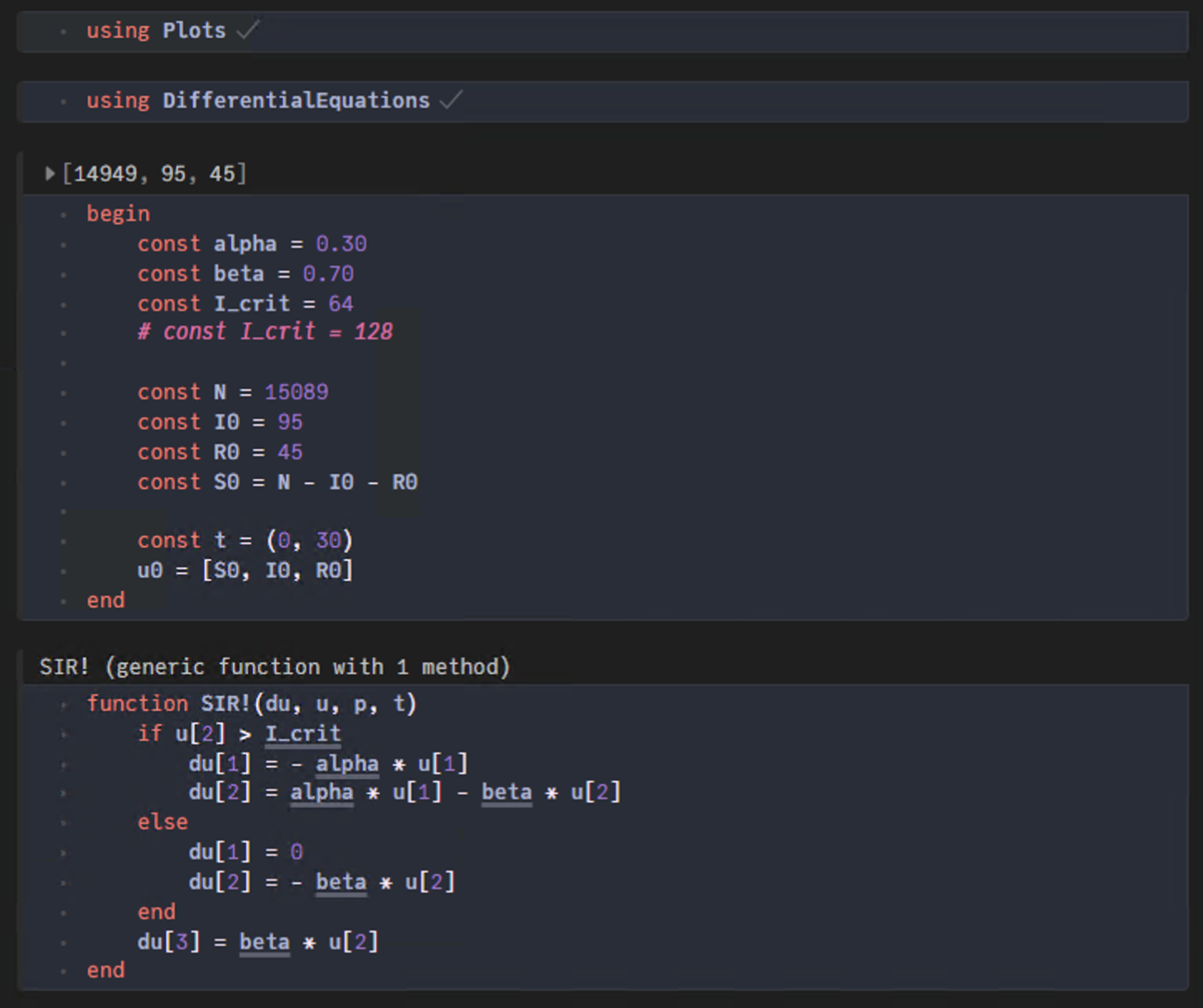
1. Начнем выполнения поставленных задач в Julia. Для этого запустим Pluto [4]. (рис. ??)



Julia. Запуск Pluto

1. Первым делом подкючим пакеты “Plots” [5] и “DifferentialEquations” [6]. Далее объявим начальные данные при помощи констант. Также объявим начальное условие для системы ДУ и промежуток времени, на котором будет проходить моделирование. После этого объявим функцию, представляющую систему. (рис. ??)

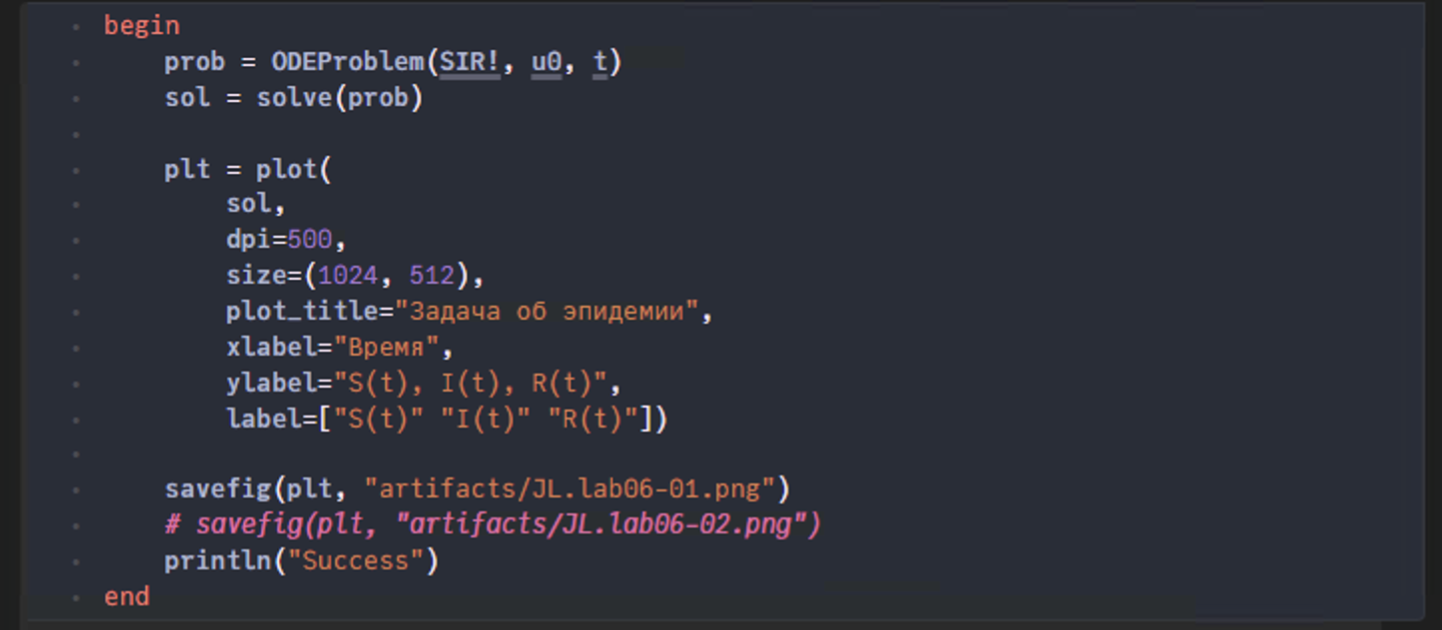
# подключение пакетов  
using Plots  
using DifferentialEquations  
  
# входные данные  
const alpha = 0.30  
const beta = 0.70  
const I\_crit = 64  
const N = 15089  
const I0 = 95  
const R0 = 45  
const S0 = N - I0 - R0  
  
const t = (0, 30) # промежуток времени  
u0 = [S0, I0, R0] # начальные условие ДУ  
  
function SIR!(du, u, p, t)  
 if u[2] > I\_crit  
 du[1] = - alpha \* u[1]  
 du[2] = alpha \* u[1] - beta \* u[2]  
 else  
 du[1] = 0  
 du[2] = - beta \* u[2]  
 end  
 du[3] = beta \* u[2]  
end



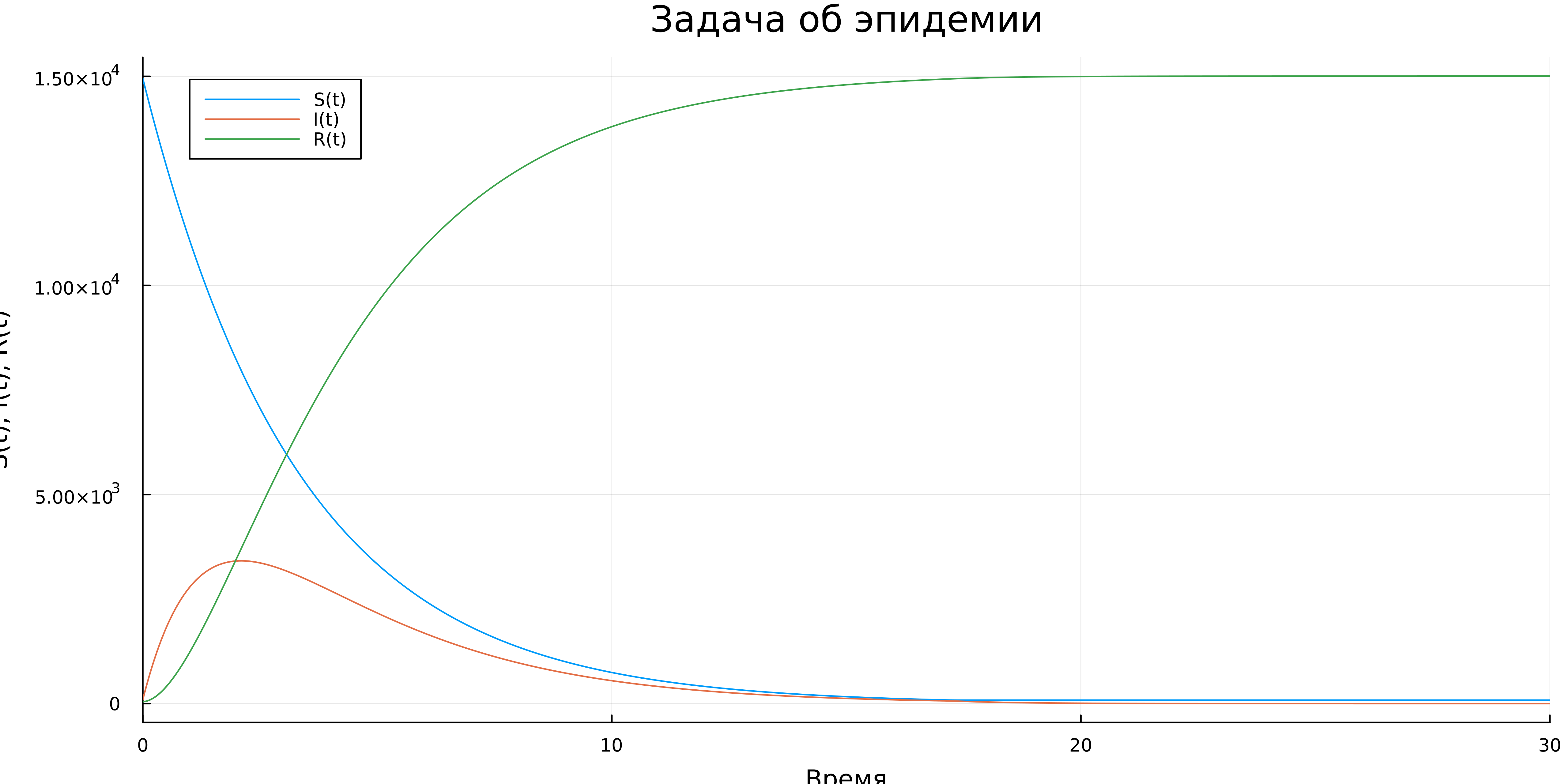
Julia. Скрипт (1). Задача об эпидемии ()

1. В следующей ячейке Pluto построим модель. При помощи ‘DifferentialEquations’ зададим и решим систему ДУ, после чего построим график решения и сохраним его. Далее запустим итоговый скрипт. (рис. ??, ??)

prob = ODEProblem(SIR!, u0, t)  
sol = solve(prob)  
  
plt = plot(  
 sol,  
 dpi=500,  
 size=(1024, 512),  
 plot\_title="Задача об эпидемии",  
 xlabel="Время",  
 ylabel="S(t), I(t), R(t)",  
 label=["S(t)" "I(t)" "R(t)"])  
  
savefig(plt, "artifacts/JL.lab06-01.png")  
println("Success")



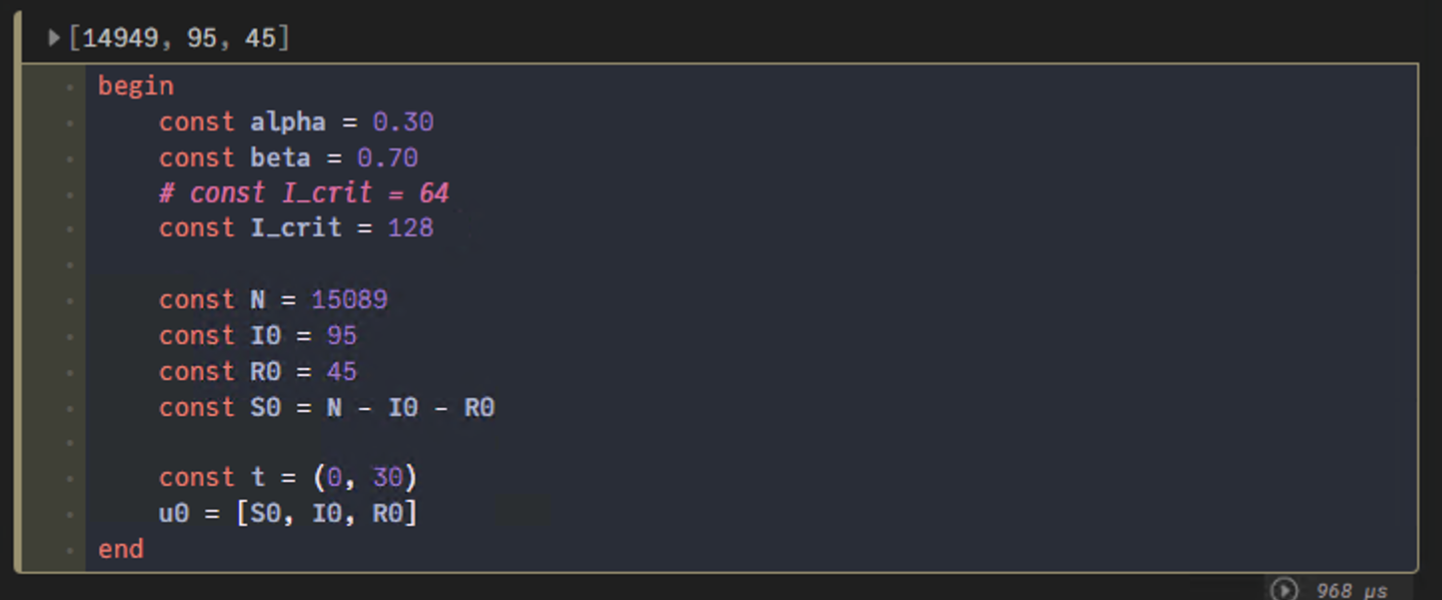
Julia. Скрипт (2). Задача об эпидемии ()



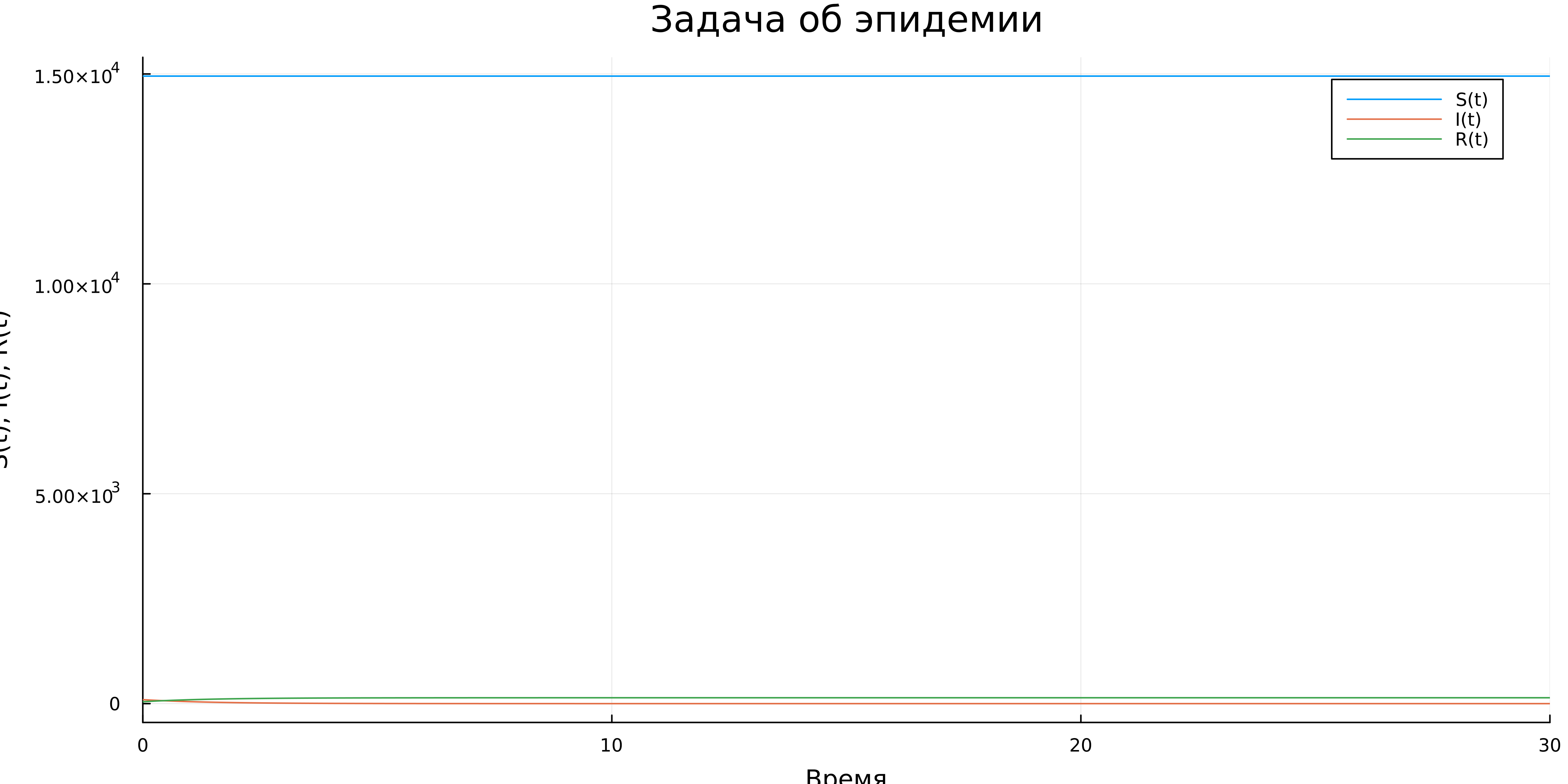
Julia. Модель. Задача об эпидемии ()

1. Изменим значения , так чтобы начальное число заболевших не превышало критическое значение. (рис. ??, ??)

const alpha = 0.30  
const beta = 0.70  
# const I\_crit = 64  
const I\_crit = 128  
  
const N = 15089  
const I0 = 95  
const R0 = 45  
const S0 = N - I0 - R0  
  
const t = (0, 30)  
u0 = [S0, I0, R0]



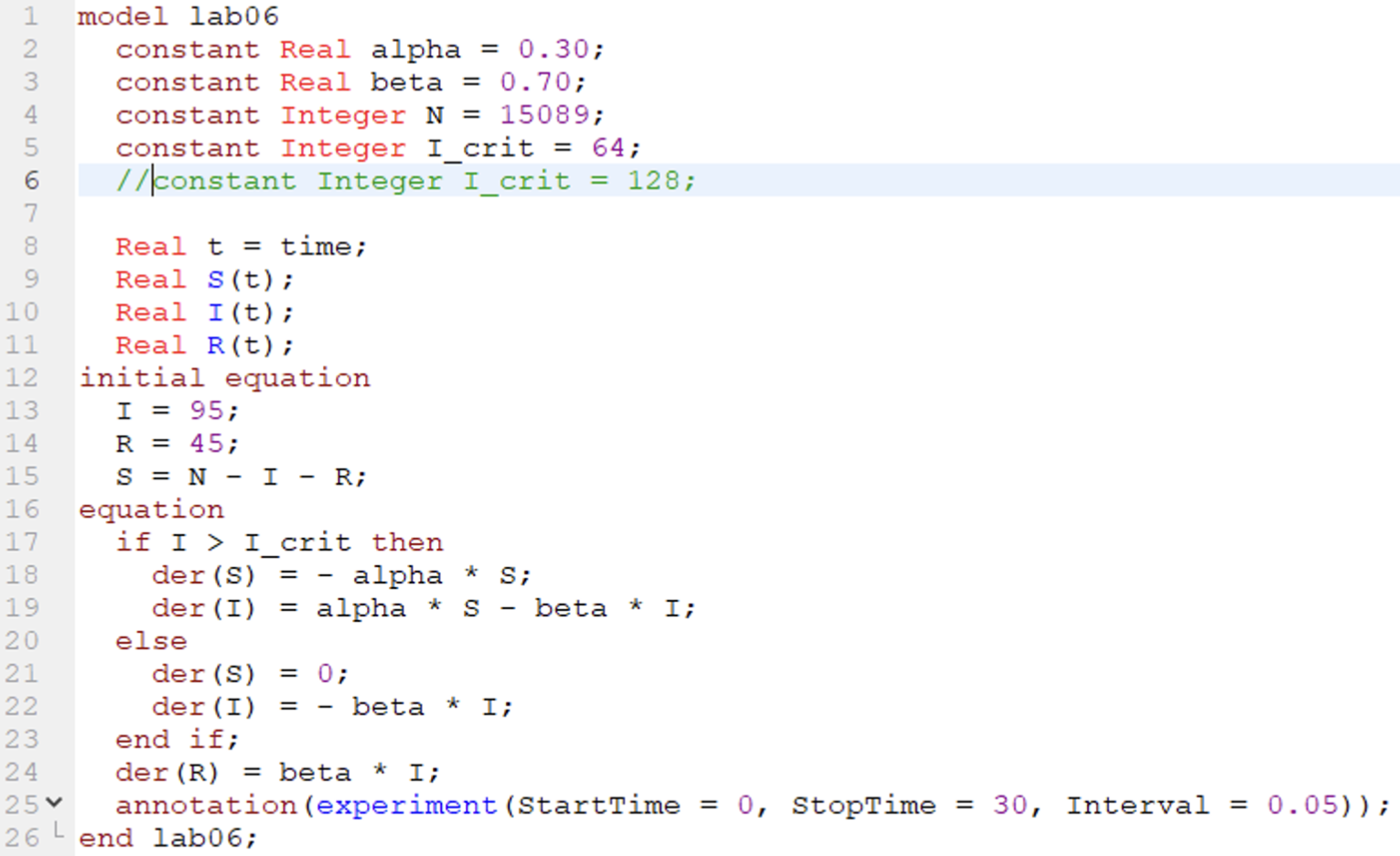
Julia. Скрипт. Задача об эпидемии ()



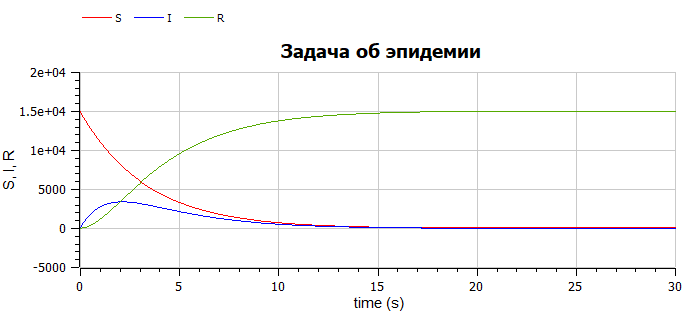
Julia. Модель. Задача об эпидемии ()

1. Постройте графики изменения числа особей в каждой из трех групп на Modelica. Для начала рассмотрим случай, когда . (рис. ??, ??)

model lab06  
 constant Real alpha = 0.30;  
 constant Real beta = 0.70;  
 constant Integer N = 15089;  
 constant Integer I\_crit = 64;  
 Real t = time;  
 Real S(t);  
 Real I(t);  
 Real R(t);  
initial equation  
 I = 95;  
 R = 45;  
 S = N - I - R;  
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 = 0.05));  
end lab06;



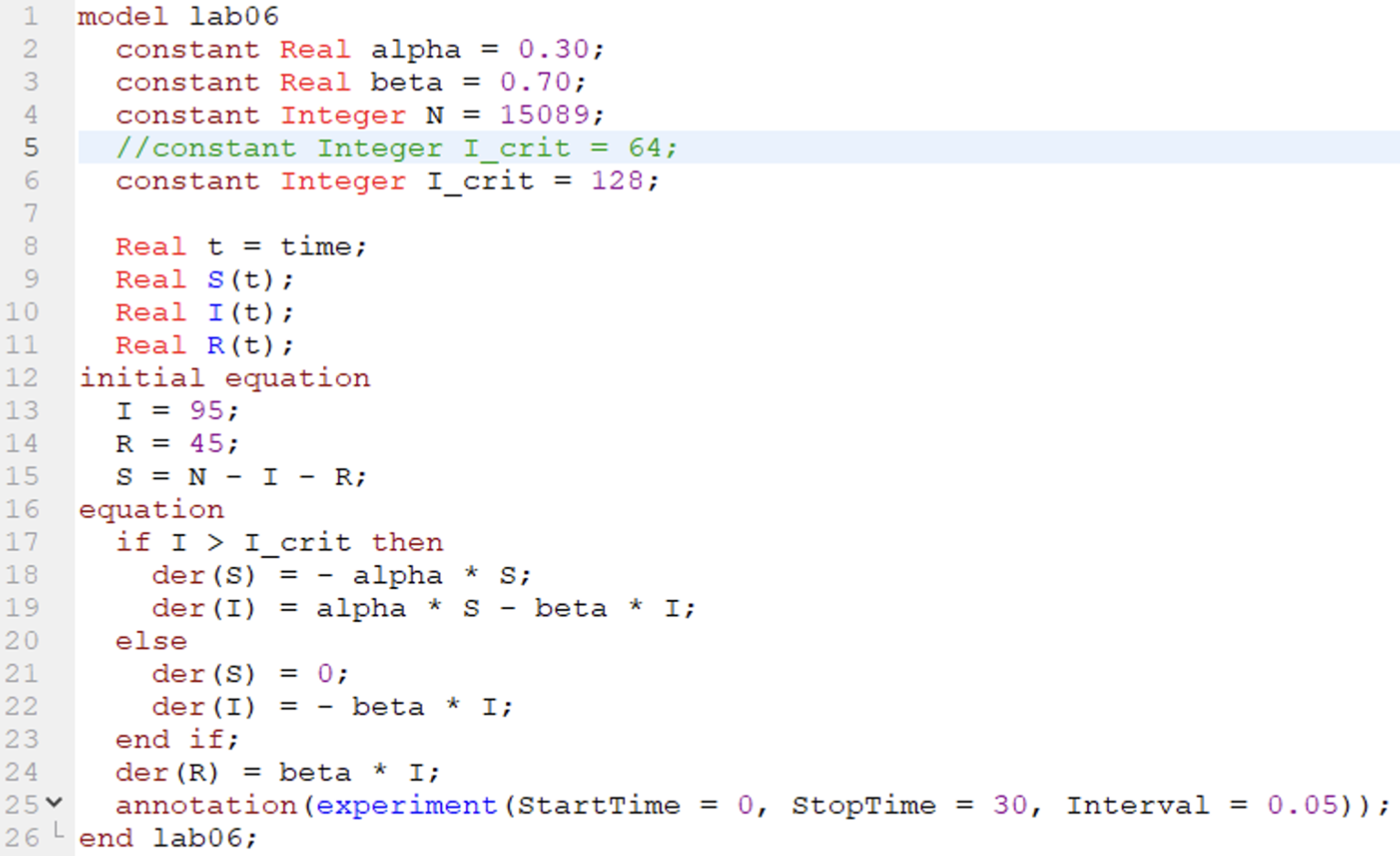
Modelica. Скрипт. Задача об эпидемии ()



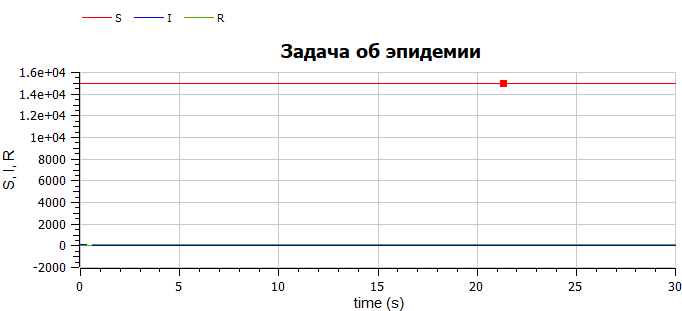
Modelica. Модель. Задача об эпидемии ()

1. Теперь рассмотрим случай, когда . Для этого изменим значение I критического. (рис. ??, ??)

model lab06  
 constant Real alpha = 0.30;  
 constant Real beta = 0.70;  
 constant Integer N = 15089;  
 constant Integer I\_crit = 128;  
 Real t = time;  
 Real S(t);  
 Real I(t);  
 Real R(t);  
initial equation  
 I = 95;  
 R = 45;  
 S = N - I - R;  
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 = 0.05));  
end lab06;



Modelica. Скрипт. Задача об эпидемии ()



Modelica. Модель. Задача об эпидемии ()

# 6 Анализ результатов

Работа выполненна без непредвиденных проблем в соответствии с руководством. Ошибок и сбоев не произошло.

Моделирование на OMEdit было проще и быстрее, чем при использовании средств Julia. Скрипт на Modelica вышел более лакончиным, понятным и коротким. Более того OpenModelica быстрее обрабатывала скрипт и симмулировала модель. Стоит отметить, что OpenModelica имеет множество разлиных полезных инструментов для настройки с симмуляцией и работой с ней.

К плюсам Julia можно отнести, что она является языком программирования, который хорошо подходит для математических и технических задач. Отметим, что скрипт на Julia выполняется долго из-за подключения пакетов, каждый раз при его запуске. При использовании Pluto, нет необходимости каждый раз с нуля выполнять скрипт, таким образом скорость выполнения может даже превышать скорость моделирования в OMEdit.

# 7 Выводы

Мы улучшили практические навыки в области дифференциальных уравнений, улучшили навыки моделирования на Julia, а также навыки моделирования на OpenModelica. Изучили видоизмененную модель заражения SIR и решили при ее помощи задачу об эпидемии.

# Список литературы

1. Julia [Электронный ресурс]. URL: <http://www.unn.ru/books/met_files/JULIA_tutorial.pdf>.

2. OpenModelica [Электронный ресурс]. URL: <https://ru.wikipedia.org/wiki/OpenModelica>.

3. Модель хищник-жертва [Электронный ресурс]. RUDN. URL: <https://esystem.rudn.ru/mod/resource/view.php?id=967249>.

4. Pluto [Электронный ресурс]. URL: <https://plutojl.org/>.

5. Plots in Julia [Электронный ресурс]. URL: <https://docs.juliaplots.org/latest/tutorial/>.

6. Differential Equations in Julia [Электронный ресурс]. URL: <https://docs.sciml.ai/DiffEqDocs/stable/getting_started/>.