Отчет по лабораторной работе №7

по дисциплине: Математическое моделирование

Ким Михаил Алексеевич

Содержание

# 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 математическую модель рекламной компании.

# 2 Задание

Постройте график распространения рекламы, математическая модель которой описывается следующим уравнением:

При этом объем аудитории , в начальный момент о товаре знает человек. Для случая 2 определите в какой момент времени скорость распространения рекламы будет иметь максимальное значение.

# 3 Теоретическое введение

## 3.1 Модель рекламной кампании

**Модель рекламной кампании** — математическая модель, описывающая скорость распространения информации о новом товаре какой-либо компании среди потенциальных покупателей. В нашем случае будем считать, что при распространении информации о товаре на покупателя, он сразу же готов купить рекламируемый товар.

Оценка скорости распространения информации о товаре важна при оценки прибыли от будущих продаж товара по сравнению с избытками издержек, потраченных на рекламу. Вначале расходы на рекламу могут превышать прибыль, но по мере увеличения числа продаж увеличивается прибыль. Однако реклама становится бесполезной, когда рынок насыщается.

Математическая модель рекламной кампании описывается следующим ОДУ:

где — число потенциальных клиентов; — число клиентов, информированных о товаре и готовых его купить; — изменение числа клиентов, информированных о товаре и готовых его купить, со временем; — величина, характеризующая интенсивность рекламной компании; — величина, характеризующая интенсивность т.н. «сарафанного радио».

При значительно большем, чем , график зависимости от будет являться экспоненциальным графиком, а математическая модель будет называться моделью Мальтуса.

При значительно меньшем, чем , получим уравнение логистической кривой [1].

## 3.2 Уравнение модели Мальтуса

**Модель Мальтуса** — это математическая модель, разработанная теоретиком демографии Томасом Мальтусом в XVIII веке, для описания изменения численности населения в течение времени. Мальтус утверждал, что население удваивается каждые 25 лет, а производство продовольствия может увеличиться только линейно. Следовательно, рост населения будет приводить к недостатку пищи и, в конечном итоге, к голоду, болезням и смерти, которые уменьшат численность населения до уровня, соответствующего доступным ресурсам [2].

Модель Мальтуса описывает экспоненциальный рост численности населения. Она основана на предположении, что скорость роста численности населения пропорциональна численности населения в данный момент времени. Если обозначить численность населения в момент времени через , то модель Мальтуса можно записать в следующей форме:

где — скорость изменения численности населения со временем, — текущая численность населения, — коэффициент рождаемости.

Решением этого дифференциального уравнения является экспоненциальная функция:

где — численность населения в момент времени , — исходная численность населений, — темп прироста населения («мальтузианский параметр»).

Таким образом, модель Мальтуса описывает экспоненциальный рост численности населения, то есть увеличение численности населения со временем происходит не пропорционально, а с постоянной скоростью, и эта скорость также увеличивается со временем. Однако, следует отметить, что в реальной жизни экспоненциальный рост на неограниченном промежутке времени невозможен, так как имеются ограничения в ресурсах и пространстве, необходимых для поддержания роста численности населения.

Модель Мальтуса имеет множество ограничений: она не учитывает такие факторы, как миграция, изменения в общественной политике и технологическом прогрессе, которые также влияют на изменение численности населения. Несмотря на это, модель Мальтуса остается важным теоретическим инструментом в изучении демографии и популяционных процессов [3].

## 3.3 Уравнение логистической кривой

**Логистическое уравнение** — это S-образная кривая (сигмоидальная кривая), изначально используемая при построении математических моделей, описывающих изменение размера популяции со временем с учетом ограничений, налагаемых окружающей средой. Уравнение было предложено Пьером Ферхюльстем в 1838 году [4].

Логистическое уравнение имеет следующий вид:

где — размер популяции в момент времени , — скорость роста популяции (без учета ограничений), — предельная вместимость среды.

Первое слагаемое в скобках описывает скорость роста популяции, а второе - ограничивает этот рост учитывая, что на определенном уровне популяции возможности среды ограничивают скорость дальнейшего роста [5].

Важно отметить, что при малых значениях , то есть когда популяция еще не насытила среду, рост популяции описывается экспоненциальной моделью (без второго слагаемого в скобках). Однако при увеличении размера популяции, ограничения среды начинают влиять на скорость роста, и популяция переходит на устойчивое состояние - точку равновесия, которая соответствует величине .

Логистическое уравнение находит применение в различных областях, таких как экология, демография, экономика, теория управления и другие [6].

# 4 Выполнение лабораторной работы

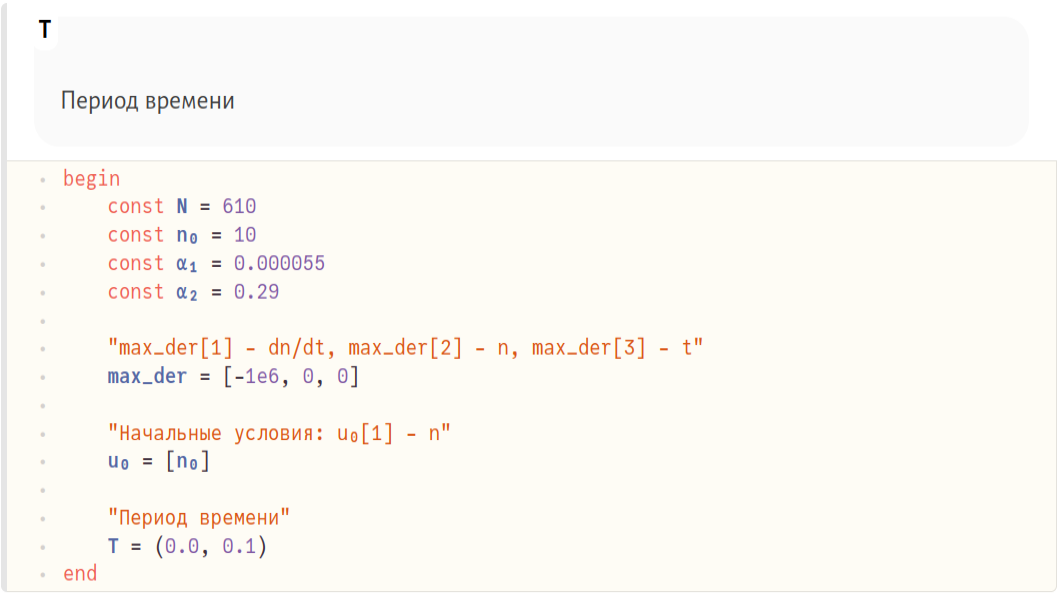
## 4.1 Pluto.jl

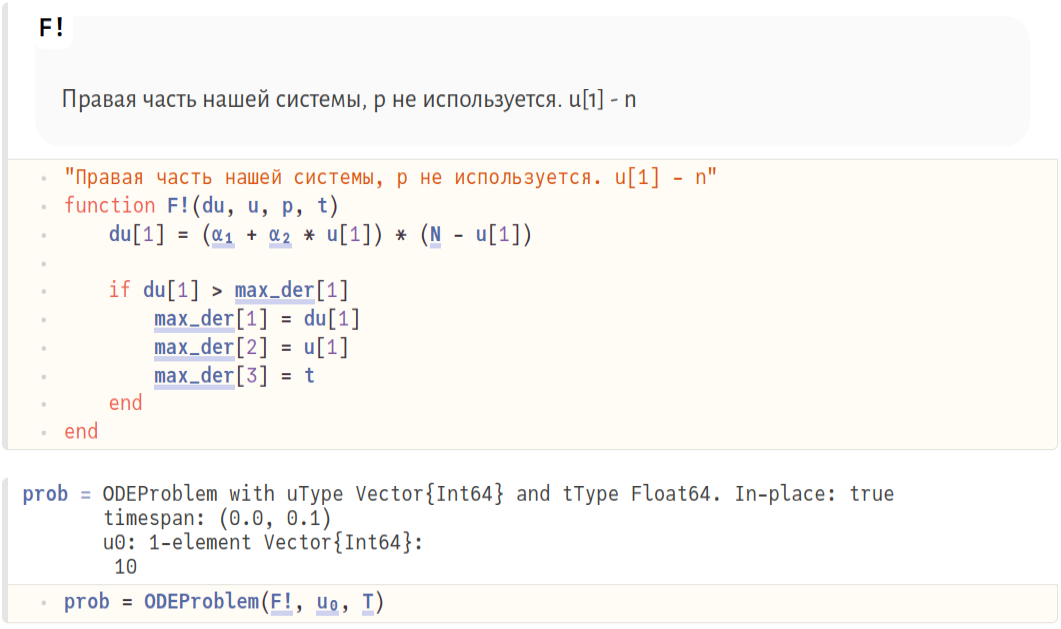
### 4.1.1 Задание №1

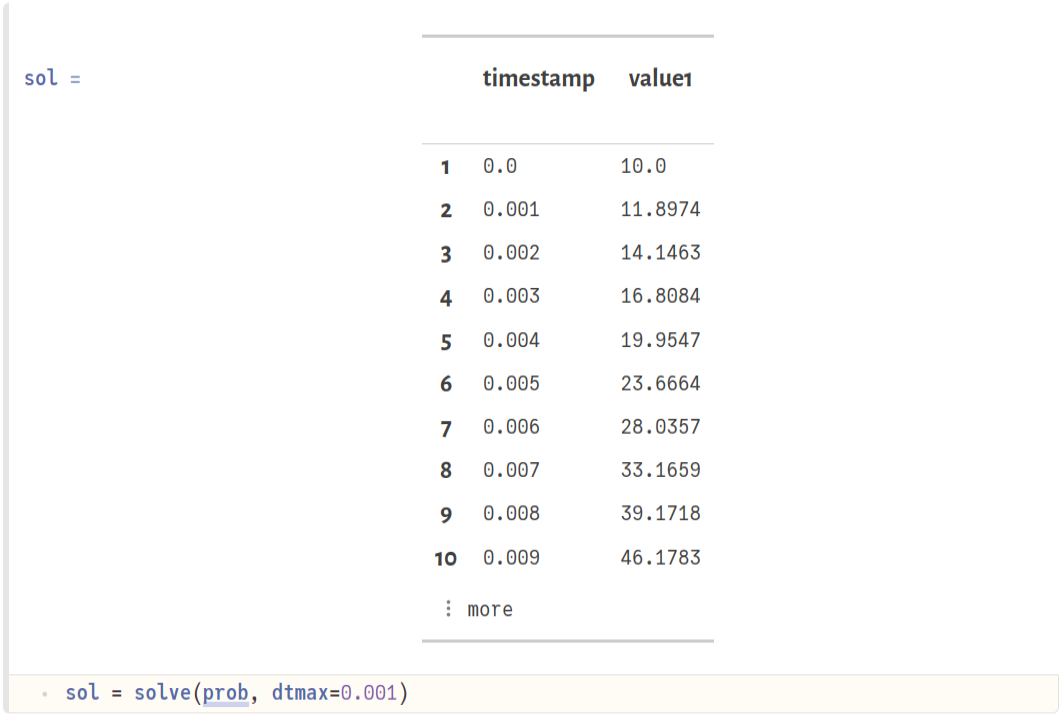
1. Пишем программу, воспроизводящую модель на языке программирования Julia с использованием интерактивного блокнота Pluto (рис. [1](#fig:01), [2](#fig:02), [3](#fig:03), [4](#fig:04), [5](#fig:05), [6](#fig:06)).

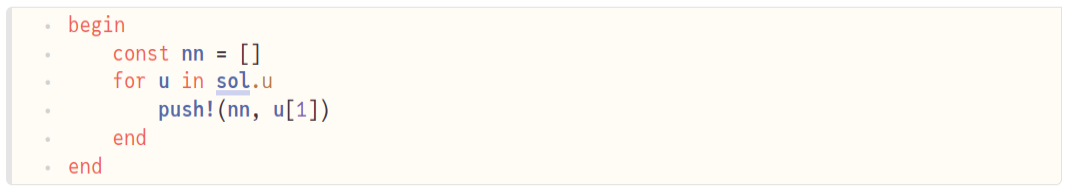
* begin  
   import Pkg  
   Pkg.activate()  
   using DifferentialEquations  
   using LaTeXStrings  
   import Plots  
  end
* begin  
   const N = 610  
   const n₀ = 10  
   const α₁ = 0.000055  
   const α₂ = 0.29  
    
   "max\_der[1] - dn/dt, max\_der[2] - n, max\_der[3] - t"  
   max\_der = [-1e6, 0, 0]  
    
   "Начальные условия: u₀[1] - n"  
   u₀ = [n₀]  
    
   "Период времени"  
   T = (0.0, 0.1)  
  end
* "Правая часть нашей системы, p не используется. u[1] - n"  
  function F!(du, u, p, t)  
   du[1] = (α₁ + α₂ \* u[1]) \* (N - u[1])  
    
   if du[1] > max\_der[1]  
   max\_der[1] = du[1]  
   max\_der[2] = u[1]  
   max\_der[3] = t  
   end  
  end
* prob = ODEProblem(F!, u₀, T)
* sol = solve(prob, dtmax=0.001)
* begin  
   const nn = []  
   for u in sol.u  
   push!(nn, u[1])  
   end  
  end
* begin  
   fig = Plots.plot(  
   dpi=150,  
   grid=:xy,  
   gridcolor=:black,  
   gridwidth=1,  
   size=(800, 400),  
   legend=:outerbottom,  
   xlabel="t",  
   ylabel="n(t)",  
   plot\_title="Эффективность рекламы")  
    
   Plots.plot!(fig[1], sol.t, nn, color=:blue, label="n(t) — число потребителей, узнавших о товаре и готовых его купить")  
   Plots.vline!(fig[1], [max\_der[3]], color=:grey, label="")  
   Plots.scatter!(fig[1], [max\_der[3]], [max\_der[2]], color=:grey, markerstrokewidth=0.2, markersize=4, label="t=" \* string(max\_der[3]) \* " — время, в момент которого скорость распространения рекламы максимальна")  
  end

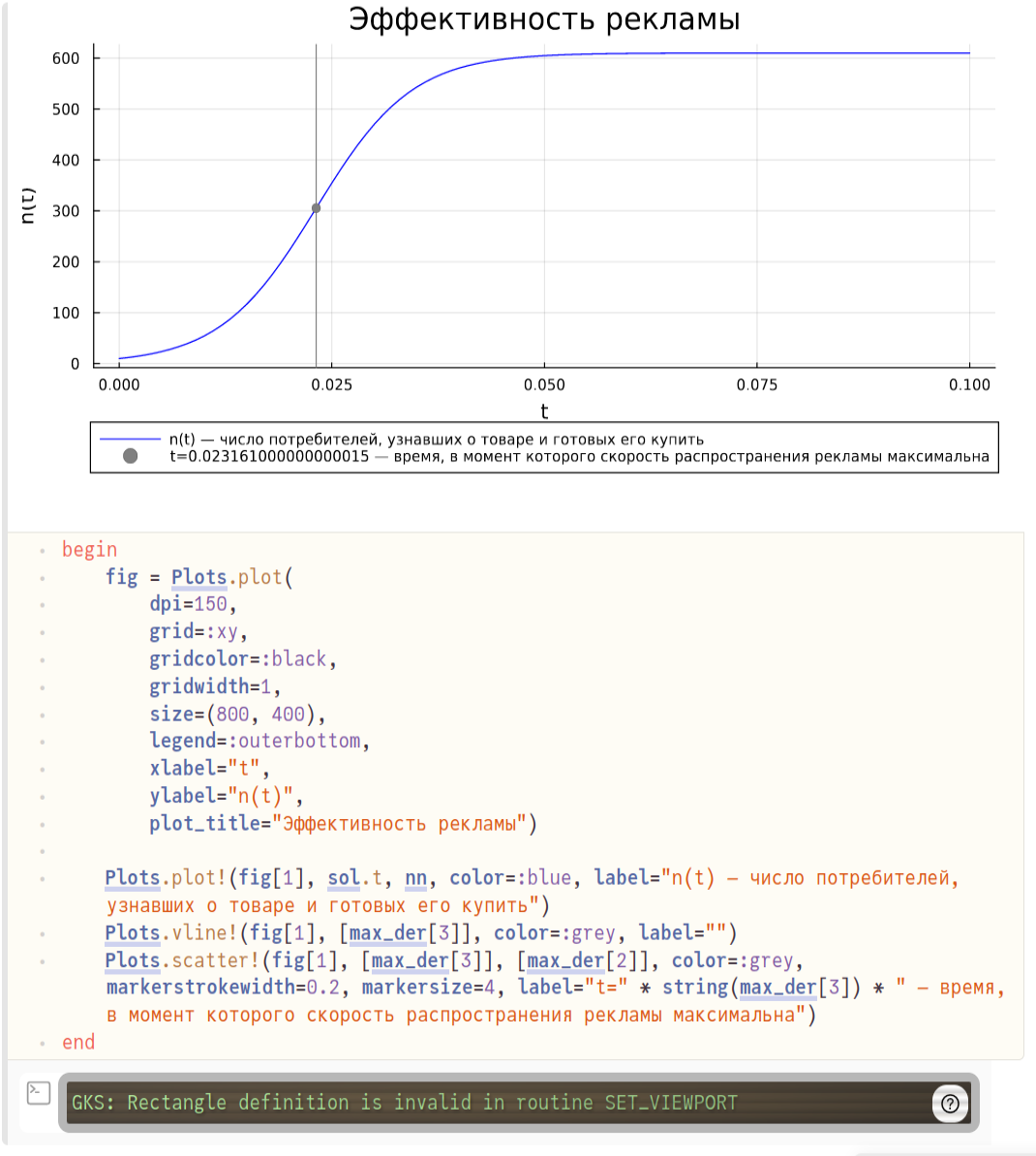
* 
* Figure 1: Импорт библиотек

* 
* Figure 2: Задание коэффициентов; списка, содержащего , , для нахождения и отрисовки точки с максимальной производной; начальных условий; периода времени

* 
* Figure 3: Запись уравнения в виде функции, а также стандартного поиска максимума среди значений производной с записью необходимых данных в список. Постановка проблемы

* 
* Figure 4: Решение задачи (максимальное равно )

* 
* Figure 5: Формирование массива, содержащего значения функции

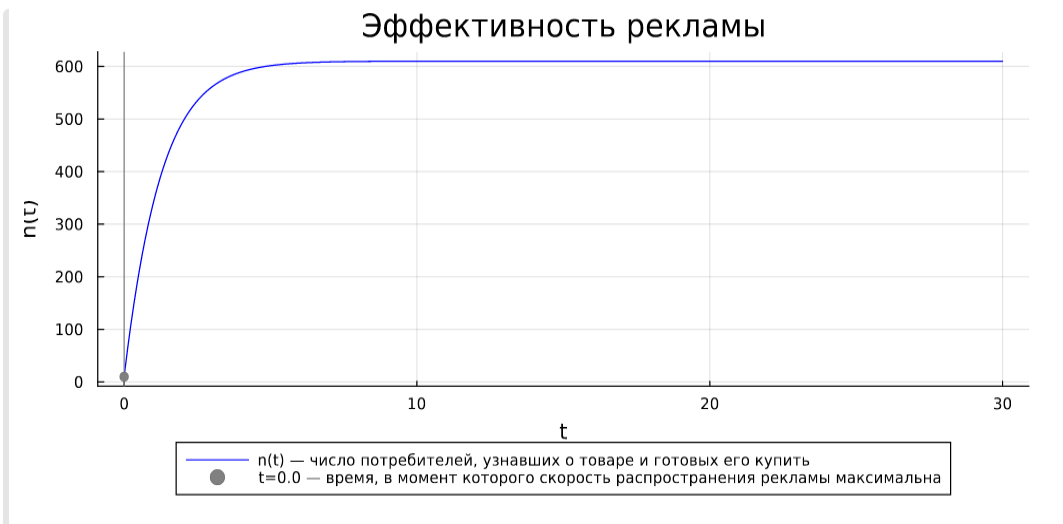
* 
* Figure 6: Отрисовка графика

### 4.1.2 Задание №2

1. Изменены значения коэффициентов и , а также период времени T. Остальные блоки кода оставляем без изменений. Любуемся результатом (рис. [7](#fig:07), [8](#fig:08)).

* begin  
   const N = 610  
   const n₀ = 10  
   const α₁ = 0.77  
   const α₂ = 0.00017  
    
   "max\_der[1] - dn/dt, max\_der[2] - n, max\_der[3] - t"  
   max\_der = [-1e6, 0, 0]  
    
   "Начальные условия: u₀[1] - n"  
   u₀ = [n₀]  
    
   "Период времени"  
   T = (0.0, 30.0)  
  end

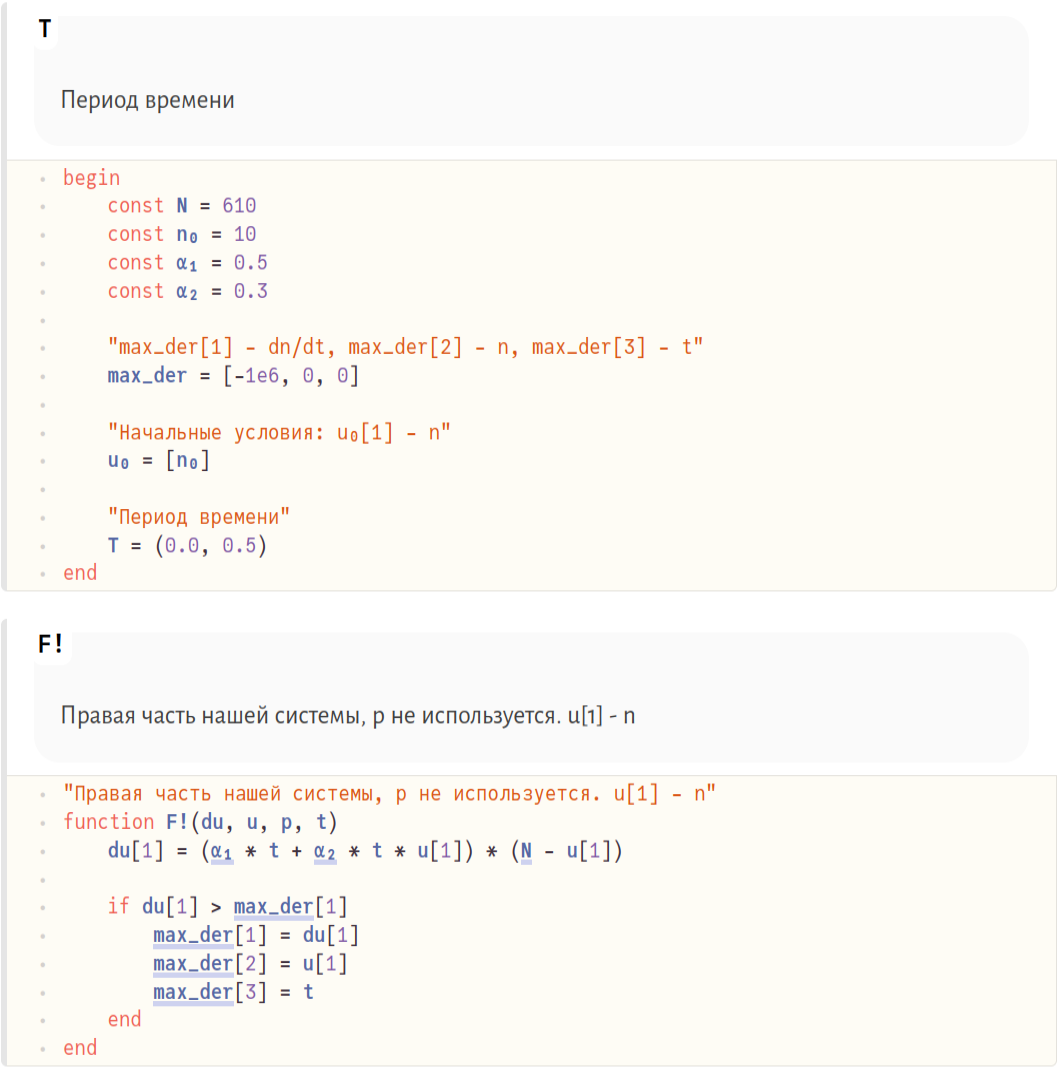
* 
* Figure 7: Изменение значений , , периода времени T

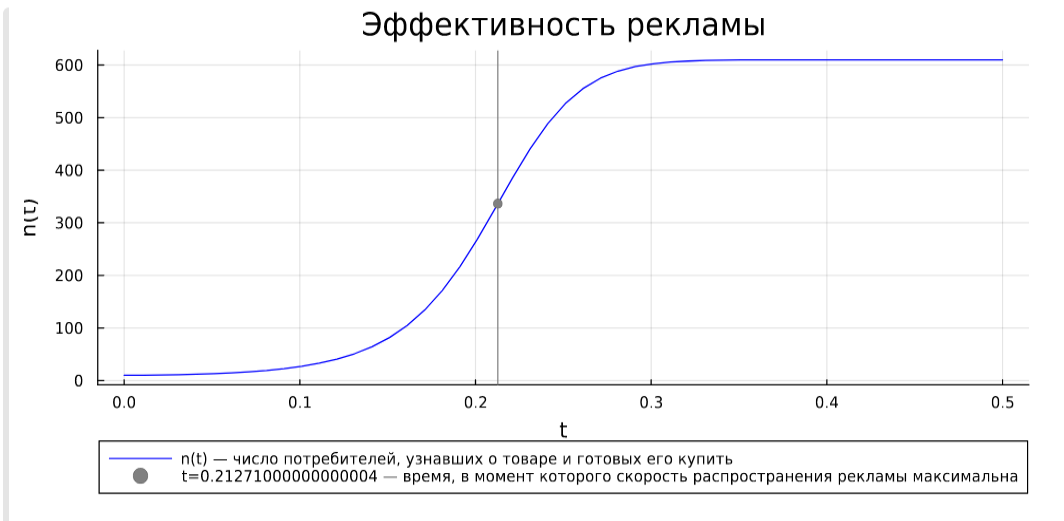
* 
* Figure 8: Результат в виде графика

### 4.1.3 Задание №3

1. Изменены значения коэффициентов и , период времени T и ОДУ в функции F!. Остальные блоки кода оставляем без изменений. Любуемся результатом (рис. [9](#fig:09), [10](#fig:10)).

* begin  
   const N = 610  
   const n₀ = 10  
   const α₁ = 0.5  
   const α₂ = 0.3  
    
   "max\_der[1] - dn/dt, max\_der[2] - n, max\_der[3] - t"  
   max\_der = [-1e6, 0, 0]  
    
   "Начальные условия: u₀[1] - n"  
   u₀ = [n₀]  
    
   "Период времени"  
   T = (0.0, 0.5)  
  end
* ```Julia
* “Правая часть нашей системы, p не используется. u[1] - n” function F!(du, u, p, t) du[1] = (α₁ \* t + α₂ \* t \* u[1]) \* (N - u[1])
* if du[1] > max\_der[1]  
   max\_der[1] = du[1]  
   max\_der[2] = u[1]  
   max\_der[3] = t  
   end
* end ```

* 
* Figure 9: Изменение значений , , периода времени T, функции F!

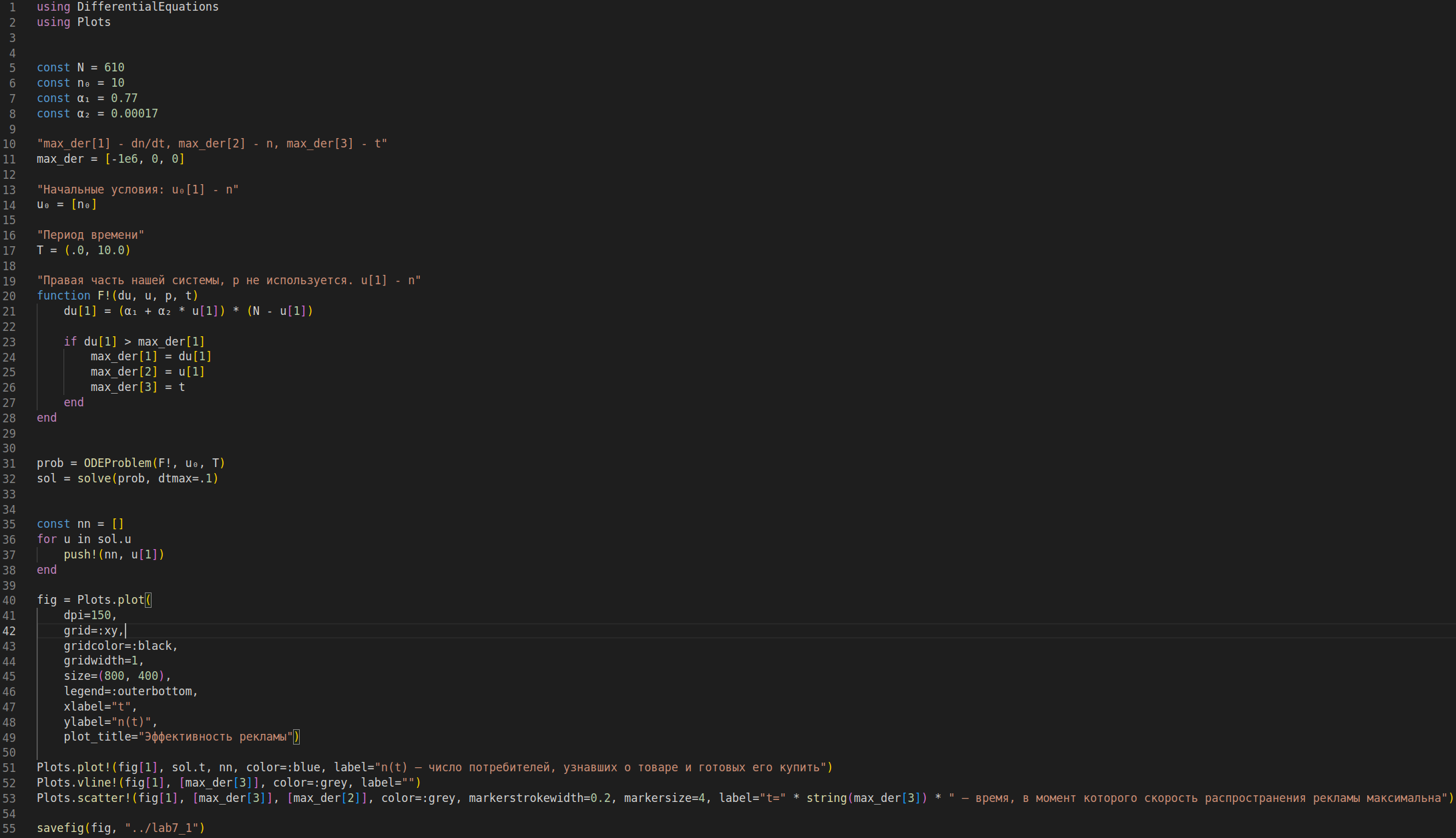
* 
* Figure 10: Результат в виде графика

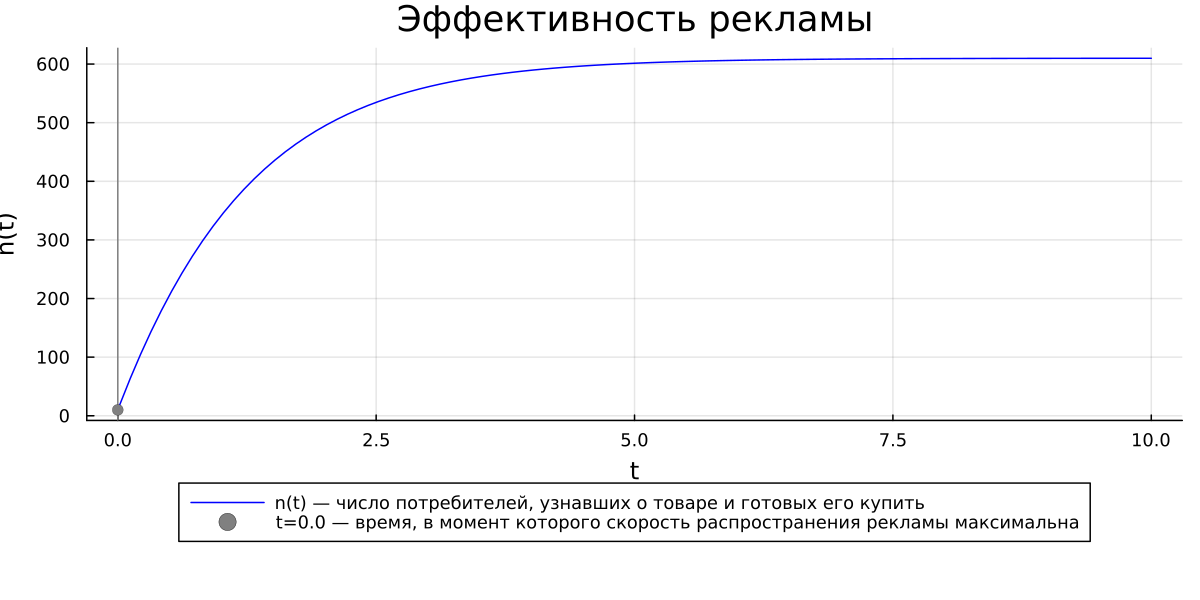
## 4.2 Julia

### 4.2.1 Задание №1

1. Код на Julia в файле аналогичен тому же, что написан с использованием Pluto (рис. [11](#fig:11), [12](#fig:001)). 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 N = 610  
  const n₀ = 10  
  const α₁ = 0.77  
  const α₂ = 0.00017  
    
  "max\_der[1] - dn/dt, max\_der[2] - n, max\_der[3] - t"  
  max\_der = [-1e6, 0, 0]  
    
  "Начальные условия: u₀[1] - n"  
  u₀ = [n₀]  
    
  "Период времени"  
  T = (.0, 10.0)  
    
  "Правая часть нашей системы, p не используется. u[1] - n"  
  function F!(du, u, p, t)  
   du[1] = (α₁ + α₂ \* u[1]) \* (N - u[1])  
    
   if du[1] > max\_der[1]  
   max\_der[1] = du[1]  
   max\_der[2] = u[1]  
   max\_der[3] = t  
   end  
  end  
    
    
  prob = ODEProblem(F!, u₀, T)  
  sol = solve(prob, dtmax=.1)  
    
    
  const nn = []  
  for u in sol.u  
   push!(nn, u[1])  
  end  
    
  fig = Plots.plot(  
   dpi=150,  
   grid=:xy,  
   gridcolor=:black,  
   gridwidth=1,  
   size=(800, 400),  
   legend=:outerbottom,  
   xlabel="t",  
   ylabel="n(t)",  
   plot\_title="Эффективность рекламы")  
    
  Plots.plot!(fig[1], sol.t, nn, color=:blue, label="n(t) — число потребителей, узнавших о товаре и готовых его купить")  
  Plots.vline!(fig[1], [max\_der[3]], color=:grey, label="")  
  Plots.scatter!(fig[1], [max\_der[3]], [max\_der[2]], color=:grey, markerstrokewidth=0. 2, markersize=4, label="t=" \* string(max\_der[3]) \* " — время, в момент которого скорость распространения рекламы максимальна")  
    
  savefig(fig, "../lab7\_1")

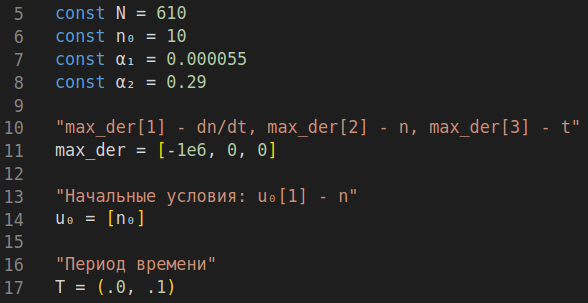
* 
* Figure 11: Код программы на Julia. Аналогичен коду задания для Pluto.jl

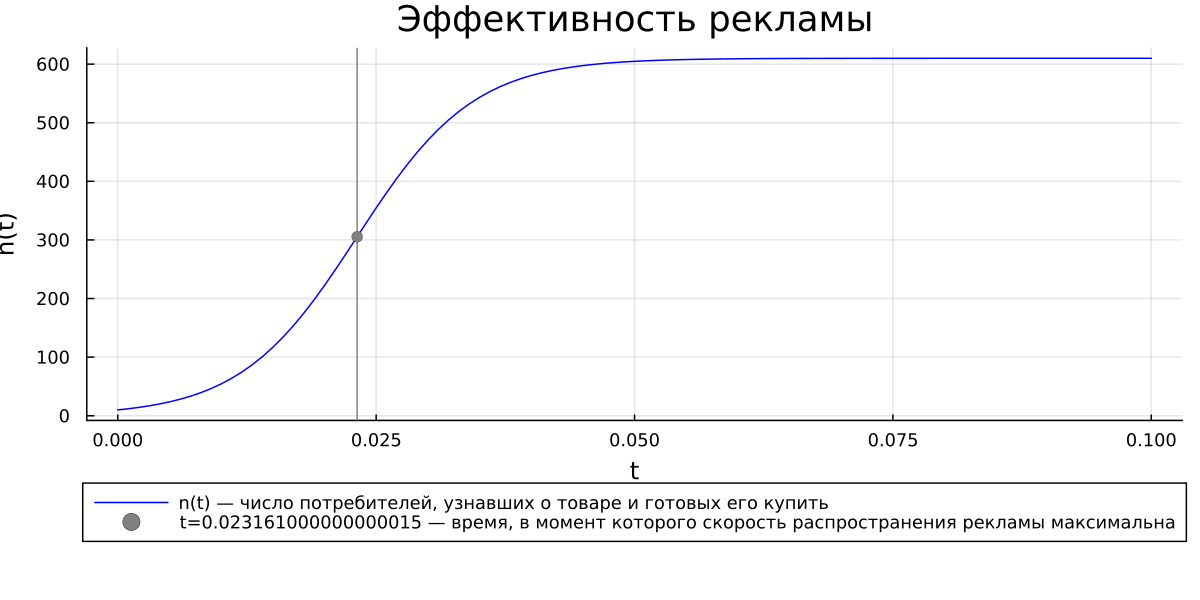
* 
* Figure 12: Результат в виде графика

### 4.2.2 Задание №2

1. Изменяем значения коэффициентов и , а также период времени T (подробное объяснение давалось в предыдущей главе) (рис. [13](#fig:12), [14](#fig:002)).

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

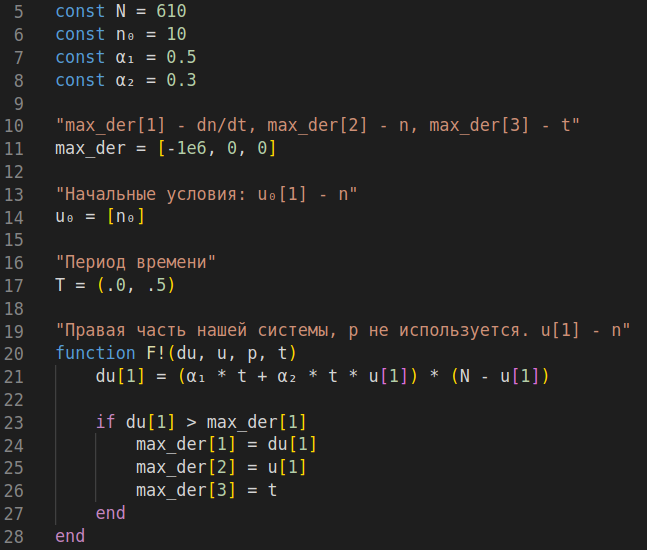
* 
* Figure 13: Измененная часть кода

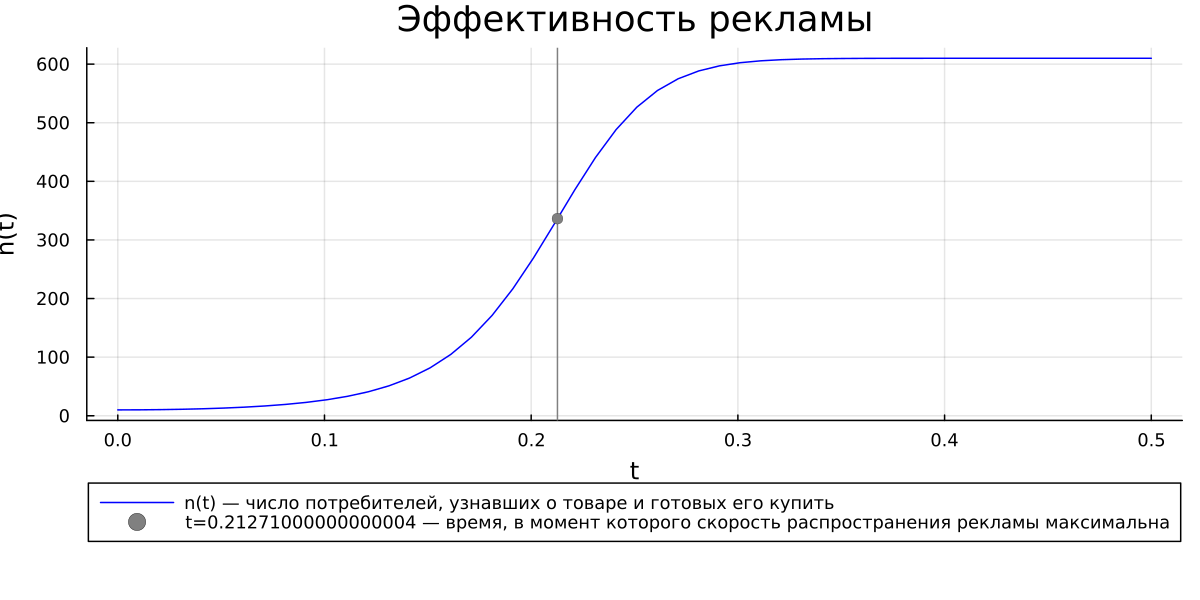
* 
* Figure 14: Результат в виде графиков

### 4.2.3 Задание №3

1. Изменяем значения коэффициентов и , период времени T и ОДУ в функции F! (подробное объяснение давалось в предыдущей главе) (рис. [15](#fig:13), [16](#fig:003)).

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

* 
* Figure 15: Измененная часть кода

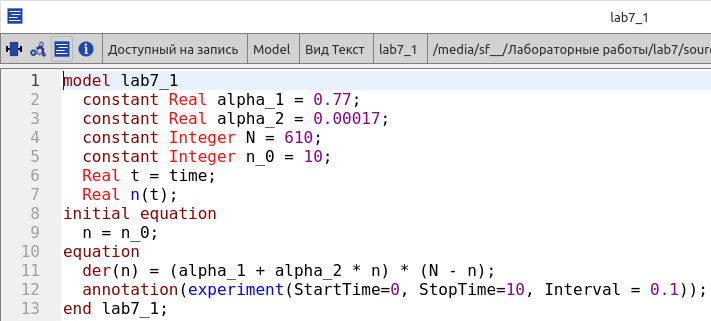
* 
* Figure 16: Результат в виде графиков

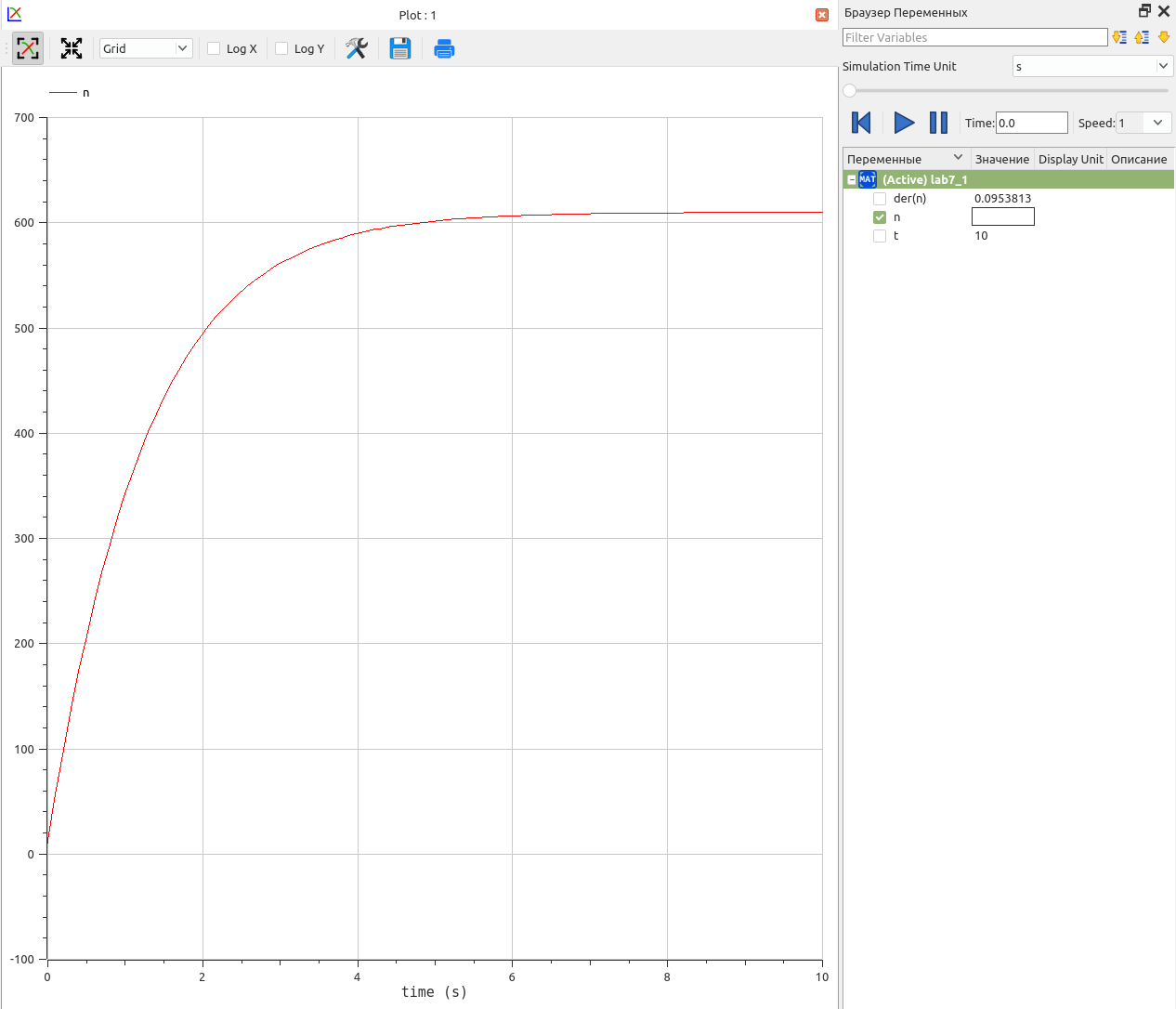
## 4.3 Modelica

### 4.3.1 Задание №1

1. По аналогии с Julia пишем программу, воспроизводящую модель рекламной кампании на языке моделирования Modelica с использованием ПО OpenModelica. Любуемся результатами (рис. [17](#fig:14), [18](#fig:15)).

* model lab7\_1  
   constant Real alpha\_1 = 0.77;  
   constant Real alpha\_2 = 0.00017;  
   constant Integer N = 610;  
   constant Integer n\_0 = 10;  
   Real t = time;  
   Real n(t);  
  initial equation  
   n = n\_0;  
  equation  
   der(n) = (alpha\_1 + alpha\_2 \* n) \* (N - n);  
   annotation(experiment(StartTime=0, StopTime=10, Interval = 0.1));  
  end lab7\_1;

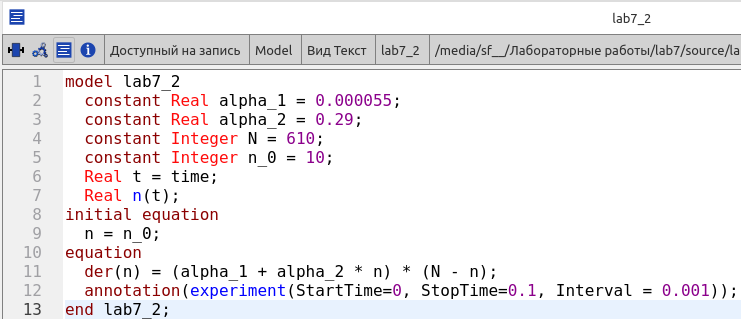
* 
* Figure 17: Определяем коэффициенты, функцию от времени, ОДУ, а также начальное/конечное время и частоту разбиения при симуляции

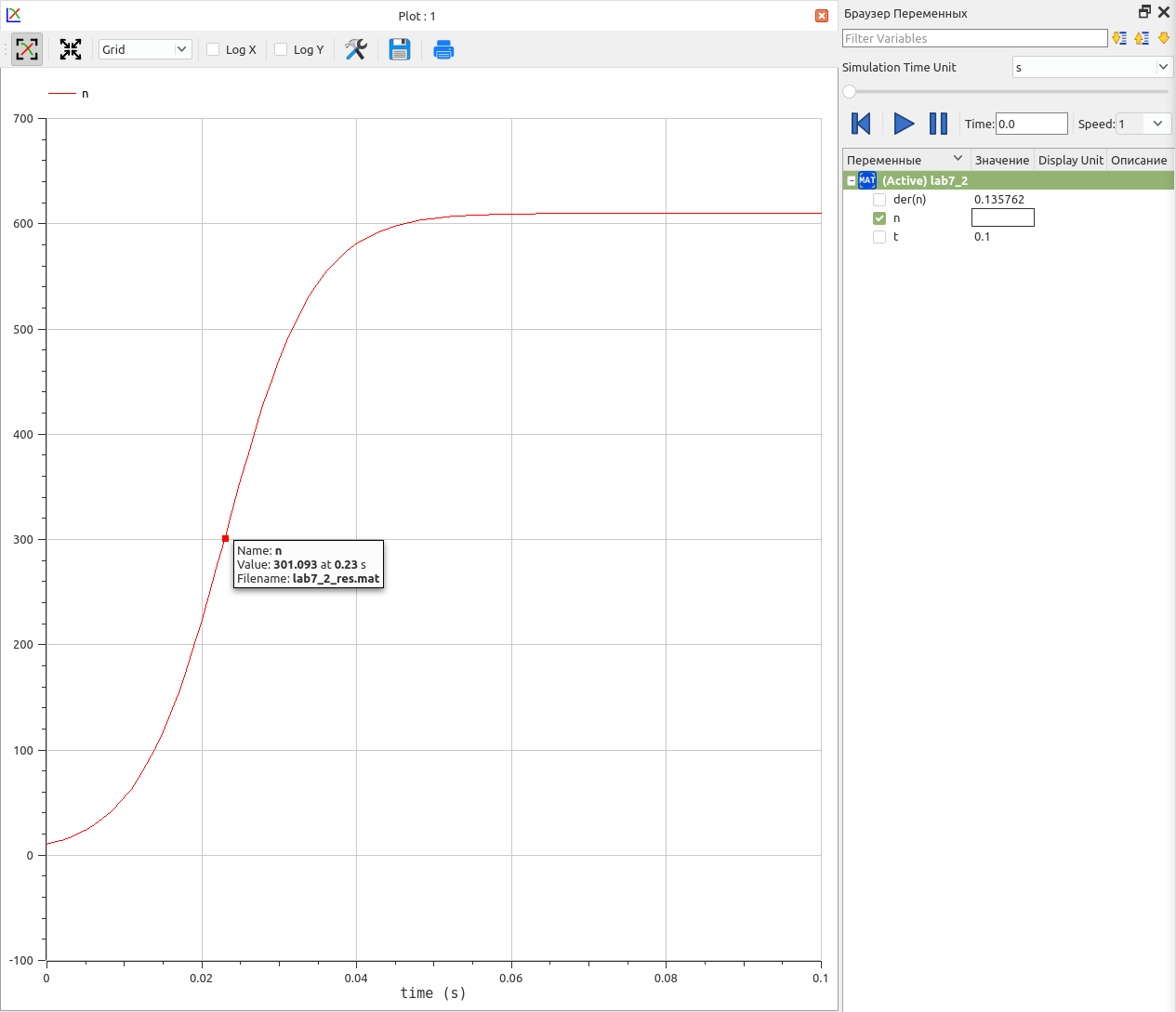
* 
* Figure 18: Результат в виде графика зависимости от

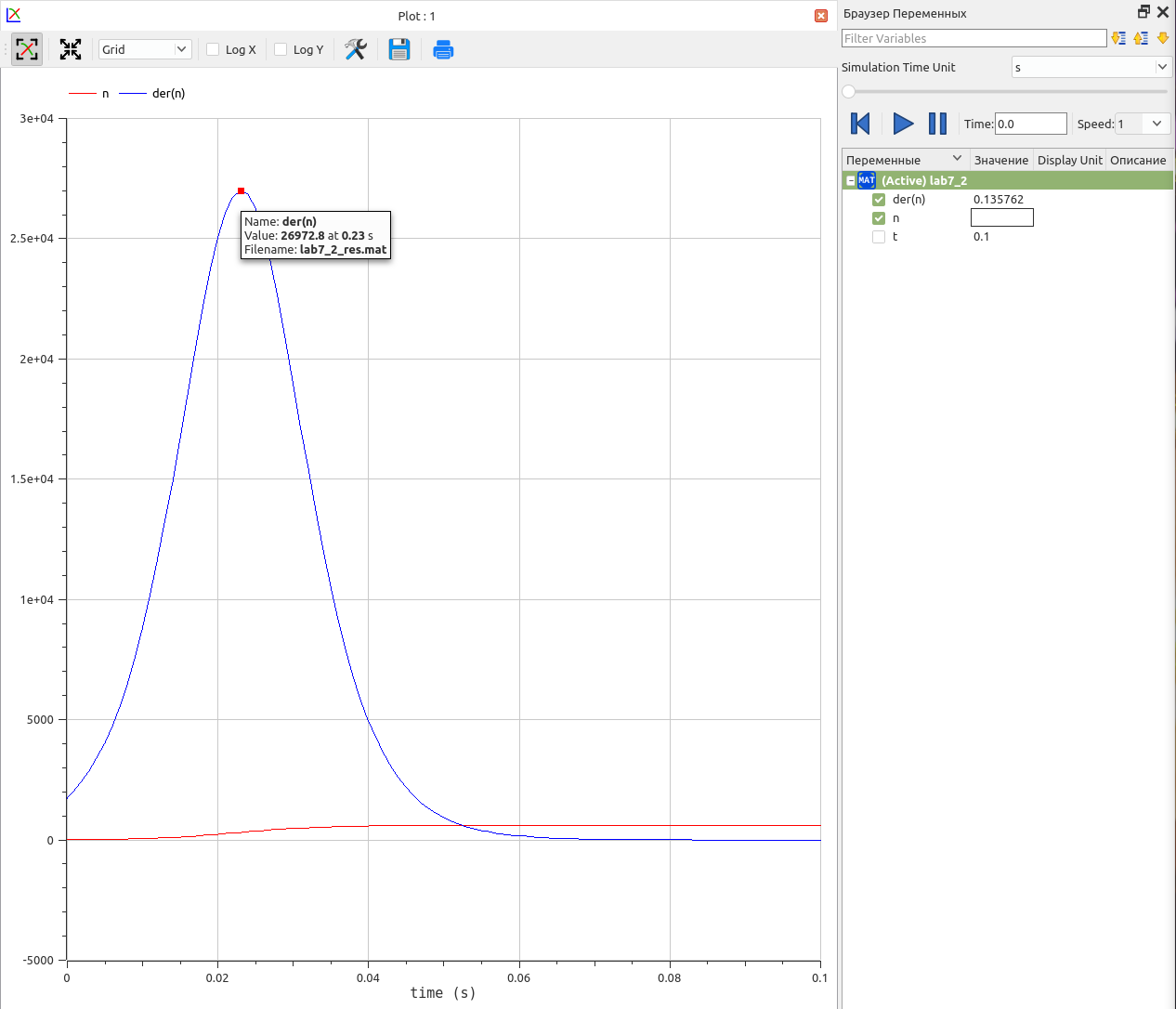
### 4.3.2 Задание №2

1. По аналогии с Julia пишем программу для второго случая. Любуемся результатами (рис. [19](#fig:16), [20](#fig:17), [21](#fig:18)).

* model lab7\_2  
   constant Real alpha\_1 = 0.000055;  
   constant Real alpha\_2 = 0.29;  
   constant Integer N = 610;  
   constant Integer n\_0 = 10;  
   Real t = time;  
   Real n(t);  
  initial equation  
   n = n\_0;  
  equation  
   der(n) = (alpha\_1 + alpha\_2 \* n) \* (N - n);  
   annotation(experiment(StartTime=0, StopTime=0.1, Interval = 0.001));  
  end lab7\_2;

* 
* Figure 19: По сравнению с предыдущим случаем измянются значение коэффициентов alpha\_1, alpha\_2, период времени и частота разбиения

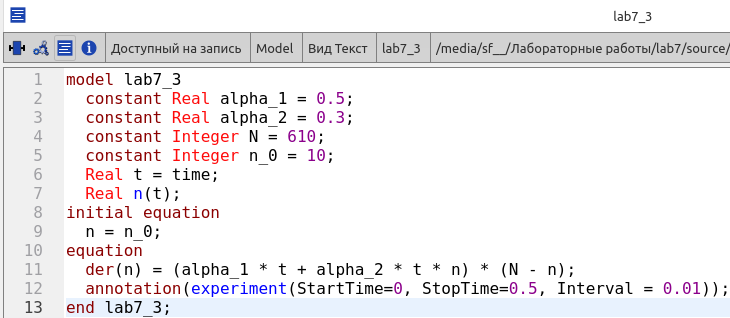
* 
* Figure 20: Результат в виде графика зависимости от

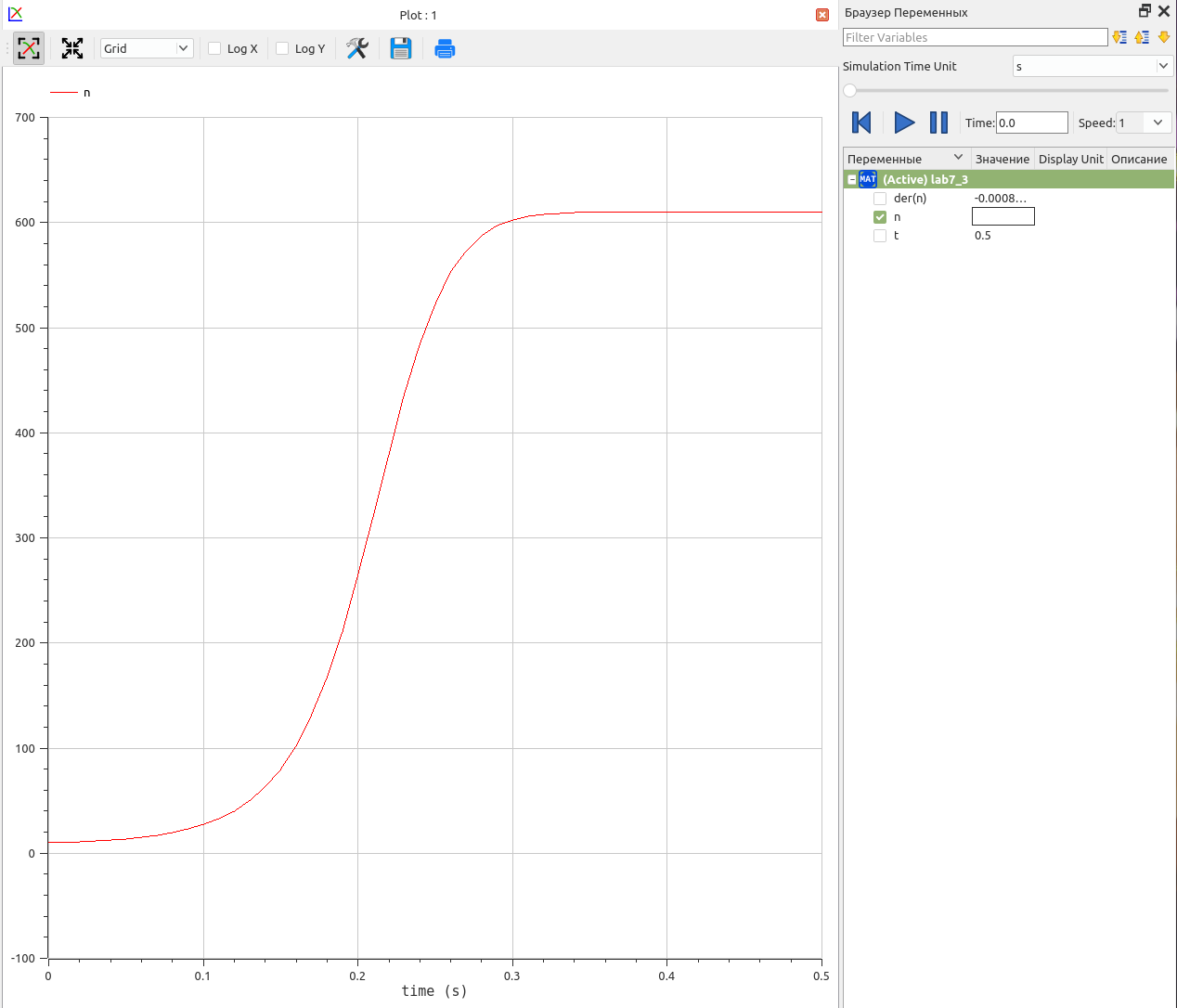
* 
* Figure 21: Максимальное значение производной достигается в момент времени

### 4.3.3 Задание №3

1. По аналогии с Julia пишем программу для третьего случая. Любуемся результатами (рис. [22](#fig:19), [23](#fig:20)).

* model lab7\_3  
   constant Real alpha\_1 = 0.5;  
   constant Real alpha\_2 = 0.3;  
   constant Integer N = 610;  
   constant Integer n\_0 = 10;  
   Real t = time;  
   Real n(t);  
  initial equation  
   n = n\_0;  
  equation  
   der(n) = (alpha\_1 \* t + alpha\_2 \* t \* n) \* (N - n);  
   annotation(experiment(StartTime=0, StopTime=0.5, Interval = 0.01));  
  end lab7\_3;

* 
* Figure 22: По сравнению с предыдущим случаем измянются значение коэффициентов alpha\_1, alpha\_2, ОДУ, период времени и частота разбиения

* 
* Figure 23: Результат в виде графика зависимости от

# 5 Анализ результатов

На текущем примере построения математической модель рекламной кампании мы можем продолжить сравнивать язык программирования Julia и язык моделирования Modelica. Если быть откровенным, по сравнению с анализом результатов при выполнении предыдущей лабораторной работы изменения незначительны: тенденция к сглаживанию негативных моментов при выполнении лабораторной работы на языке программирования Julia по сравнению с языком моделирования Modelica 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 языке Julia можно явно найти момент времени, во время которого скорость изменения функции (т.е. ) максимальна, т.к. мы можем напрямую взаимодействовать со значениями производной в каждый момент времени, обусловленный шагом разбиения. Это позволяет достаточно легко находить максимальное значение производной на периоде, момент времени в этой точки, а также само значение функции в этой точке.

При написании же программы на Modelica приходится вручную искать максимальное значение по графику производной функции .

С другой стороны, хотелось бы отметить, 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, что может привести к ошибкам, связанными с усидчивостью, при описании модели.

# 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 рекламной кампании.

# Список литературы

1. Эффективность рекламы [Электронный ресурс]. RUDN, 2023. URL: <https://esystem.rudn.ru/mod/resource/view.php?id=967253>.

2. Who Is Thomas Malthus? [Электронный ресурс]. Investopedia, 2023. URL: <https://www.investopedia.com/terms/t/thomas-malthus.asp#:~:text=The%20Malthusian%20growth%20model%20is,population%20doubles%20at%20predictable%20intervals.>

3. Malthusian growth model [Электронный ресурс]. Wikimedia Foundation, Inc., 2022. URL: <https://en.wikipedia.org/wiki/Malthusian_growth_model>.

4. Logistic function [Электронный ресурс]. Wikimedia Foundation, Inc., 2023. URL: <https://en.wikipedia.org/wiki/Logistic_function>.

5. The Logistic Equation [Электронный ресурс]. Northwestern University. URL: <https://sites.math.northwestern.edu/~mlerma/courses/math214-2-03f/notes/c2-logist.pdf>.

6. Exponential & logistic growth [Электронный ресурс]. Khan Academy. URL: <https://www.khanacademy.org/science/ap-biology/ecology-ap/population-ecology-ap/a/exponential-logistic-growth>.