Отчет по лабораторной работе №7

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 математическую модель рекламной компании.

2 Задание

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

1. $\frac{dn}{dt} = (0.77 + 0.00017n(t))(N - n(t))$
2. $\frac{dn}{dt} = (0.000055 + 0.29n(t))(N - n(t))$
3. $\frac{dn}{dt} = (0.5 \cdot t + 0.3 \cdot t \cdot n(t))(N - n(t))$

При этом объем аудитории $N = 610$, в начальный момент о товаре знает 10 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:

$$\frac{dn}{dt} = (\alpha_1(t) + \alpha_2(t)n(t))(N - n(t)),$$

где N — число потенциальных клиентов; $n(t)$ — число клиентов, информированных о товаре и готовых его купить; $\frac{dn}{dt}$ — изменение числа клиентов, информированных о товаре и готовых его купить, со временем; $\alpha_1(t)$ — величина, характеризующая интенсивность рекламной компании; $\alpha_2(t)$ — величина, характеризующая интенсивность т.н. «сарафанного радио».

При $\alpha_1(t)$ значительно большем, чем $\alpha_2(t)$, график зависимости $n(t)$ от t будет являться экспоненциальным графиком, а математическая модель будет называться моделью Мальтуса.

При $\alpha_1(t)$ значительно меньшем, чем $\alpha_2(t)$, 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 t через $P(t)$, то модель Мальтуса можно записать в следующей форме:

$$\frac{dP}{dt} = r \cdot N,$$

где dP/dt — скорость изменения численности населения со временем, P — текущая численность населения, r — коэффициент рождаемости.

Решением этого дифференциального уравнения является экспоненциальная функция:

$$P(t) = P_0 \cdot e^{rt},$$

где $P(t)$ — численность населения в момент времени t, P_0 — исходная численность населений, r — 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:

$$\frac{dN}{dt} = rN(1 - \frac{N}{K})$$

где N — размер популяции в момент времени t, r — скорость роста популяции (без учета ограничений), K — 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 N, 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 K.

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. 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 N = 610
    const n⊠ = 10
    const ⊠₁ = 0.000055
    const ⊠₂ = 0.29

    "max_der[1] - dn/dt, max_der[2] - n, max_der[3] - t"
    max_der = [-1e6, 0, 0]
```

```julia
    "Начальные условия: u☒[1] - n"
    u☒ = [n☒]

    "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] = (☒₁ + ☒₂ * 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!, u☒, 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 потребит
    Plots.vline!(fig[1], [max_der[3]], color=:grey, label="")
    Plots.scatter!(fig[1], [max_der[3]], [max_der[2]], color=:grey,      mark
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 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
```

Рис. 4.2: Задание коэффициентов; списка, содержащего $\dot{n}, n, t$ для нахождения и отрисовки точки с максимальной производной; начальных условий; периода времени

F!

Правая часть нашей системы, p не используется. u[1] - n

```
"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 with uType Vector{Int64} and tType Float64. In-place: true
       timespan: (0.0, 0.1)
       u0: 1-element Vector{Int64}:
        10
```

```
prob = ODEProblem(F!, u₀, T)
```

Рис. 4.3: Запись уравнения в виде функции, а также стандартного поиска максимума среди значений производной с записью необходимых данных в список. Постановка проблемы sol =

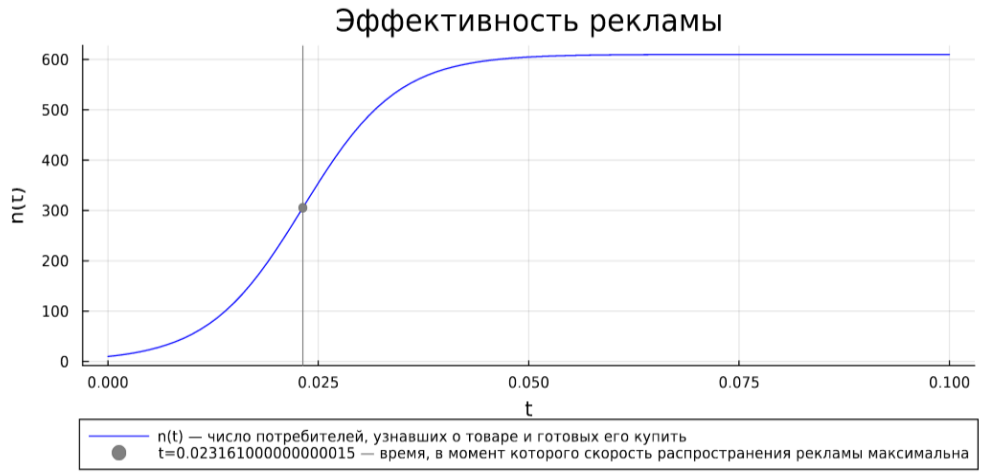
	timestamp	value1
1	0.0	10.0
2	0.001	11.8974
3	0.002	14.1463
4	0.003	16.8084
5	0.004	19.9547
6	0.005	23.6664
7	0.006	28.0357
8	0.007	33.1659
9	0.008	39.1718
10	0.009	46.1783
⋮ more		

```
sol = solve(prob, dtmax=0.001)
```

Рис. 4.4: Решение задачи (максимальное δt равно 0.001)

```
begin
    const nn = []
    for u in sol.u
        push!(nn, u[1])
    end
end
```

Рис. 4.5: Формирование массива, содержащего значения функции $n(t)$




```
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
```

```
GKS: Rectangle definition is invalid in routine SET_VIEWPORT
```

Рис. 4.6: Отрисовка графика

4.1.2 Задание №2

1. Изменены значения коэффициентов α_1 и α_2, а также период времени T. Остальные блоки кода оставляем без изменений. Любуемся результатом (рис. 4.7, 4.8).

```
begin
    const N = 610
    const n⊠ = 10
    const ⊠₁ = 0.77
    const ⊠₂ = 0.00017

    "max_der[1] - dn/dt, max_der[2] - n, max_der[3] - t"
    max_der = [-1e6, 0, 0]

    "Начальные условия: u⊠[1] - n"
    u⊠ = [n⊠]

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

T

Период времени

```
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
```

Рис. 4.7: Изменение значений α_1, α_2, периода времени T

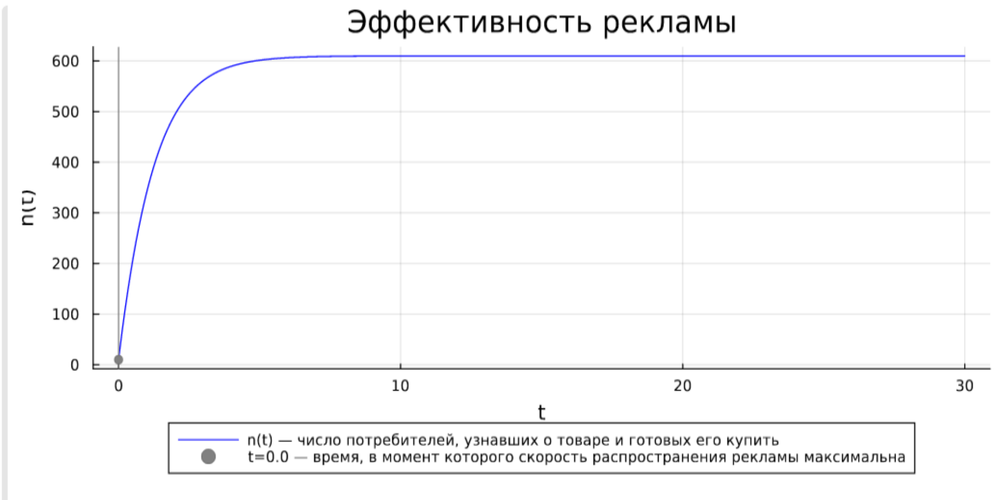


Рис. 4.8: Результат в виде графика

4.1.3 Задание №3

1. Изменены значения коэффициентов α_1 и α_2, период времени `T` и ОДУ в функции `F!`. Остальные блоки кода оставляем без изменений. Любуемся результатом (рис. 4.9, 4.10).

```
begin
    const N = 610
    const n⊠ = 10
    const ⊠₁ = 0.5
    const ⊠₂ = 0.3

    "max_der[1] - dn/dt, max_der[2] - n, max_der[3] - t"
    max_der = [-1e6, 0, 0]

    "Начальные условия: u⊠[1] - n"
    u⊠ = [n⊠]
```

~~~
    "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] = ($\boxtimes_1$ * t + $\boxtimes_2$ * 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 “‘

T

Период времени

```
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
```

F!

Правая часть нашей системы, p не используется. u[1] - n

```
"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
```

Рис. 4.9: Изменение значений α_1, α_2, периода времени T, функции F!

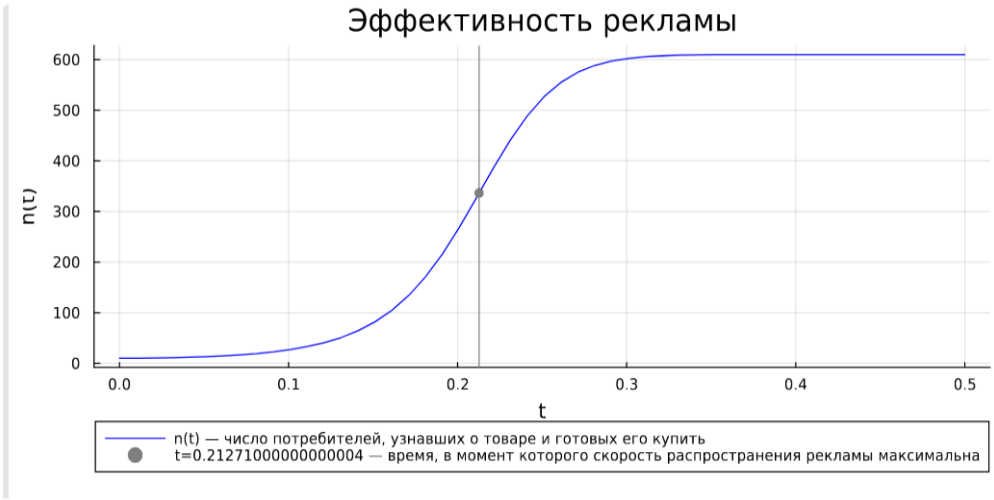


Рис. 4.10: Результат в виде графика

4.2 Julia

4.2.1 Задание №1

1. Код на Julia в файле аналогичен тому же, что написан с использованием Pluto (рис. 4.11, 4.12). 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 n⊠ = 10
```

```
const ⊠₁ = 0.77
const ⊠₂ = 0.00017

"max_der[1] - dn/dt, max_der[2] - n, max_der[3] - t"
max_der = [-1e6, 0, 0]

"Начальные условия: u⊠[1] - n"
u⊠ = [n⊠]

"Период времени"
T = (.0, 10.0)

"Правая часть нашей системы, p не используется. u[1] - n"
function F!(du, u, p, t)
    du[1] = (⊠₁ + ⊠₂ * 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!, u⊠, 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
Plots.vline!(fig[1], [max_der[3]], color=:grey, label="")
Plots.scatter!(fig[1], [max_der[3]], [max_der[2]], color=:grey, markerstroke

savefig(fig, "../lab7_1")
```

```julia
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")
```

Рис. 4.11: Код программы на Julia. Аналогичен коду задания для Pluto.jl

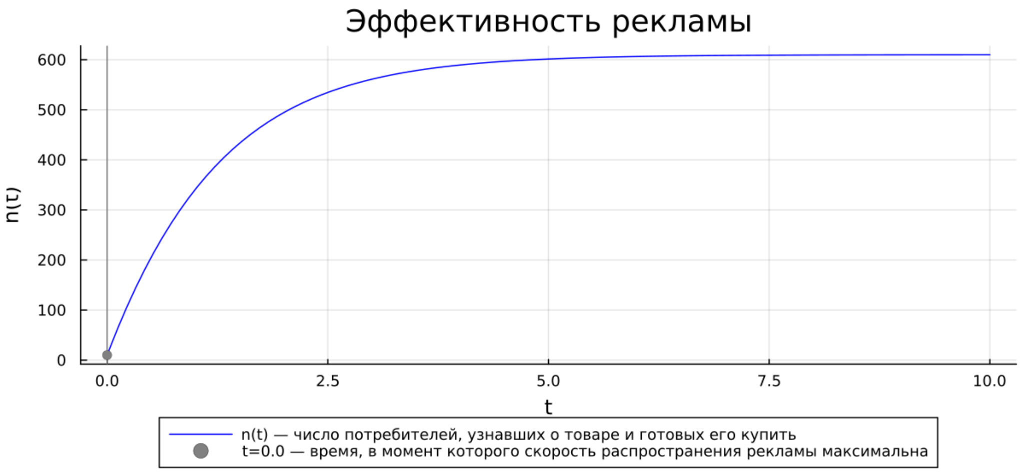


Рис. 4.12: Результат в виде графика

4.2.2 Задание №2

1. Изменяем значения коэффициентов α_1 и α_2, а также период времени T (подробное объяснение давалось в предыдущей главе) (рис. 4.13, 4.14).

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

```
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, .1)
```

Рис. 4.13: Измененная часть кода

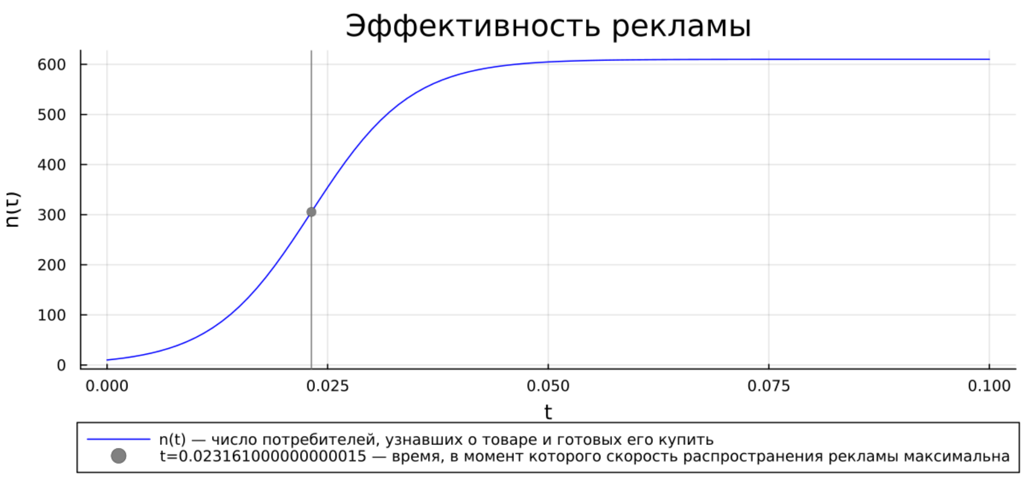
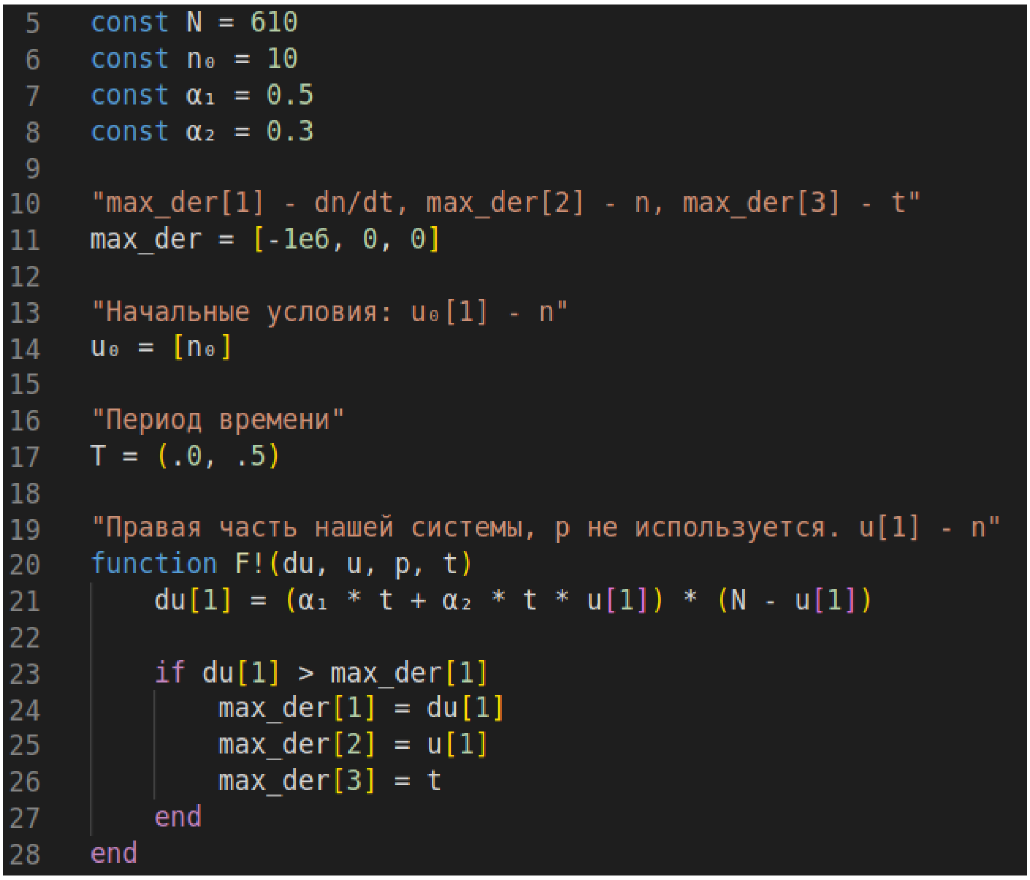


Рис. 4.14: Результат в виде графиков

4.2.3 Задание №3

1. Изменяем значения коэффициентов α_1 и α_2, период времени `T` и ОДУ в функции `F!` (подробное объяснение давалось в предыдущей главе) (рис. 4.15, 4.16).

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



```
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, .5)

"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
```

Рис. 4.15: Измененная часть кода

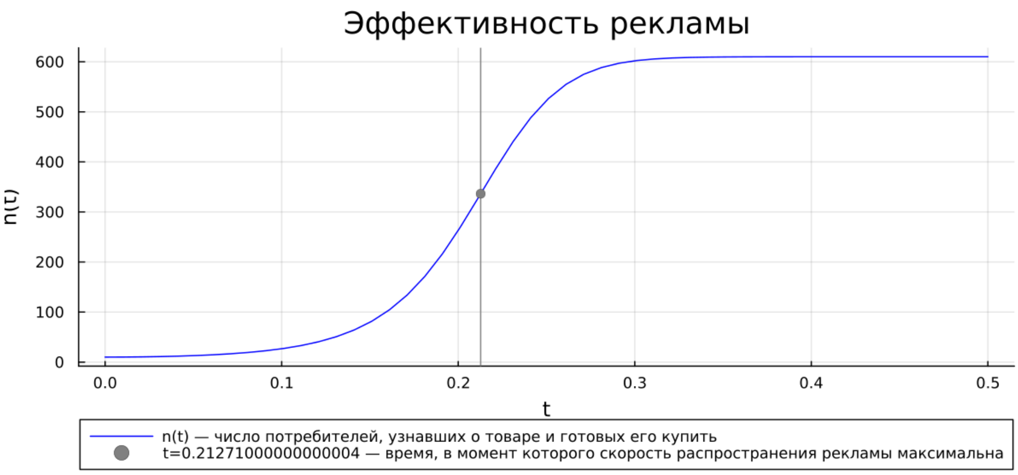


Рис. 4.16: Результат в виде графиков

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. 4.17, 4.18).

```
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;
```

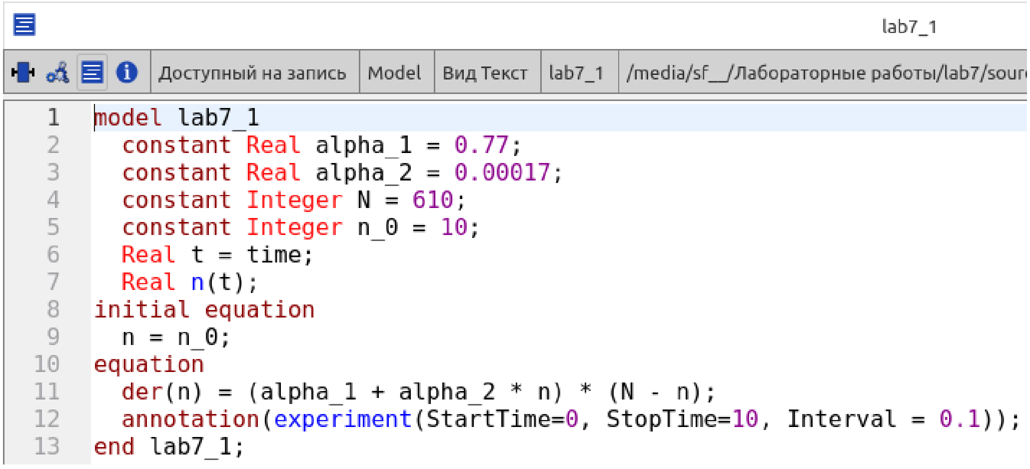


Рис. 4.17: Определяем коэффициенты, функцию n от времени, ОДУ, а также начальное/конечное время и частоту разбиения при симуляции

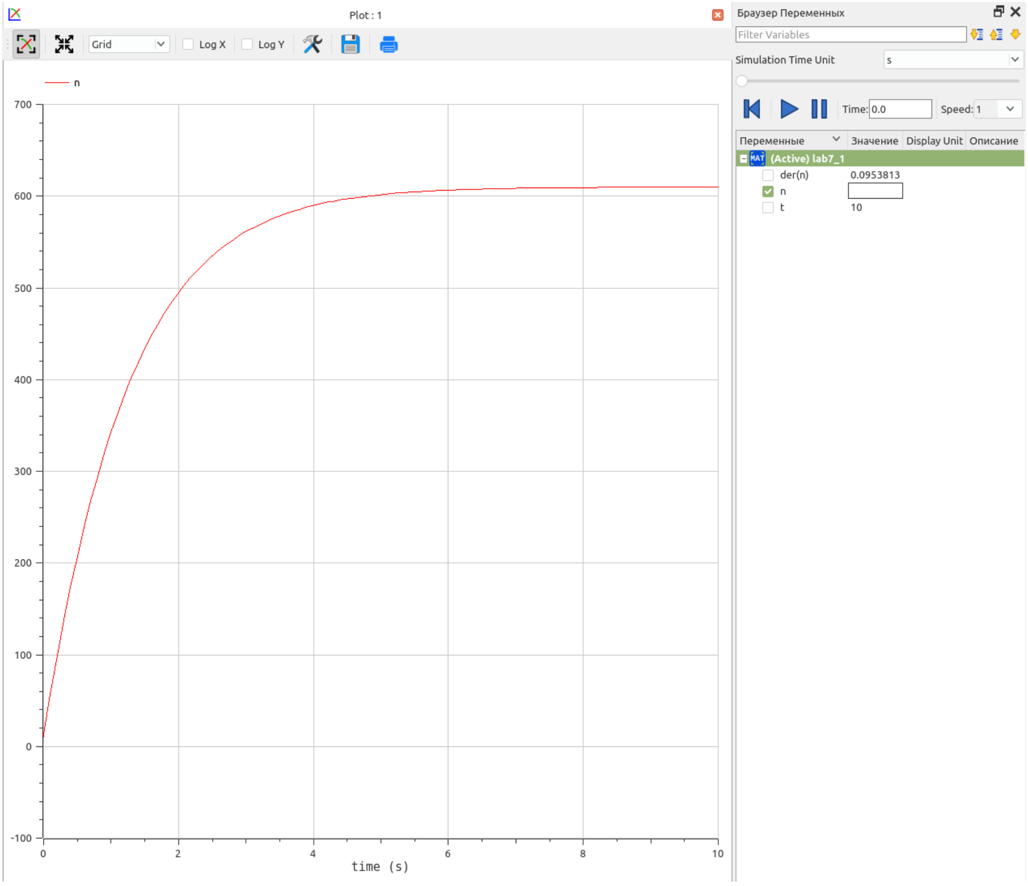


Рис. 4.18: Результат в виде графика зависимости n от t

4.3.2 Задание №2

1. По аналогии с Julia пишем программу для второго случая. Любуемся результатами (рис. 4.19, 4.20, 4.21).

```
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;
```

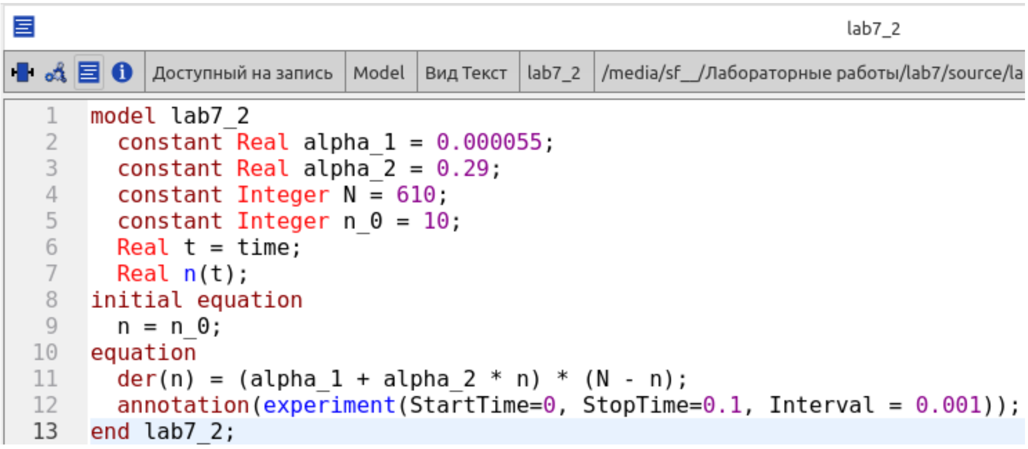


Рис. 4.19: По сравнению с предыдущим случаем изменяются значение коэффициентов `alpha_1`, `alpha_2`, период времени и частота разбиения

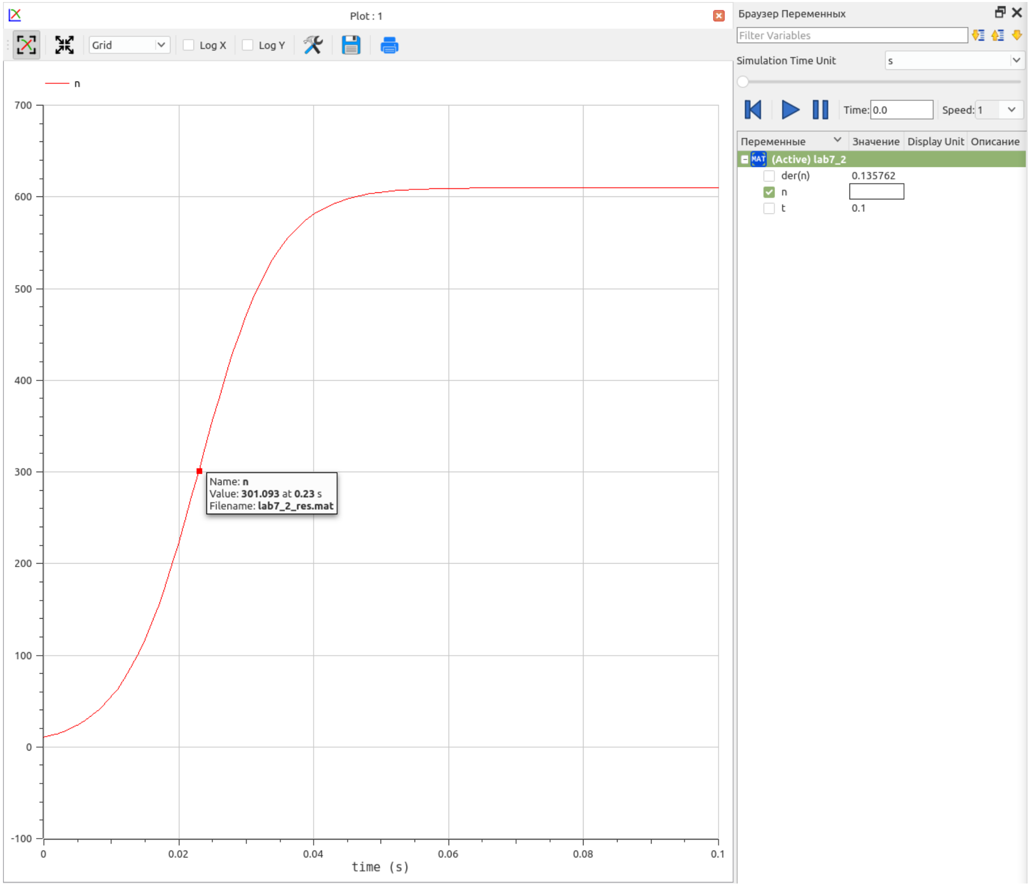


Рис. 4.20: Результат в виде графика зависимости n от t

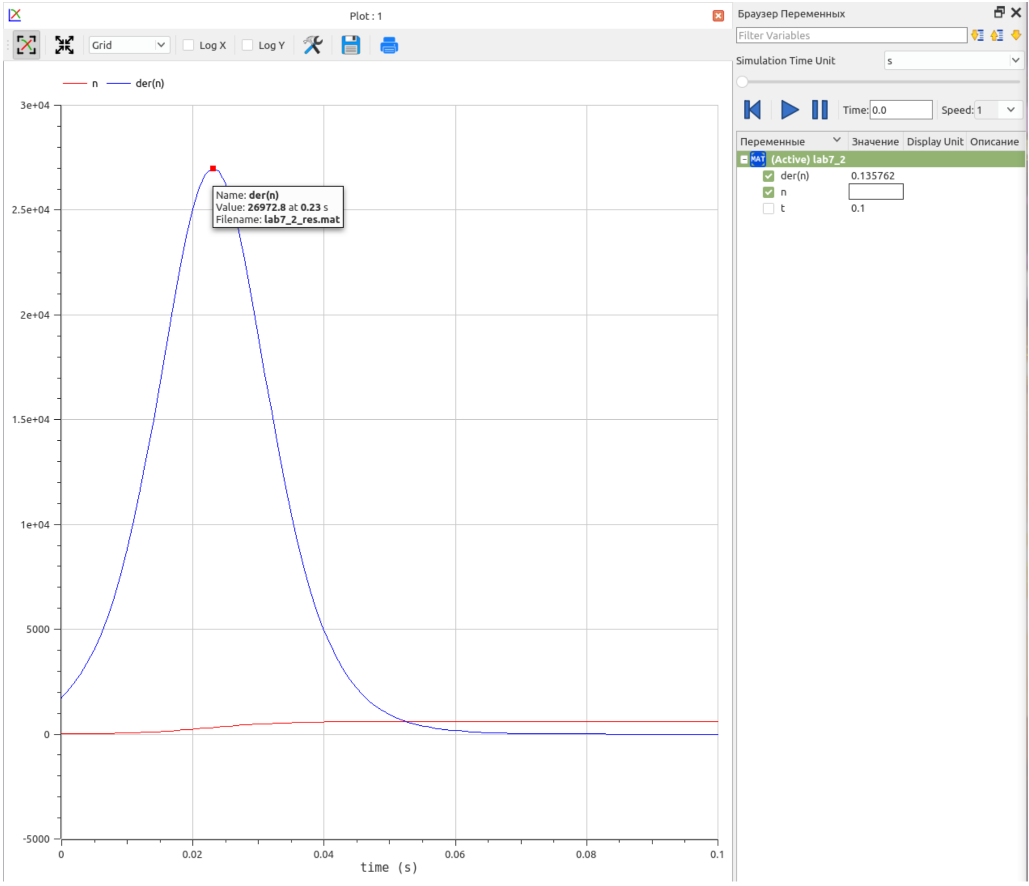


Рис. 4.21: Максимальное значение производной достигается в момент времени $t = 0.23$

4.3.3 Задание №3

1. По аналогии с Julia пишем программу для третьего случая. Любуемся результатами (рис. 4.22, 4.23).

```
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;
```

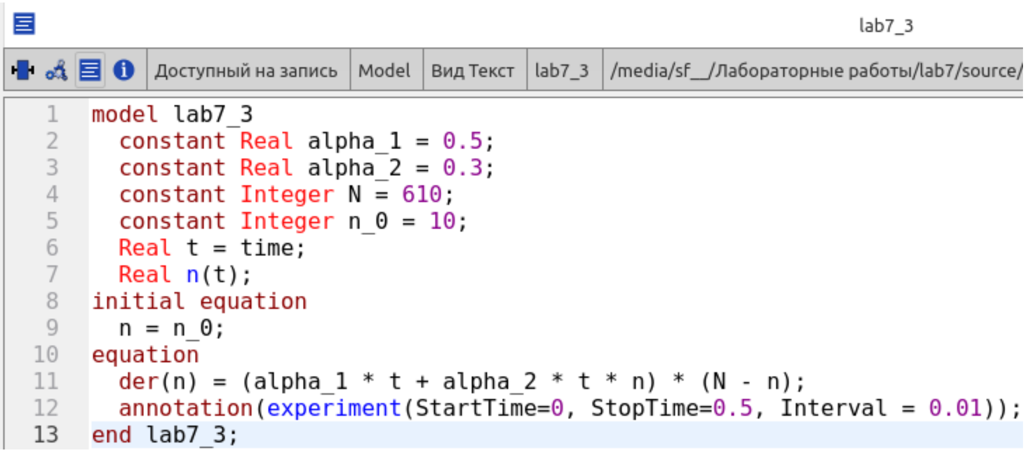


Рис. 4.22: По сравнению с предыдущим случаем измяняются значение коэффициентов `alpha_1`, `alpha_2`, ОДУ, период времени и частота разбиения

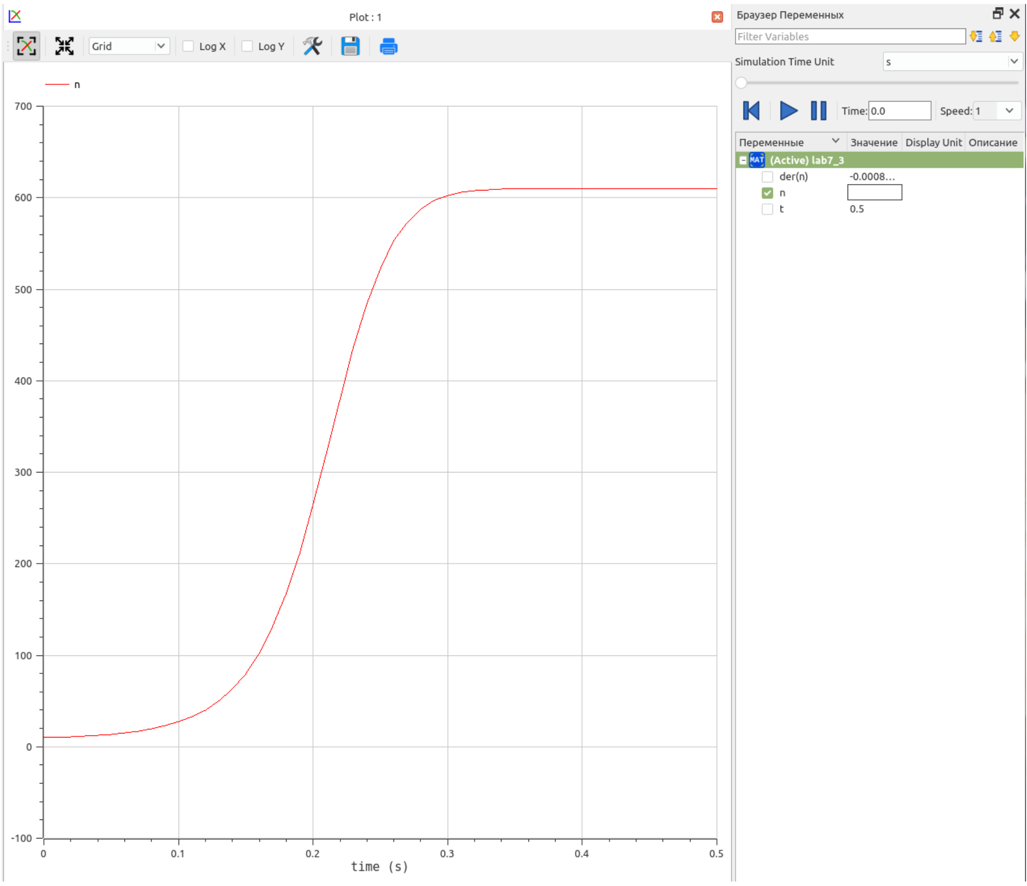


Рис. 4.23: Результат в виде графика зависимости n от t

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 $n(t)$ (т.е. $\dot{n}$) 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 $n(t)$ в этой точке.

При написании же программы на Modelica приходится вручную искать максимальное значение по графику производной функции $n(t)$.

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.