

Шаблон отчёта по лабораторной работе

Простейший вариант

Дмитрий Сергеевич Кулябов

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Отчёт 01_exponential_growth	9
6	Экспоненциальный рост	10
6.1	Инициализация проекта и загрузка пакетов	10
6.2	Определение модели	11
6.3	Первый запуск с параметрами по умолчанию	11
6.4	Визуализация результатов	13
6.5	Анализ результатов	14
6.6	Сохранение всех результатов	15
7	Отчёт 02_exponential_growth	16
8	Параметрическое исследование экспоненциального роста	17
8.1	Активация проекта и загрузка пакетов	17
8.2	Определение модели	18
8.3	Определение параметров в Dict	18
8.4	Функция-обертка для запуска одного эксперимента	19
8.5	Запуск базового эксперимента	20
8.6	Визуализация базового эксперимента	21
8.7	Параметрическое сканирование	22
8.8	Запуск всех экспериментов и сбор результатов	24
8.9	Анализ и визуализация результатов сканирования	25
8.10	Бенчмаркинг с разными параметрами	29
8.11	Сохранение всех результатов	32
9	Выводы	35
	Список литературы	36

Список иллюстраций

4.1	V Сольвеевский конгресс (1927) «Электроны и фотоны»	8
-----	---	---

Список таблиц

3.1	Описание некоторых каталогов файловой системы GNU Linux	7
-----	---	---

1 Цель работы

Здесь приводится формулировка цели лабораторной работы. Формулировки цели для каждой лабораторной работы приведены в методических указаниях.

Цель данного шаблона — максимально упростить подготовку отчётов по лабораторным работам. Модифицируя данный шаблон, студенты смогут без труда подготовить отчёт по лабораторным работам, а также познакомиться с основными возможностями разметки Markdown.

2 Задание

Здесь приводится описание задания в соответствии с рекомендациями методического пособия и выданным вариантом.

3 Теоретическое введение

Здесь описываются теоретические аспекты, связанные с выполнением работы. Например, в табл. 3.1 приведено краткое описание стандартных каталогов Unix.

Таблица 3.1: Описание некоторых каталогов файловой системы GNU Linux

Имя каталога	Описание каталога
/	Корневая директория, содержащая всю файловую
/bin	Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям
/etc	Общесистемные конфигурационные файлы и файлы конфигурации установленных программ
/home	Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя
/media	Точки монтирования для сменных носителей
/root	Домашняя директория пользователя root
/tmp	Временные файлы
/usr	Вторичная иерархия для данных пользователя

Более подробно про Unix см. в [1–4].

4 Выполнение лабораторной работы

Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию (рис. 4.1).



Рисунок 4.1: V Сольвеевский конгресс (1927) «Электроны и фотоны»

5 Отчёт 01_exponential_growth

6 Экспоненциальный рост

Цель: Исследовать решение уравнения $du/dt = \alpha u$.

6.1 Инициализация проекта и загрузка пакетов

```
using DrWatson
@quickactivate "../project"

using Plots
default(fmt = :png)
gr(fmt = :png)

using DifferentialEquations
using DataFrames
using JLD2

script_name = splitext(basename(PROGRAM_FILE))[1]
mkpath(plotsdir(script_name))
mkpath(datadir(script_name))
```

```
└ Warning: DrWatson could not find find a project file by recursively cl
⊠ (given dir: C:\Users\12232)
└ @ DrWatson C:\Users\12232\.julia\packages\DrWatson\2QF5p\src\proje
```

```
"C:\\Users\\12232\\Documents\\GitHub\\2026-1--study--simulation-modeling\\labs\\lab01\\project\\data\\"
```

6.2 Определение модели

Уравнение экспоненциального роста: $\frac{du}{dt} = \alpha u$, $u(0) = u_0$

```
function exponential_growth!(du, u, p, t)
    α = p
    du[1] = α * u[1]
end
```

```
exponential_growth! (generic function with 1 method)
```

6.3 Первый запуск с параметрами по умолчанию

Зададим начальные параметры:

```
u0 = [1.0] # 1x1 matrix of Float64
α = 0.3 # 1x1 matrix of Float64
tspan = (0.0, 10.0) # 1x1 matrix of Float64

prob = ODEProblem(exponential_growth!, u0, tspan, α)
sol = solve(prob, Tsit5(), saveat=0.1)
```

```
retcode: Success
```

```
Interpolation: 1st order linear
```

```
t: 101-element Vector{Float64}:
```

```
 0.0
```

```
 0.1
```

0.2

0.3

0.4

0.5

0.6

0.7

0.8

0.9

1.0

1.1

1.2

⊠

8.9

9.0

9.1

9.2

9.3

9.4

9.5

9.6

9.7

9.8

9.9

10.0

u: 101-element Vector{Vector{Float64}}:

[1.0]

[1.030454533950446]

[1.0618365551529674]

[1.0941743028794098]

[1.1274968605386673]
[1.1618342327450653]
[1.1972173476990624]
[1.233678057187257]
[1.2712491879500347]
[1.3099646073864084]
[1.3498590188273822]
[1.390968273354968]
[1.4333293964993763]

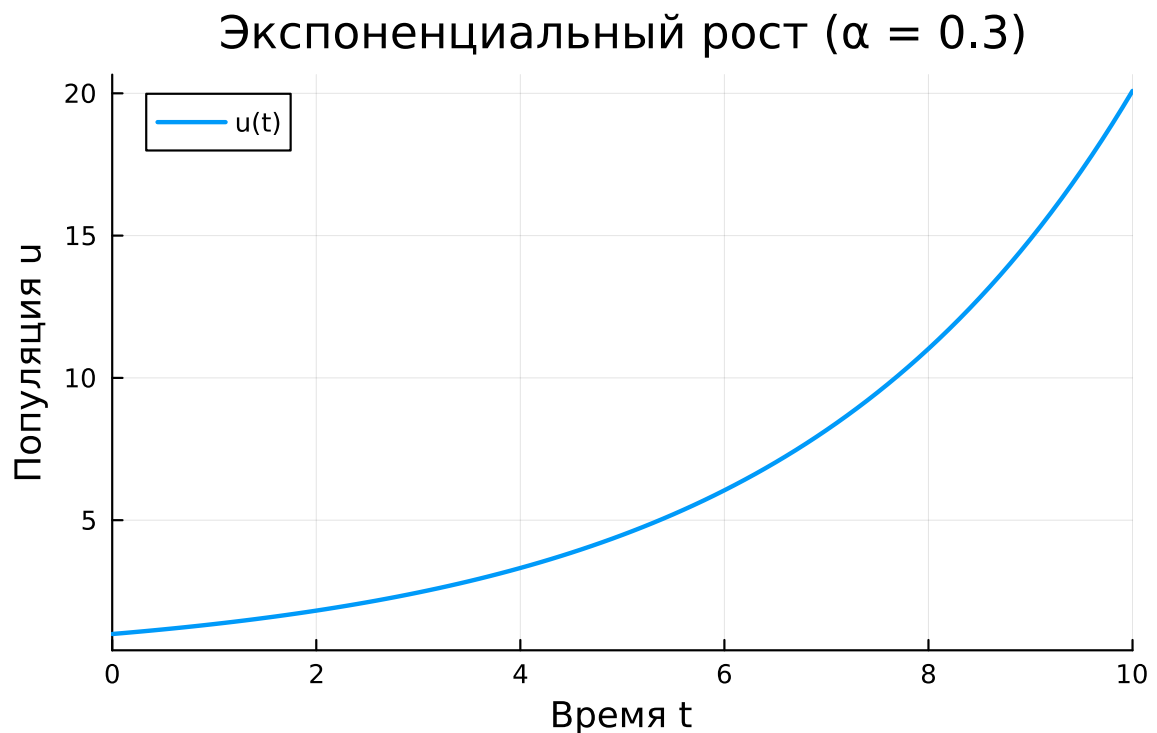
⊠

[14.439892233074872]
[14.87962750503581]
[15.33275596650523]
[15.79968870415309]
[16.280848966976613]
[16.776672166300525]
[17.287605875776965]
[17.814109831385352]
[18.35665593143248]
[18.91572823655283]
[19.491822969707854]
[20.08544851618676]

6.4 Визуализация результатов

Построим график решения:

```
plot(sol, label="u(t)", xlabel="Время t", ylabel="Популяция u",
      title="Экспоненциальный рост ( $\alpha = 0.3$ )", lw=2, legend=:topleft)
```



Сохраним график в папку plots

```
savefig(plotsdir(script_name, "exponential_growth_α=$α.png"))
```

```
"C:\\Users\\12232\\Documents\\GitHub\\2026-1--study--
simulation-modeling\\labs\\lab01\\project\\plots\\exponential_growt
```

6.5 Анализ результатов

Создадим таблицу с данными:

```
df = DataFrame(t=sol.t, u=first.(sol.u))
println("Время 5: ")
println(first(df, 5))
```

5 rows × 2 columns:

5×2 DataFrame

Row	t	u
	Float64	Float64
1	0.0	1.0
2	0.1	1.03045
3	0.2	1.06184
4	0.3	1.09417
5	0.4	1.1275

Вычислим удвоение популяции:

```
u_final = last(sol.u)[1]
doubling_time = log(2) / λ
println("\nУдвоение популяции: ", round(doubling_time; digits=2))
```

Удвоение популяции: 2.31

6.6 Сохранение всех результатов

```
@save datadir(script_name, "all_results.jld2") df
```

7 Отчёт 02_exponential_growth

8 Параметрическое исследование экспоненциального роста

8.1 Активация проекта и загрузка пакетов

ИЗМЕНЕНИЕ: Добавлен DrWatson для управления проектом и параметрами

```
using DrWatson
@quickactivate "../project" # XXXXXXXX XXXXXX DrWatson

using Plots
default(fmt = :png)
gr(fmt = :png)

using DifferentialEquations
using DataFrames
using JLD2
using BenchmarkTools
```

```
└ Warning: DrWatson could not find find a project file by recursively cl
  (given dir: C:\Users\12232)
└ @ DrWatson C:\Users\12232\.julia\packages\DrWatson\2QF5p\src\proje
```

Установка каталогов

```
script_name = splitext(basename(PROGRAM_FILE))[1]
mkpath(plotsdir(script_name))
mkpath(datadir(script_name))
```

```
"C:\\Users\\12232\\Documents\\GitHub\\2026-1--study--
simulation-modeling\\labs\\lab01\\project\\data\\"
```

8.2 Определение модели

Модель: $du/dt = \alpha \cdot u$

```
function exponential_growth!(du, u, p, t)
    α = p.α # **XXXXXXXXXX:** XXXXXXXX XXXXXXX XXXXXXXXXXX XXX XXXXXXXXXXX XXXXXXX
    du[1] = α * u[1]
end
```

exponential_growth! (generic function with 1 method)

8.3 Определение параметров в Dict

ОСНОВНОЕ ИЗМЕНЕНИЕ: Все параметры собраны в Dict для систематизации
Базовый набор параметров (один эксперимент)

```
base_params = Dict(
    :u0 => [1.0], # XXXXXXXX XXXXXXXX
    :α => 0.3, # XXXXXXXX XXXXXXX
    :tspan => (0.0, 10.0), # XXXXXXXX XXXXXXXX
    :solver => Tsit5(), # XXXXXXX XXXXXXXX
    :saveat => 0.1, # XXX XXXXXXXXXXX XXXXXXXXXXXX
    :experiment_name => "base_experiment"
```

```
)

println("XXXXXXXX XXXXXXXXXXX XXXXXXXXXXXXXXX: ")
for (key, value) in base_params
    println(" $key = $value")
end
```

XXXXXXXX XXXXXXXXXXX XXXXXXXXXXXXXXX:

λ = 0.3

u0 = [1.0]

saveat = 0.1

solver = Tsit5{typeof(OrdinaryDiffEqCore.trivial_limiter!), typeof(

experiment_name = base_experiment

tspan = (0.0, 10.0)

8.4 Функция-обертка для запуска одного эксперимента

ИСПРАВЛЕНИЕ: Возвращаем Dict со строковыми ключами

```
function run_single_experiment(params::Dict)
    @unpack u0,  $\lambda$ , tspan, solver, saveat = params
    prob = ODEProblem(exponential_growth!, u0, tspan, ( $\lambda$ = $\lambda$ ,)) # XXXXXXX
    sol = solve(prob, solver; saveat=saveat)
    final_population = last(sol.u)[1] # XXXXXXX XXXXXXXXXXXXXXX
    doubling_time = log(2) /  $\lambda$ 
    return Dict(
        "solution" => sol,
        "time_points" => sol.t,
```

end

```
run_single_experiment (generic function with 1 method)
```

8.5 Запуск базового эксперимента

ИЗМЕНЕНИЕ: Используем produce_or_load для автоматического кэширования

```
println("  📁 📁: ", path)
```

XXXXXXXXXX XXXXXXXXXXXX: 20.08544851618676

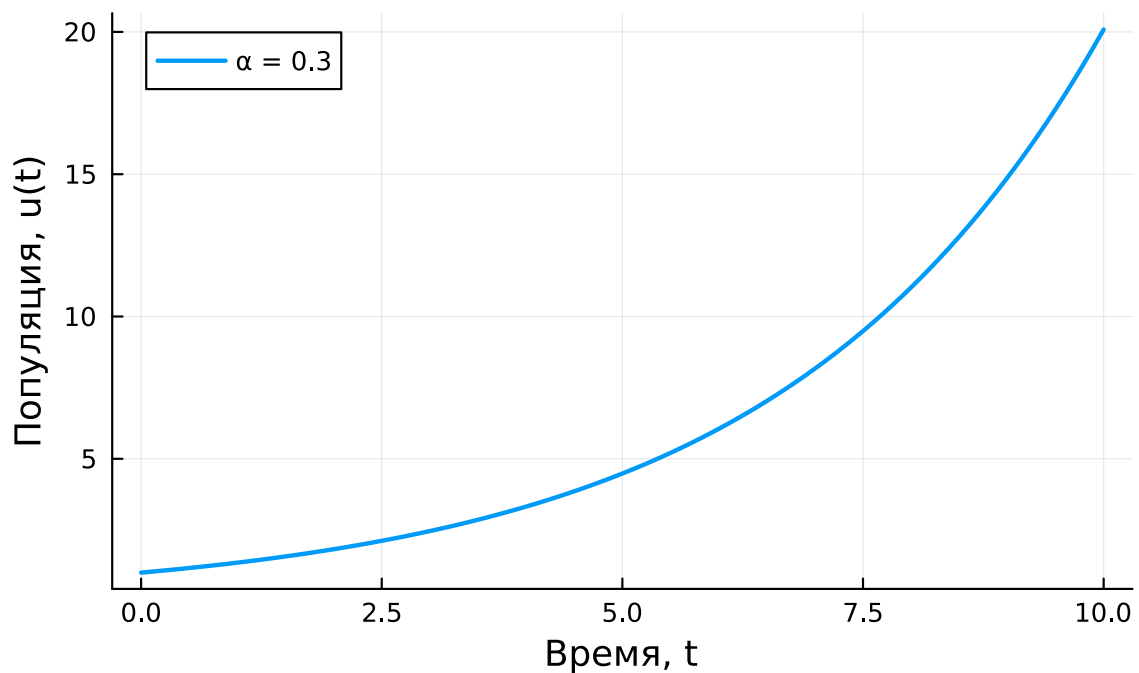
XXXXX XXXXXXXXXXXX: 2.31

XXXXX XXXXXXXXXXXX: C:\Users\12232\Documents\GitHub\2026-1--
study--simulation-modeling\labs\lab01\project\data\single\exp_growt

8.6 Визуализация базового эксперимента

```
p1 = plot(data["time_points"], data["population_values"],  
          label="X = $(base_params[:X])",  
          xlabel="XXXXX, t",  
          ylabel="XXXXXXXXXX, u(t)",  
          title="XXXXXXXXXXXXXXXXXXXX XXXX (XXXXXXX XXXXXXXXXXXX)",  
          lw=2,  
          legend=:topleft,  
          grid=true  
)
```

Экспоненциальный рост (базовый эксперимент)



Сохраним график в папку plots

```
savefig(plotsdir(script_name, "single_experiment.png"))
```

```
"C:\\Users\\12232\\Documents\\GitHub\\2026-1--study--  
simulation-modeling\\labs\\lab01\\project\\plots\\single_experiment
```

8.7 Параметрическое сканирование

НОВАЯ СЕКЦИЯ: Исследование влияния параметра α

Сетка параметров для сканирования

```
param_grid = Dict(  
    :u0 => [[1.0]], #   
    :alpha => [0.1, 0.3, 0.5, 0.8, 1.0], #   
    :tspan => [(0.0, 10.0)], #   
)
```

```

:solver => [Tsit5()],          # 1000000000 10000 1000000000
:saveat => [0.1],             # 1000000000 1000 10000000000
:experiment_name => ["parametric_scan"]
)

```

Dict{Symbol, Vector} with 6 entries:

```

:Δ          => [0.1, 0.3, 0.5, 0.8, 1.0]
:u0         => [[1.0]]
:saveat     => [0.1]
:solver     => [Tsit5{typeof(trivial_limiter!), typeof(trivial_li
:experiment_name => ["parametric_scan"]
:tspan      => [(0.0, 10.0)]

```

Генерация всех комбинаций параметров

```

all_params = dict_list(param_grid)

println("\n" * "="^60)
println("XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX")
println("XXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX: ", length(all_params))
println("XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Δ: ", param_grid[:Δ])
println("="^60)

```

```

=====
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX: 5
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Δ: [0.1, 0.3, 0.5, 0.8, 1.0]
=====

```

8.8 Запуск всех экспериментов и сбор результатов

НОВАЯ СЕКЦИЯ: Автоматический запуск и сохранение всех вариантов

```
all_results = []
all_dfs = []

for (i, params) in enumerate(all_params)
  println("Experiment: $i/$(length(all_params)) | P = $(params[:P])")

  data, path = produce_or_load(
    datadir(script_name, "parametric_scan"), # Directory
    params, # Parameters
    run_single_experiment, # Function to run a single experiment
    prefix = "scan", # Prefix for the data files
    tag = false, # Whether to tag the data files
    verbose = false # Whether to print verbose output
  ) # Returns a DataFrame and a path

  result_summary = merge(
    params,
    Dict{
      :final_population => data["final_population"],
      :doubling_time => data["doubling_time"],
      :filepath => path # Path to the data files
    }
  ) # Returns a Dict containing the parameters and the results

  push!(all_results, result_summary)
```

```

df = DataFrame(
  t = data["time_points"],
  u = data["population_values"],
  ⬚ = fill(params[:⬚], length(data["time_points"]))
) # ⬚⬚⬚⬚⬚⬚⬚⬚ ⬚⬚⬚⬚⬚ ⬚⬚⬚⬚⬚ ⬚⬚⬚ ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚
push!(all_dfs, df)
end

```

```

⬚⬚⬚⬚⬚⬚⬚⬚: 1/5 | ⬚ = 0.1
⬚⬚⬚⬚⬚⬚⬚⬚: 2/5 | ⬚ = 0.3
⬚⬚⬚⬚⬚⬚⬚⬚: 3/5 | ⬚ = 0.5
⬚⬚⬚⬚⬚⬚⬚⬚: 4/5 | ⬚ = 0.8
⬚⬚⬚⬚⬚⬚⬚⬚: 5/5 | ⬚ = 1.0

```

8.9 Анализ и визуализация результатов сканирования

НОВАЯ СЕКЦИЯ: Сравнительный анализ всех экспериментов

Сводная таблица результатов

```

results_df = DataFrame(all_results)
println("\n⬚⬚⬚⬚⬚⬚⬚⬚ ⬚⬚⬚⬚⬚⬚ ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚:")
println(results_df[!, [:⬚, :final_population, :doubling_time]])

```

⬚⬚⬚⬚⬚⬚⬚⬚ ⬚⬚⬚⬚⬚⬚⬚ ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚:

5×3 DataFrame

Row	⬚	⬚	final_population	doubling_time
-----	---	---	------------------	---------------

	Float64	Float64	Float64
1	0.1	2.71828	6.93147
2	0.3	20.0854	2.31049
3	0.5	148.409	1.38629
4	0.8	2980.57	0.866434
5	1.0	22021.0	0.693147

Сравнительный график всех траекторий

```
p2 = plot(size=(800, 500), dpi=150)

for params in all_params

    data, _ = produce_or_load(
        datadir(script_name, "parametric_scan"),
        params,
        run_single_experiment,
        prefix = "scan"
    ) # (time_points, population_values)

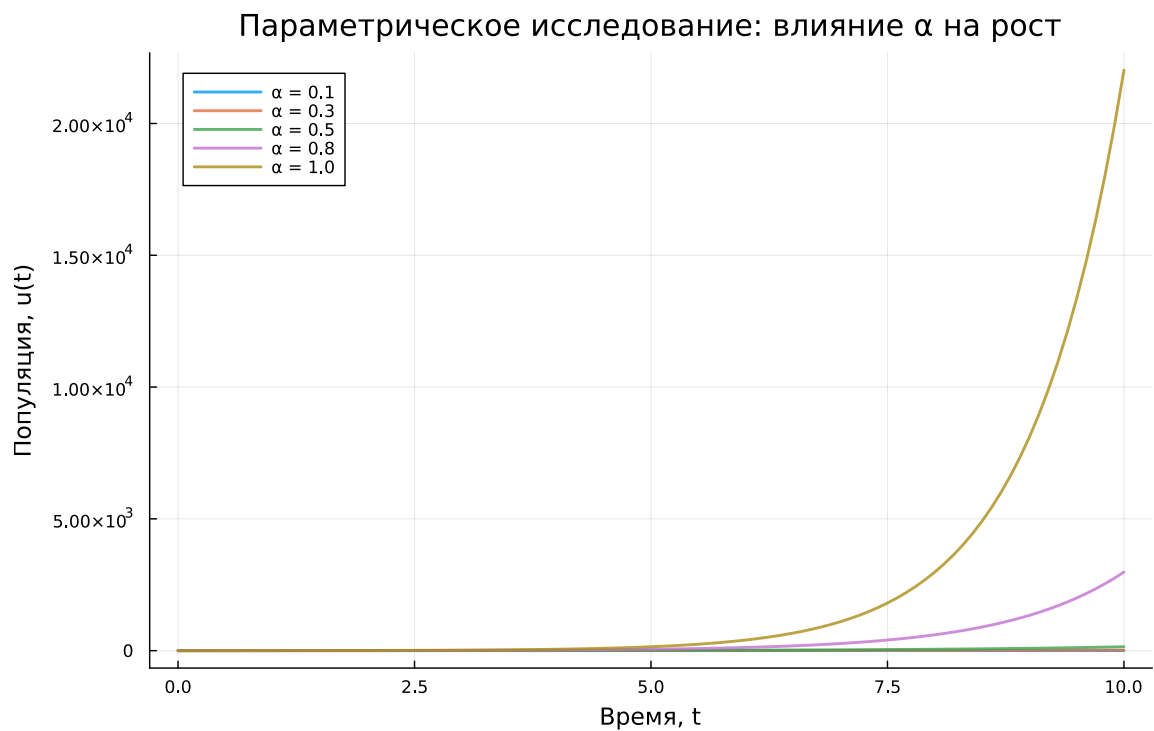
    plot!(p2, data["time_points"], data["population_values"],
        label="$x = $(params[:x])",
        lw=2,
        alpha=0.8
    )
end

plot!(p2,
    xlabel="time, t",
```

```

ylabel="Популяция, u(t)",
title="Параметрическое исследование: влияние α на рост",
legend=:topleft,
grid=true
)

```



Сохраним график в папку plots

```

savefig(plotsdir(script_name, "parametric_scan_comparison.png"))

```

"C:\\Users\\12232\\Documents\\GitHub\\2026-1--study--simulation-modeling\\labs\\lab01\\project\\plots\\parametric_scan_c

График зависимости времени удвоения от α

```

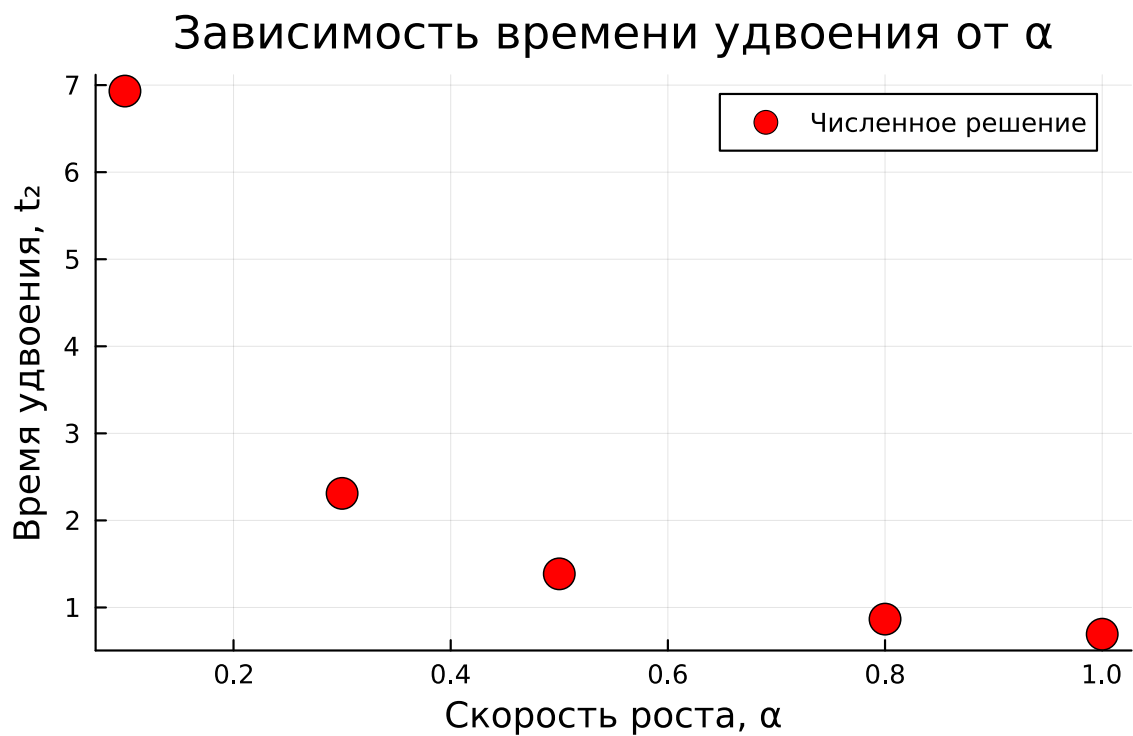
p3 = plot(results_df[:, results_df.doubling_time,
seriestype=:scatter,

```

```

label="Численное решение",
xlabel="Скорость роста, α",
ylabel="Время удвоения, t₂",
title="Зависимость времени удвоения от α",
markersize=8,
markercolor=:red,
legend=:topright
)

```



Теоретическая кривая: $t_2 = \ln(2)/\alpha$

```

α_range = 0.1:0.01:1.0

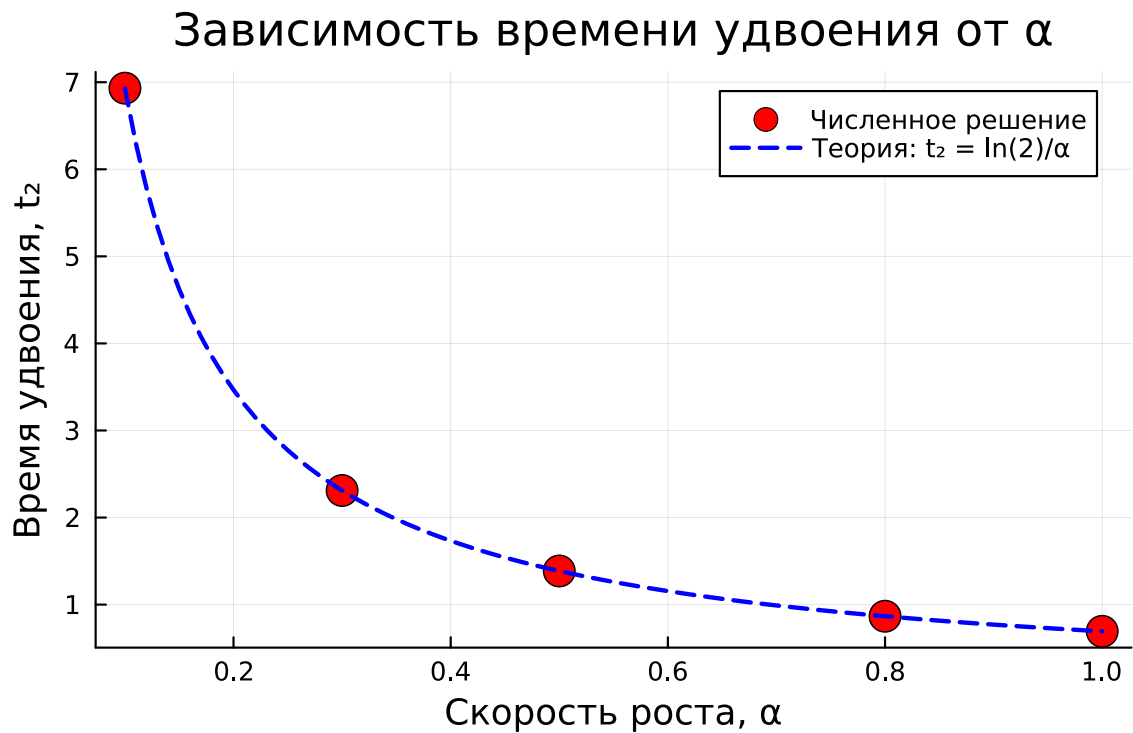
plot!(p3, α_range, log(2) ./ α_range,
      label="Теоретическая кривая: t₂ = ln(2)/α",
      lw=2,

```

```

linestyle=:dash,
linecolor=:blue
)

```



Сохраним график в папку plots

```

savefig(plotsdir(script_name, "doubling_time_vs_alpha.png"))

```

```

"C:\\Users\\12232\\Documents\\GitHub\\2026-1--study--
simulation-modeling\\labs\\lab01\\project\\plots\\doubling_time_vs_

```

8.10 Бенчмаркинг с разными параметрами

ИЗМЕНЕНИЕ: Бенчмаркинг для разных значений α

```
println("\n" * "="^60)

println("XXXXXXXXXX XX XXXXX XXXXXXXX X")

println("="^60)

benchmark_results = []

for x_value in param_grid[:x]

    bench_params = Dict(
        :u0 => [1.0],
        :x => x_value,
        :tspan => (0.0, 10.0),
        :solver => Tsit5(),
        :saveat => 0.1
    ) # XXXXXXXXXXXXXXXXXXXX XXX XXXXXXXXXXXX

    function benchmark_run() # XXXXXXXX XX XXXXXXXX
        prob = ODEProblem(exponential_growth!,
            bench_params[:u0],
            bench_params[:tspan],
            (x=bench_params[:x],))
        return solve(prob, bench_params[:solver];
            saveat=bench_params[:saveat])
    end

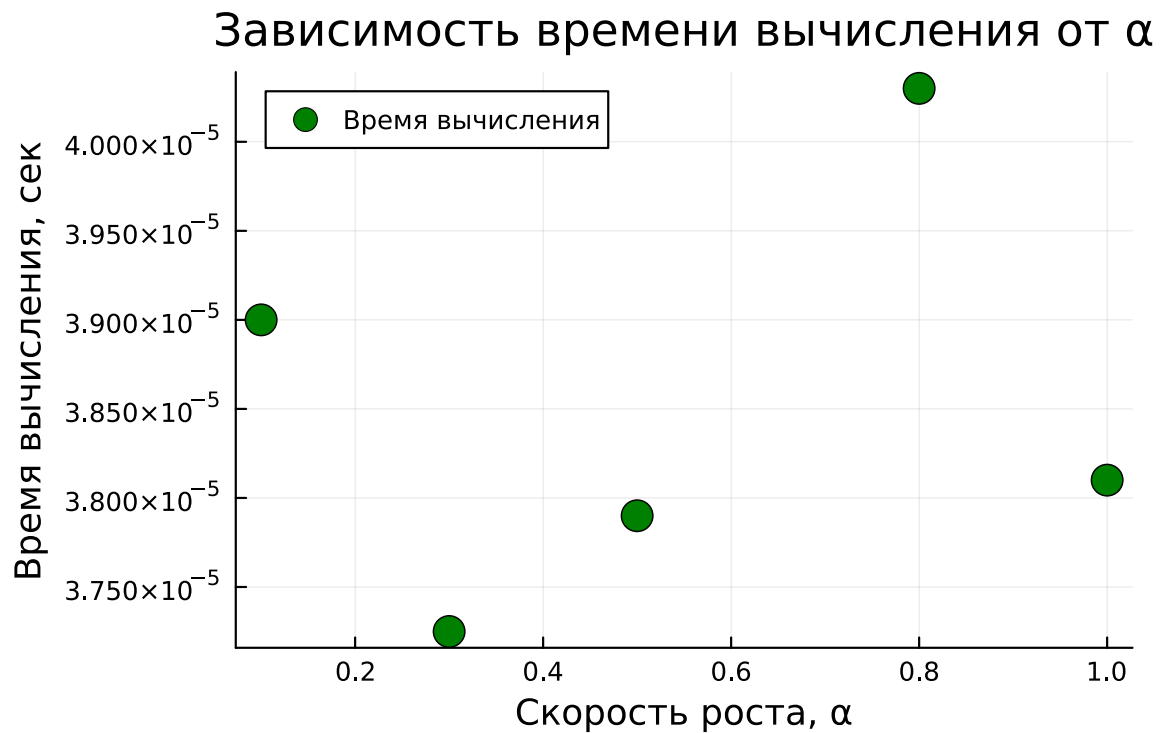
    println("\nXXXXXXXXXX XX x = $x_value:")
    b = @benchmark $benchmark_run() samples=100 evals=1 # XXXXXXXX XXXXXXXX
    push!(benchmark_results, (x=x_value, time=median(b).time/1e9))
end
```



```

ylabel="Время вычисления, сек",
title="Зависимость времени вычисления от  $\alpha$ ",
markersize=8,
markercolor=:green,
legend=:topleft
)

```



Сохраним график в папку plots

```

savefig(plotsdir(script_name, "computation_time_vs_alpha.png"))

```

```

"C:\\Users\\12232\\Documents\\GitHub\\2026-1--study--
simulation-modeling\\labs\\lab01\\project\\plots\\computation_time_

```

8.11 Сохранение всех результатов

НОВАЯ СЕКЦИЯ: Сохранение сводных данных для последующего анализа

```

@save datadir(script_name, "all_results.jld2") base_params param_gr

@save datadir(script_name, "all_plots.jld2") p1 p2 p3 p4

println("\n" * "="^60)
println("XXXXXXXXXXXXXXXX XXXXXX XXXXXXXXXXXX")
println("="^60)
println("\nXXXXXXXXXXXXXXXX XXXXXXXXXXXX X:")
println("  • data/${script_name}/single/                - XXXXXXXX XXXXXXXX")
println("  • data/${script_name}/parametric_scan/          - XXXXXXXXXXXXXXXXXXXX")
println("  • data/${script_name}/all_results.jld2          - XXXXXXXX XXXXXXXX")
println("  • plots/${script_name}/                        - XXX XXXXXXXX")
println("  • data/${script_name}/all_plots.jld2            - XXXXXXXX XXXXXXXX")
println("\nXXX XXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXX:")
println("  using JLD2, DataFrames")
println("  @load \"data/${script_name}/all_results.jld2\"")
println("  println(results_df)")

```

=====

XXXXXXXXXXXXXXXX XXXXXX XXXXXXXXXXXX

=====

XXXXXXXXXXXXXXXX XXXXXXXXXXXX X:

• data//single/	- XXXXXXXX XXXXXXXXXXXX
• data//parametric_scan/	- XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
• data//all_results.jld2	- XXXXXXXX XXXXXXXX
• plots//	- XXX XXXXXXXX
• data//all_plots.jld2	- XXXXXXXX XXXXXXXX

```
using JLD2, DataFrames
```

```
@load "data//all_results.jld2"
```

```
println(results_df)
```

9 Выводы

Здесь кратко описываются итоги проделанной работы.

Список литературы

1. *Newham C.* Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 p. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learning-bash-Shell-Programming-Nutshell/dp/0596009658>.
2. *Robbins A.* Bash Pocket Reference. — O'Reilly Media, 2016. — 156 p. — ISBN 978-1491941591.
3. *Zarrelli G.* Mastering Bash. — Packt Publishing, 2017. — 502 p. — ISBN 9781784396879.
4. *Таненбаум Э., Бос Х.* Современные операционные системы.. — 4-е изд. — СПб. : Питер, 2015.. — 1120 с. — (Классика Computer Science).