

Отчет по лабораторной работе №2

по дисциплине: Математическое моделирование

Ким Михаил Алексеевич

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
3.1	Pluto Notebook	6
3.2	Модель Ланчестера	7
4	Выполнение лабораторной работы	8
4.1	Подготовка системы для работы	8
4.1.1	Установка Pluto	8
4.2	Выполнение лабораторной работы	9
4.2.1	Pluto.jl	9
4.2.2	Julia	20
4.2.3	OpenModelica	26
5	Анализ результатов	33
6	Выводы	34
	Список литературы	35

Список иллюстраций

4.1	Установка пакета Pluto	8
4.2	Запуск Pluto	8
4.3	Установка пакета LaTeXStrings	9
4.4	Импорт библиотек. Определение коэффициентов, начальных условий, периода времени, функции для решения ОДУ	13
4.5	Формирование проблемы и численный поиск решений	14
4.6	Исключение численности войск меньше нуля	15
4.7	Отрисовка графиков	16
4.8	Измененные коэффициенты, период времени	19
4.9	Измененные графики	20
4.10	Код повторяет код из Pluto.jl	22
4.11	Результат работы программы	23
4.12	Измененные строки кода по сравнению с предыдущим листингом	25
4.13	Результат работы программы	26
4.14	Задание констант, изменяющихся переменных, начальных условий, системы уравнений, времени моделирования	27
4.15	График зависимости численности армии от времени	28
4.16	График зависимости численности армии X от армии Y	29
4.17	Задание иных констант, системы уравнений, времени моделирования	30
4.18	График зависимости численности армии от времени	31
4.19	График зависимости численности армии X от армии Y	32

1 Цель работы

Ознакомиться с базовым функционалом интерактивного блокнота Pluto. Рассмотреть простейшую модель боевых действий — модель Ланчестера. Построить данную математическую модель при помощи языка программирования Julia, интерактивного блокнота Pluto, языка моделирования Modelica и программного обеспечения OpenModelica.

2 Задание

1. Записать модель ведения боевых действий между регулярными войсками при помощи системы ОДУ.
2. Записать модель ведения боевых действий с участием регулярных войск и партизанских отрядов при помощи системы ОДУ.
3. Задать начальные условия задачи.
4. Постройте графики изменения численности войск армий.

3 Теоретическое введение

3.1 Pluto Notebook

Pluto Notebook - это интерактивный Jupyter Notebook-подобный инструмент для языка программирования Julia. Pluto Notebook позволяет выполнять вычисления, создавать графики и диаграммы, а также описывать отчеты, объединяя код, результаты вычислений и текст в одном документе [1].

Одним из главных преимуществ Pluto Notebook является возможность просматривать результаты вычислений в режиме реального времени, что упрощает отладку и улучшает итеративный процесс написания кода. Кроме того, Pluto Notebook имеет простой и интуитивно понятный пользовательский интерфейс [2].

Pluto Notebook также обладает рядом других преимуществ, включая возможность автоматического обновления результатов вычислений при изменении входных данных, поддержку встроенных виджетов и возможность экспорта документов в различные форматы, включая HTML и PDF [3].

В целом, Pluto Notebook представляет собой мощный и удобный инструмент для проведения вычислений и написания отчетов на языке программирования Julia.

3.2 Модель Ланчестера

Модель боевых действий Ланчестера - это математическая модель, которая используется для описания боевых действий на ранних стадиях войны. Она была разработана Фредериком Ланчестером в начале 20-го века и с тех пор была широко применена в военном делопроизводстве.

Суть модели заключается в том, что она позволяет определить относительную эффективность двух воюющих сторон в зависимости от количества сил каждой стороны. При этом модель предполагает, что вооруженные силы каждой стороны действуют независимо друг от друга и стремятся уничтожить противника [4].

Ключевым понятием в модели является коэффициент Ланчестера, который определяет относительную мощность каждой стороны. Он рассчитывается на основе количества единиц техники и личного состава каждой стороны. Чем больше коэффициент Ланчестера у одной стороны, тем больше вероятность ее победы в бою.

Модель Ланчестера может быть применена для анализа различных сценариев боевых действий, таких как атака/оборона, маневренная война и т.д. Она также может быть использована для принятия решений по размещению и использованию вооруженных сил в различных ситуациях.

Хотя модель Ланчестера имеет свои ограничения и не учитывает многие факторы, такие как ландшафт, погодные условия и тактику противника, она все еще является полезным инструментом для анализа и планирования военных действий на ранних стадиях войны [5].

4 Выполнение лабораторной работы

4.1 Подготовка системы для работы

4.1.1 Установка Pluto

1. В интерактивной командной строке REPL языка программирования Julia переходим в менеджер пакетов и устанавливаем пакет **PLuto** (рис. 4.1).

```
julia  
]  
add Pluto
```



A screenshot of the Julia REPL terminal. The prompt is `(@v1.8) pkg>` and the command entered is `add Pluto`. The text is displayed in a monospaced font with syntax highlighting: `(@v1.8)` is blue, `pkg>` is green, and `add Pluto` is orange.

Рис. 4.1: Установка пакета Pluto

2. Запускаем интерактивный блокнот Pluto (рис. 4.2).

```
import Pluto; Pluto.run()
```



A screenshot of the Julia REPL terminal. The prompt is `julia>` and the command entered is `import Pluto; Pluto.run()`. The text is displayed in a monospaced font with syntax highlighting: `julia>` is green, and `import Pluto; Pluto.run()` is orange.

Рис. 4.2: Запуск Pluto

3. Устанавливаем дополнительно пакет **LaTeXStrings** для корректной отрисовки графиков внутри Pluto (рис. 4.3).

```
julia  
]  
add LaTeXStrings
```

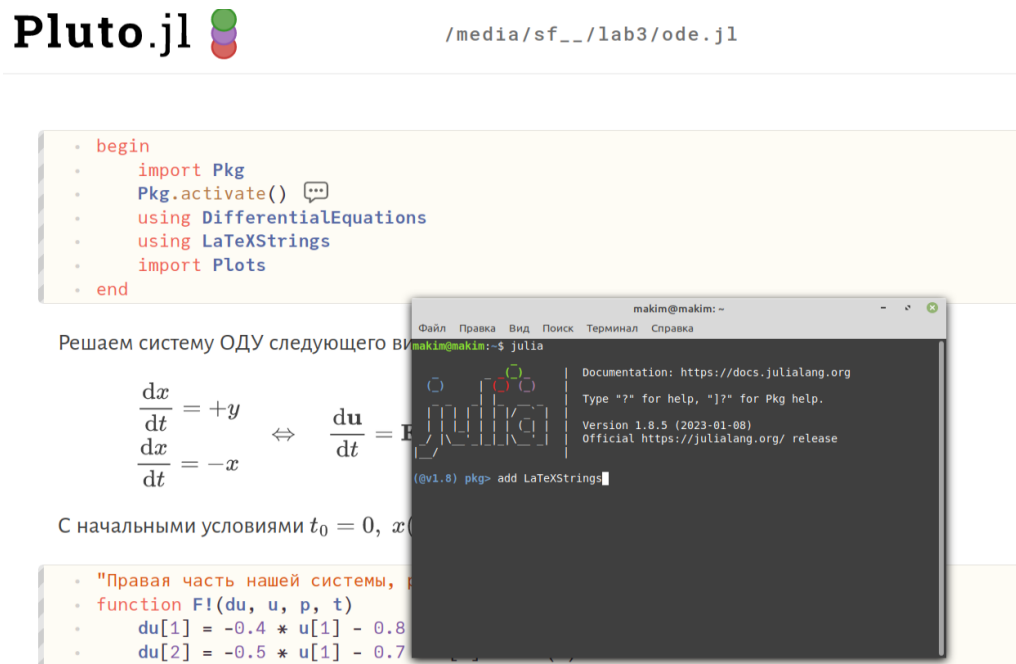


Рис. 4.3: Установка пакета LaTeXStrings

4.2 Выполнение лабораторной работы

4.2.1 Pluto.jl

1. Пишем программу, описывающую модель боевых действий между регулярными войсками (рис. 4.4, 4.5, 4.6, 4.7).

```
begin  
    import Pkg
```

```

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

begin
    const a = 0.333
    const b = 0.777
    const c = 0.5
    const h = 0.65

    "Начальные условия: u[1] -- x, u[2] -- y"
    u0 = [10000, 29000]

    "Период времени"
    T = (0.0, 1.8)
end

"Правая часть нашей системы, p, t не используются. u[1] -- x, u[2] -- y"
function F!(du, u, p, t)
    du[1] = -a * u[1] - b * u[2] + 1.6 * sin(t)
    du[2] = -c * u[1] - h * u[2] + 1.7 * cos(t + 2)
end

prob = ODEProblem(F!, u0, T)

sol = solve(prob, saveat=0.01)

begin
    const xx = []
    const yy = []
    for u in sol.u

```

```

x, y = u

if x < 0 || y < 0
    break
end

push!(xx, x)
push!(yy, y)
end

Time = sol.t[1:size(xx)[1]]
Time

end

begin
    fig = Plots.plot(
        layout=(1, 2),
        dpi=150,
        grid=:xy,
        gridcolor=:black,
        gridwidth=1,
        # background_color=:antiquewhite,
        # aspect_ratio=:equal,
        size=(800, 400),
        plot_title="Пример нескольких графиков"
    )

    Plots.plot!(
        fig[1],
        Time,
        [xx, yy],

```

```

        color=[:red :blue],
        xlabel="Время",
        ylabel="Численность войск",
        label=["Армия X" "Армия Y"]
    )

Plots.plot!(
    fig[2],
    xx,
    yy,
    color=:gray,
    xlabel="Армия X",
    ylabel="Армия Y",
    label="Численность войск"
)
end

```

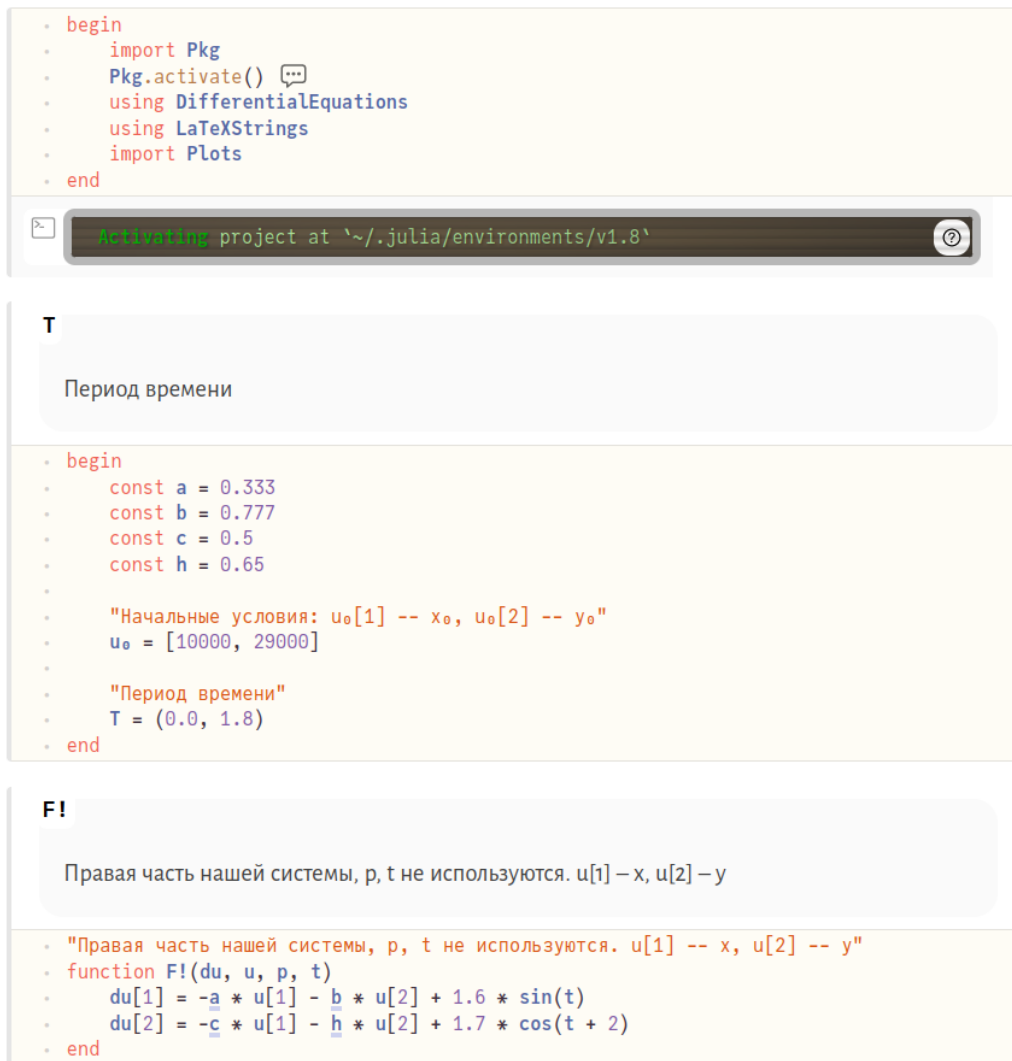


Рис. 4.4: Импорт библиотек. Определение коэффициентов, начальных условий, периода времени, функции для решения ОДУ

```

prob = ODEProblem with uType Vector{Int64} and tType Float64. In-place: true
timespan: (0.0, 1.8)
u0: 2-element Vector{Int64}:
 10000
 29000

```

```

• prob = ODEProblem(F!, u0, T)

```

	timestamp	value1	value2
32	0.31	3144.93	22826.4
33	0.32	2958.05	22663.3
34	0.33	2773.04	22502.2
35	0.34	2589.89	22343.0
36	0.35	2408.58	22185.8
37	0.36	2229.09	22030.5
38	0.37	2051.39	21877.1
39	0.38	1875.46	21725.5
40	0.39	1701.29	21575.8
41	0.4	1528.85	21428.0
42	0.41	1358.12	21282.0
43	0.42	1189.09	21137.7
44	0.43	1021.73	20995.2
45	0.44	856.025	20854.5
46	0.45	691.957	20715.5
47	0.46	529.506	20578.3
48	0.47	368.654	20442.7
49	0.48	209.781	20308.8

```

• sol = solve(prob, saveat=0.01)

```

Рис. 4.5: Формирование проблемы и численный поиск решений

```

▶ [0.0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.11, 0.12, 0.13, 0.14, 0.

• begin
•   const xx = []
•   const yy = []
•   for u in sol.u
•     x, y = u
•
•     if x < 0 || y < 0
•       break
•     end
•
•     push!(xx, x)
•     push!(yy, y)
•   end
•   Time = sol.t[1:size(xx)[1]]
•   Time
• end

```

Пример нескольких графиков

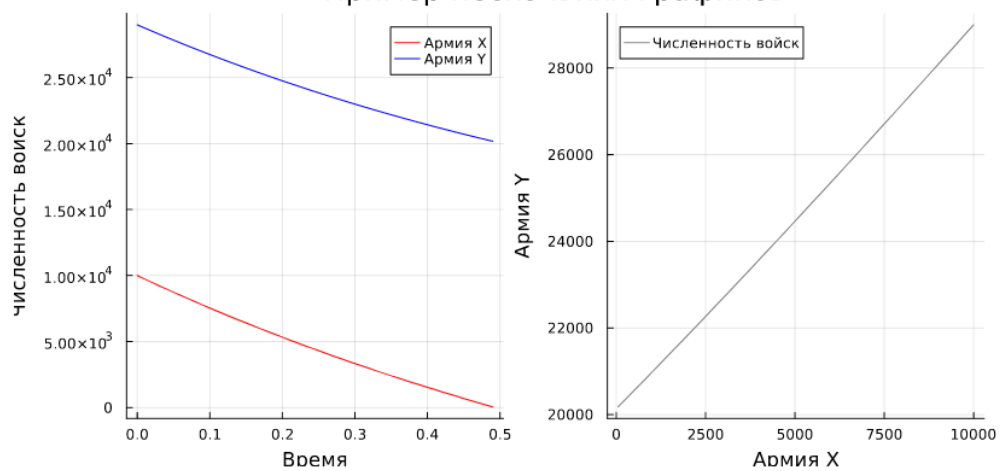


Рис. 4.6: Исключение численности войск меньше нуля

```

begin
    fig = Plots.plot(
        layout=(1, 2),
        dpi=150,
        grid=:xy,
        gridcolor=:black,
        gridwidth=1,
        # background_color=:antiquewhite,
        # aspect_ratio=:equal,
        size=(800, 400),
        plot_title="Пример нескольких графиков"
    )

    Plots.plot!(
        fig[1],
        Time,
        [xx, yy],
        color=[:red :blue],
        xlabel="Время",
        ylabel="Численность войск",
        label=["Армия X" "Армия Y"]
    )

    Plots.plot!(
        fig[2],
        xx,
        yy,
        color=:gray,
        xlabel="Армия X",
        ylabel="Армия Y",
        label="Численность войск"
    )
end

```

Рис. 4.7: Отрисовка графиков

2. Пишем программу, описывающую модель боевых действий с участием регулярных войск и партизанских отрядов. Берем большой период времени для отрисовки графиков, чтобы показать, что на большем промежутке времени численность армии Y будет принимать значения, очень близкие к нулю (рис. 4.8, 4.9).

```

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

```



```

end

begin
    const a = 0.343
    const b = 0.815
    const c = 0.227
    const h = 0.815

    "Начальные условия: u[1] -- x, u[2] -- y"
    u = [10000, 29000]

    "Период времени"
    T = (0.0, 10)
end

"Правая часть нашей системы, p, t не используются. u[1] -- x, u[2] -- y"
function F!(du, u, p, t)
    du[1] = -a * u[1] - b * u[2] + sin(2 * t) + 1
    du[2] = -c * u[1] * u[2] - h * u[2] + cos(10 * t) + 1
end

prob = ODEProblem(F!, u, T)

sol = solve(prob, saveat=0.01)

begin
    const xx = []
    const yy = []
    for u in sol.u
        x, y = u

        if x < 0 || y < 0
            break
        end
    end
end

```

```

end

push!(xx, x)
push!(yy, y)
end
Time = sol.t[1:size(xx)[1]]
Time
end

begin
fig = Plots.plot(
    layout=(1, 2),
    dpi=150,
    grid=:xy,
    gridcolor=:black,
    gridwidth=1,
    # background_color=:antiquewhite,
    # aspect_ratio=:equal,
    size=(800, 400),
    plot_title="Пример нескольких графиков"
)

Plots.plot!(
    fig[1],
    Time,
    [xx, yy],
    color=[:red :blue],
    xlabel="Время",
    ylabel="Численность войск",
    label=["Армия X" "Армия Y"]

```

```

)

Plots.plot!(
    fig[2],
    xx,
    yy,
    color=:gray,
    xlabel="Армия X",
    ylabel="Армия Y",
    label="Численность войск"
)
end

```

T

Период времени

```

• begin
•     const a = 0.343
•     const b = 0.815
•     const c = 0.227
•     const h = 0.815
•
•     "Начальные условия: u₀[1] -- x₀, u₀[2] -- y₀"
•     u₀ = [10000, 29000]
•
•     "Период времени"
•     T = (0.0, 10)
• end

```

F!

Правая часть нашей системы, p, t не используются. u[1] – x, u[2] – y

```

• "Правая часть нашей системы, p, t не используются. u[1] -- x, u[2] -- y"
• function F!(du, u, p, t)
•     du[1] = -a * u[1] - b * u[2] + sin(2 * t) + 1
•     du[2] = -c * u[1] * u[2] - h * u[2] + cos(10 * t) + 1
• end

```

Рис. 4.8: Измененные коэффициенты, период времени

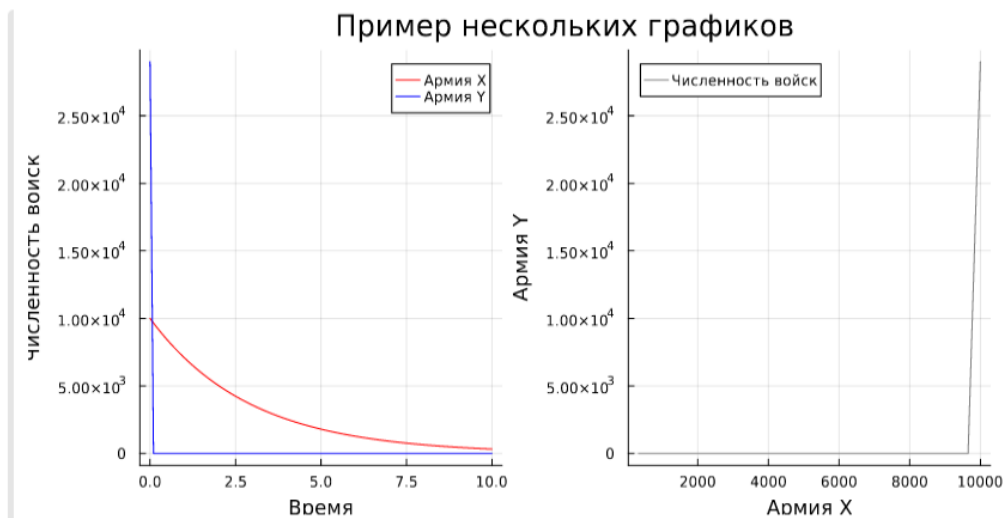


Рис. 4.9: Измененные графики

4.2.2 Julia

1. Пишем программу, описывающую модель боевых действий между регулярными войсками (рис. 4.10).

```
using Plots
```

```
using DifferentialEquations
```

```
const a = 0.333
```

```
const b = 0.777
```

```
const c = 0.5
```

```
const h = 0.65
```

```
"Начальные условия: u[1] - x, u[2] - y"
```

```
u = [10000, 29000]
```

```
"Период времени"
```

```
T = (0.0, 1.8)
```

```
function F!(du, u, p, t)
    du[1] = -a * u[1] - b * u[2] + 1.6 * sin(t)
    du[2] = -c * u[1] - h * u[2] + 1.7 * cos(t + 2)
end
```

```
prob = ODEProblem(F!, u0, T)
sol = solve(prob, saveat=0.01)
```

```
const xx = []
const yy = []
for u in sol.u
    x, y = u
    if x < 0 || y < 0
        break
    end
    push!(xx, x)
    push!(yy, y)
end
time = sol.t[1:size(xx)[1]]
```

```
plt = Plots.plot(
    layout=(1, 2),
    dpi=300,
    grid=:xy,
    gridcolor=:black,
    gridwidth=1,
```

```

        size=(800, 400),
        plot_title="Модель военных действий"
    )

    plot!(plt[1], time, [xx, yy], color=:red :blue, xlabel="Время", ylabel="Численность войск", label=["Армия X" "Армия Y"])
    plot!(plt[2], xx, yy, color=:gray, xlabel="Армия X", ylabel="Армия Y", label="Численность войск")
    savefig(plt, "lab3_1")

```

```

1  using Plots
2  using DifferentialEquations
3
4
5  const a = 0.333
6  const b = 0.777
7  const c = 0.5
8  const h = 0.65
9
10 "Начальные условия: u0[1] - x0, u0[2] - y0"
11 u0 = [10000, 29000]
12
13 "Период времени"
14 T = (0.0, 1.8)
15
16 function F!(du, u, p, t)
17     du[1] = -a * u[1] - b * u[2] + 1.6 * sin(t)
18     du[2] = -c * u[1] - h * u[2] + 1.7 * cos(t + 2)
19 end
20
21
22 prob = ODEProblem(F!, u0, T)
23 sol = solve(prob, saveat=0.01)
24
25 const xx = []
26 const yy = []
27 for u in sol.u
28     x, y = u
29     if x < 0 || y < 0
30         break
31     end
32     push!(xx, x)
33     push!(yy, y)
34 end
35 time = sol.t[1:size(xx)[1]]
36
37 plt = Plots.plot(
38     layout=(1, 2),
39     dpi=300,
40     grid=:xy,
41     gridcolor=:black,
42     gridwidth=1,
43     size=(800, 400),
44     plot_title="Модель военных действий"
45 )
46
47 plot!(plt[1], time, [xx, yy], color=:red :blue, xlabel="Время", ylabel="Численность войск", label=["Армия X" "Армия Y"])
48 plot!(plt[2], xx, yy, color=:gray, xlabel="Армия X", ylabel="Армия Y", label="Численность войск")
49 savefig(plt, "lab3")
50

```

Рис. 4.10: Код повторяет код из Pluto.jl

2. Получаем данные графики (рис. 4.11).

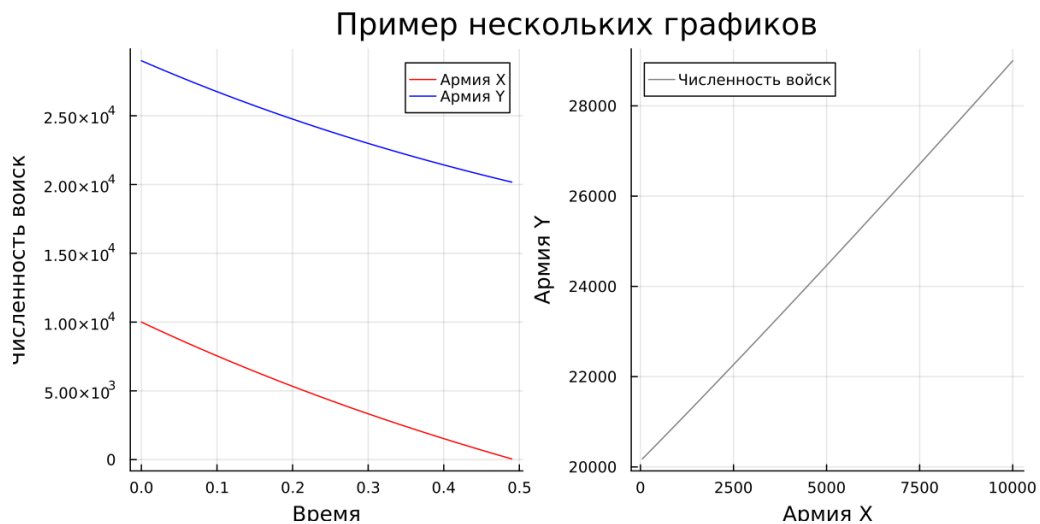


Рис. 4.11: Результат работы программы

- Пишем программу, описывающую модель боевых действий с участием регулярных войск и партизанских отрядов (рис. 4.12).

```
using Plots
```

```
using DifferentialEquations
```

```
const a = 0.343
```

```
const b = 0.815
```

```
const c = 0.227
```

```
const h = 0.815
```

```
"Начальные условия:  $u_x[1]$  -  $x$ ,  $u_x[2]$  -  $y$ "
```

```
 $u_x = [10000, 29000]$ 
```

```
"Период времени"
```

```
 $T = (0.0, 0.3)$ 
```

```

function F!(du, u, p, t)
    du[1] = -a * u[1] - b * u[2] + sin(2 * t) + 1
    du[2] = -c * u[1] * u[2] - h * u[2] + cos(10 * t) + 1
end

```

```

prob = ODEProblem(F!, u0, T)
sol = solve(prob, saveat=0.01)

```

```

const xx = []
const yy = []
for u in sol.u
    x, y = u
    if x < 0 || y < 0
        break
    end
    push!(xx, x)
    push!(yy, y)
end
time = sol.t[1:size(xx)[1]]

```

```

plt = Plots.plot(
    layout=(1, 2),
    dpi=300,
    grid=:xy,
    gridcolor=:black,
    gridwidth=1,
    size=(800, 400),
    plot_title="Модель военных действий"
)

```


)

```
plot!(plt[1], time, [xx, yy], color=[:red :blue], xlabel="Время", ylabel="Чи  
plot!(plt[2], xx, yy, color=:gray], xlabel="Армия X", ylabel="Армия Y",  
savefig(plt, "lab3_2")
```

```
5  const a = 0.343  
6  const b = 0.815  
7  const c = 0.227  
8  const h = 0.815  
9  
10 "Начальные условия: u_0[1] - x_0, u_0[2] - y_0"  
11 u_0 = [10000, 29000]  
12  
13 "Период времени"  
14 T = (0.0, 0.05)  
15  
16 function F!(du, u, p, t)  
17     du[1] = -a * u[1] - b * u[2] + sin(2 * t) + 1  
18     du[2] = -c * u[1] * u[2] - h * u[2] + cos(10 * t) + 1  
19 end
```

Рис. 4.12: Измененные строчки кода по сравнению с предыдущим листингом

4. Получаем данные графики (рис. 4.13).

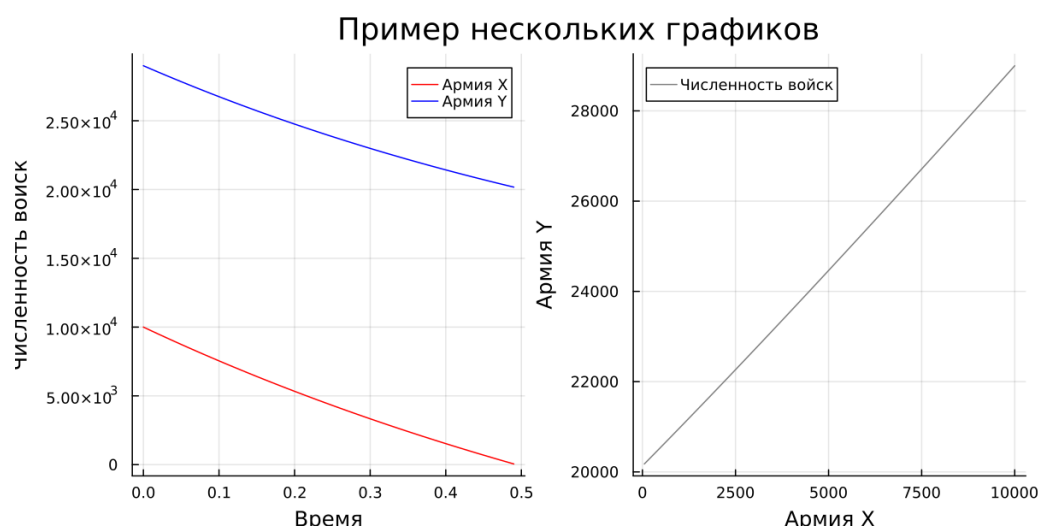


Рис. 4.13: Результат работы программы

4.2.3 OpenModelica

1. Пишем программу, описывающую модель боевых действий между регулярными войсками (рис. 4.14).

```

model lab3_1
  constant Real a = 0.333;
  constant Real b = 0.777;
  constant Real c = 0.5;
  constant Real h = 0.65;
  Real t = time;
  Real x;
  Real y;
initial equation
  x = 10000;
  y = 29000;
equation
  der(x) = -a*x - b*y + 1.6 * sin(t);

```

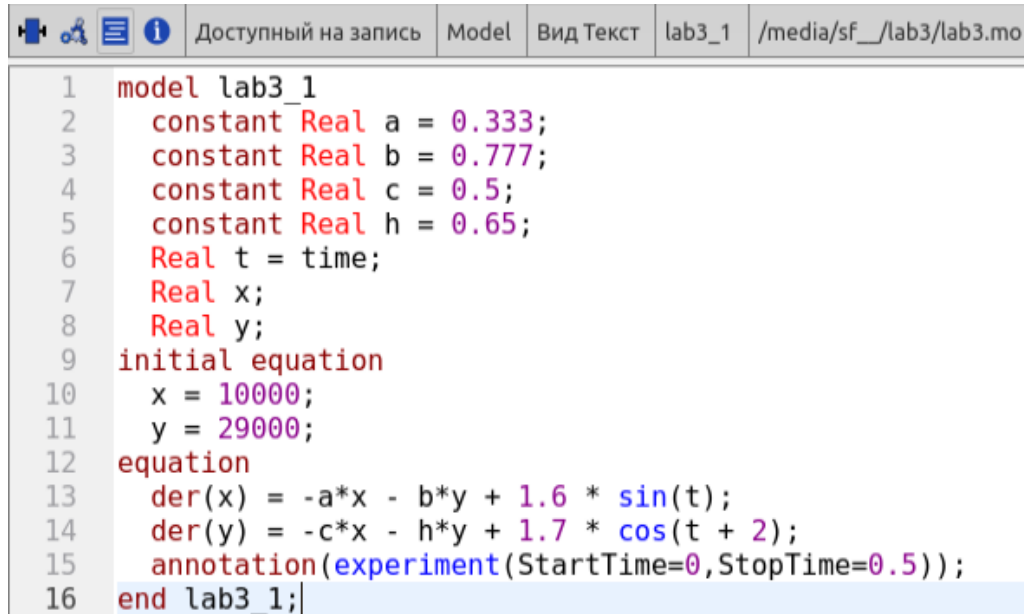
```

der(y) = -c*x - h*y + 1.7 * cos(t + 2);

annotation(experiment(StartTime=0,StopTime=0.5));

end lab3_1;

```



```

1  model lab3_1
2      constant Real a = 0.333;
3      constant Real b = 0.777;
4      constant Real c = 0.5;
5      constant Real h = 0.65;
6      Real t = time;
7      Real x;
8      Real y;
9      initial equation
10     x = 10000;
11     y = 29000;
12     equation
13     der(x) = -a*x - b*y + 1.6 * sin(t);
14     der(y) = -c*x - h*y + 1.7 * cos(t + 2);
15     annotation(experiment(StartTime=0,StopTime=0.5));
16 end lab3_1;

```

Рис. 4.14: Задание констант, изменяющихся переменных, начальных условий, системы уравнений, времени моделирования

2. Получаем данные графики (рис. 4.15, 4.16).

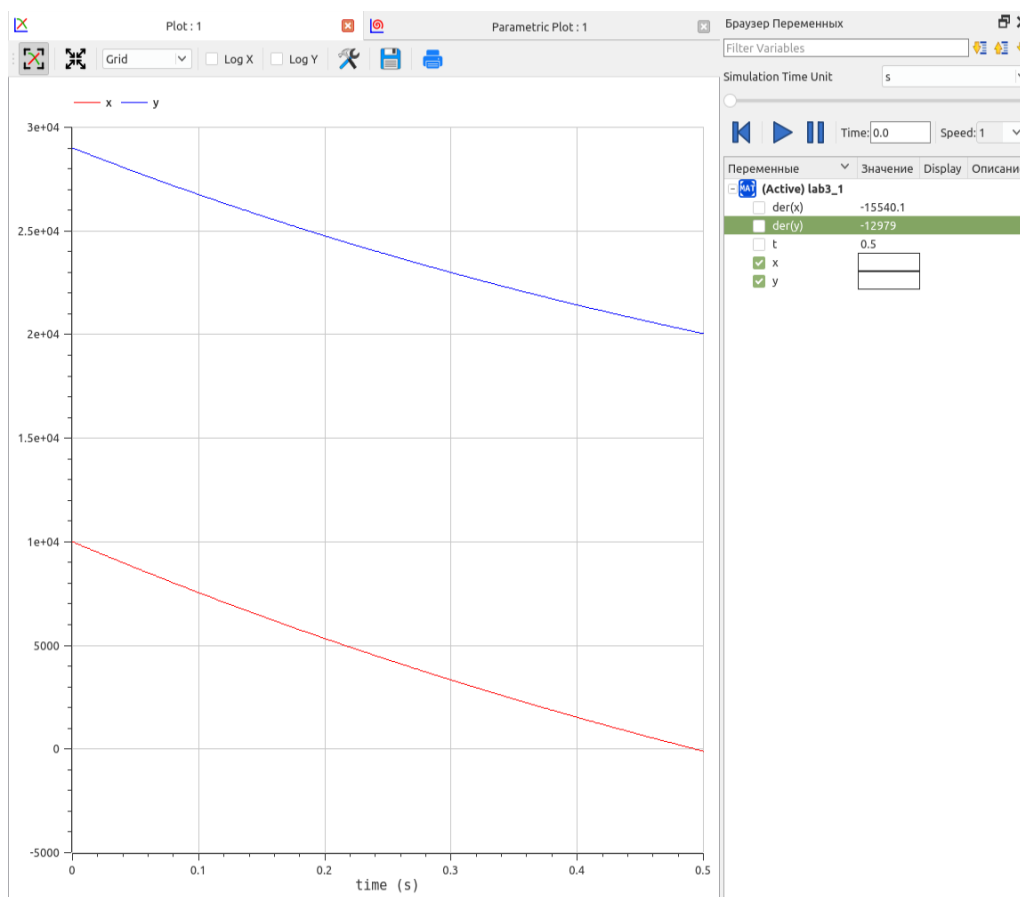


Рис. 4.15: График зависимости численности армии от времени

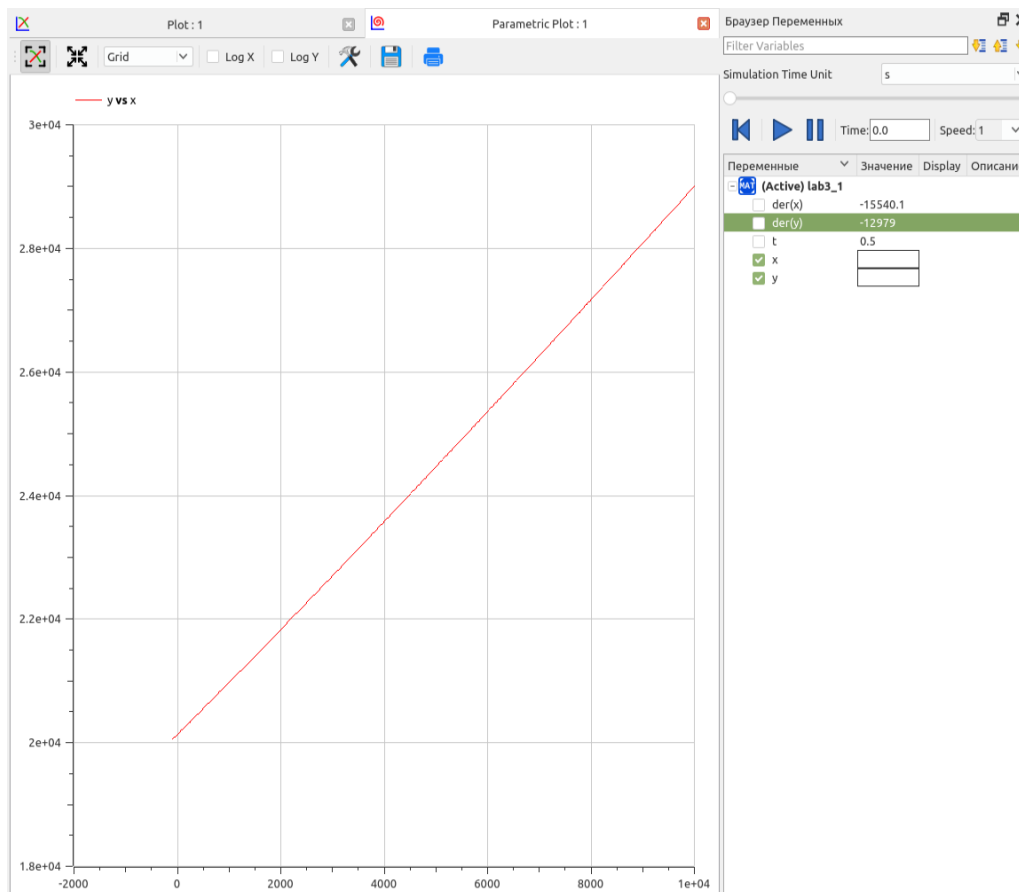


Рис. 4.16: График зависимости численности армии X от армии Y

3. Пишем программу, описывающую модель боевых действий с участием регулярных войск и партизанских отрядов (рис. 4.17).

```

model lab3_1
    constant Real a = 0.343;
    constant Real b = 0.815;
    constant Real c = 0.227;
    constant Real h = 0.815;
    Real t = time;
    Real x;
    Real y;

```

```

initial equation
    x = 10000;
    y = 29000;
equation
    der(x) = -a*x - b*y + sin(2 * t) + 1;
    der(y) = -c*x*y - h*y + cos(10 * t) + 1;
    annotation(experiment(StartTime=0,StopTime=0.01));
end lab3_1;

```

```

1  model lab3_1
2      constant Real a = 0.343;
3      constant Real b = 0.815;
4      constant Real c = 0.227;
5      constant Real h = 0.815;
6      Real t = time;
7      Real x;
8      Real y;
9  initial equation
10     x = 10000;
11     y = 29000;
12  equation
13     der(x) = -a*x - b*y + sin(2 * t) + 1;
14     der(y) = -c*x*y - h*y + cos(10 * t) + 1;
15     annotation(experiment(StartTime=0,StopTime=0.01));
16  end lab3_1;

```

Рис. 4.17: Задание иных констант, системы уравнений, времени моделирования

4. Получаем данные графики (рис. 4.18, 4.19).

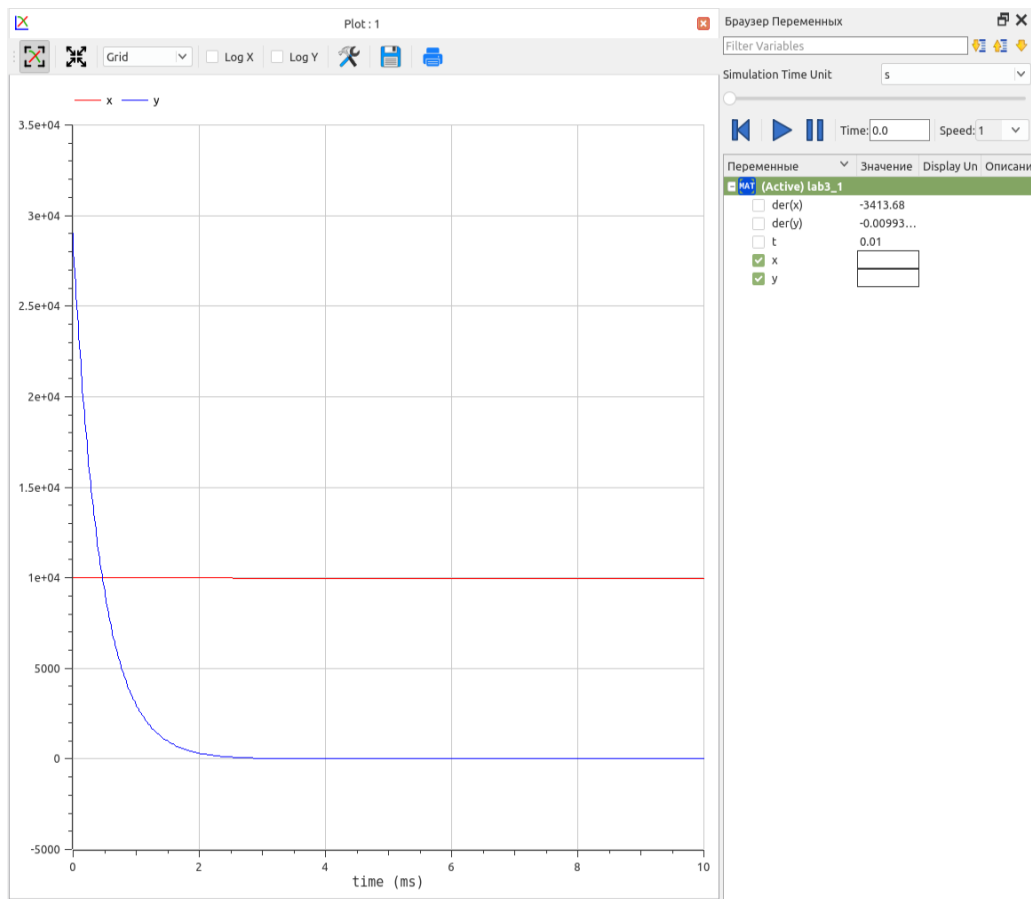


Рис. 4.18: График зависимости численности армии от времени

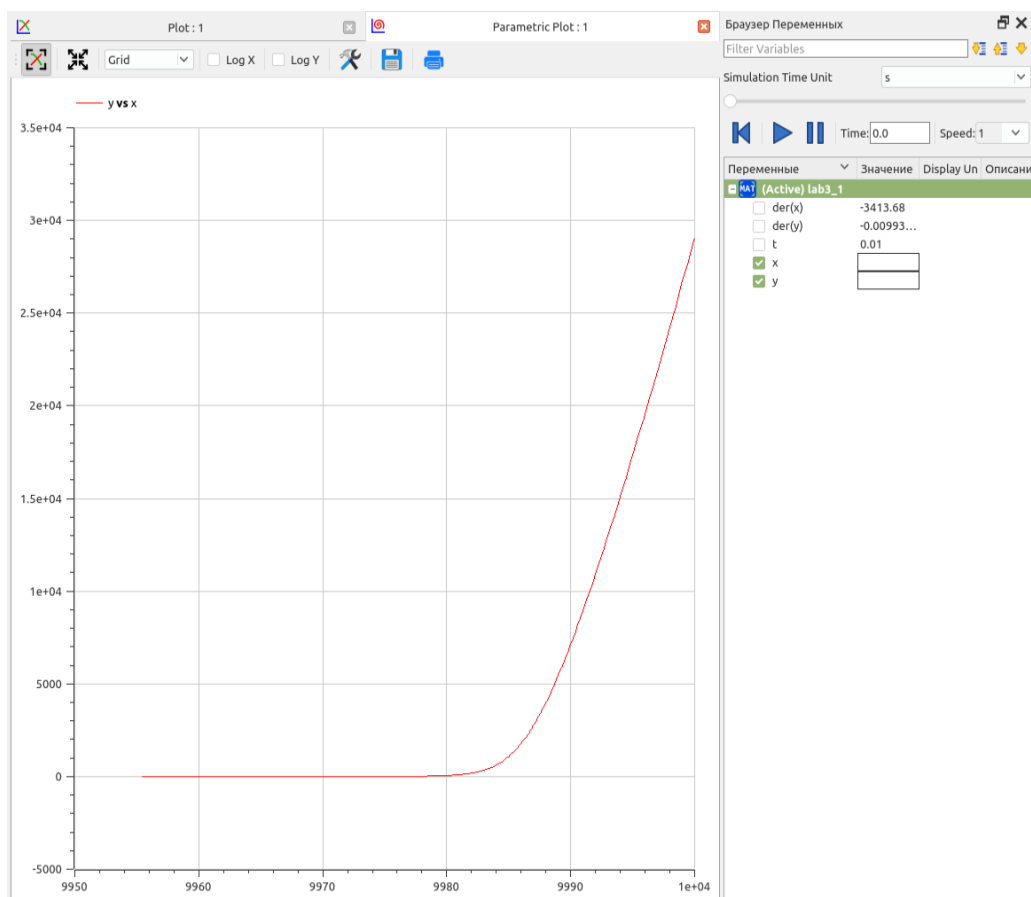


Рис. 4.19: График зависимости численности армии X от армии Y

5 Анализ результатов

На примере построения математической модели ведения боевых действий по принципу модели Ланчестера мы можем проанализировать различия языка программирования Julia и языка моделирования Modelica. Безусловно, при построении математических моделей других задач, полученный опыт может различаться. Однако, применительно к данной конкретной задаче было замечено, что суммарная длина кода на языке моделирования Modelica в разы меньше, чем длина такового на языке программирования Julia. Читабельность кода также лучше в Modelica. Скорость моделирования и возможность гибкой настройки графиков тоже выше у языка моделирования Modelica. Скорость отрисовки графиков встроенными средствами OpenModelica также выше (имеется в виду редактор OMEdit), однако при использовании Pluto.jl в дополнении к языку программирования Julia достигается соизмеримо быстрая скорость отрисовки.

С другой стороны, Julia является более простым языком в освоении на начальных этапах, т.к. синтаксис данного языка очень похож на таковой у популярного в данный момент языка программирования Python, что делает Julia более простым для «скорого» освоения программистом, знакомым с интерпретируемым языком Python.

6 Выводы

Ознакомился с базовым функционалом интерактивного блокнота Pluto. Рассмотрел простейшую модель боевых действий — модель Ланчестера. Построил данную математическую модель при помощи языка программирования Julia, интерактивного блокнота Pluto, языка моделирования Modelica и программного обеспечения OpenModelica.

Список литературы

1. Simple, reactive programming environment for Julia [Электронный ресурс]. Pluto.jl. URL: <https://plutojl.org/>.
2. Lightweight reactive notebooks for Julia [Электронный ресурс]. JuliaHub. URL: <https://juliahub.com/ui/Packages/Pluto/OJqMt/0.7.5>.
3. Introduction to Julia for CATAM [Электронный ресурс]. sje30. URL: <https://sje30.github.io/catam-julia/intro/julia-manual.html>.
4. Lanchester's laws [Электронный ресурс]. Military Wiki. URL: https://military-history.fandom.com/wiki/Lanchester%27s_laws.
5. Lanchester's laws [Электронный ресурс]. Wikimedia Foundation, Inc., 2023. URL: <https://en.wikipedia.org/wiki/OpenModelica>.