

Лабораторная работа №3

Модель боевых действий Ланчестера

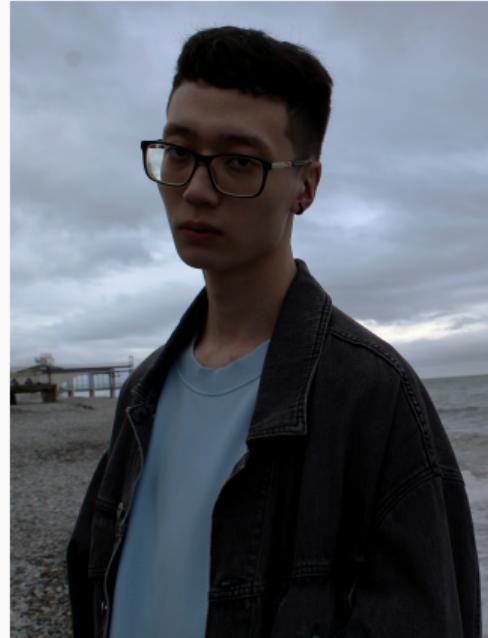
Ким М. А.

25 февраля 2023

Российский университет дружбы народов, Москва, Россия

Информация

- Ким Михаил Алексеевич
- студент уч. группы НФИбд-01-20
- Российский университет дружбы народов
- 1032201664@pfur.ru
- <https://github.com/exmanka>



Вводная часть

- Необходимость навыков моделирования реальных математических задач, построение графиков.

Объект и предмет исследования

- Язык программирования Julia
- Язык моделирования Modelica
- Модель Ланчестера

- Начать знакомство с интерактивным блокнотом Pluto.jl.
- Продолжить знакомство с функционалом языков Julia и OpenModelica.
- Создать математическую модель боевых действий Ланчестера с помощью данных языков.

Материалы и методы

- Языки:
 - Julia
 - Modelica

Процесс выполнения работы

Подготовка системы для работы

Установка интерактивного блокнота Pluto.jl

```
(@v1.8) pkg> add Pluto
```

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

Pluto.jl  /media/sf__/_/lab3/ode.jl

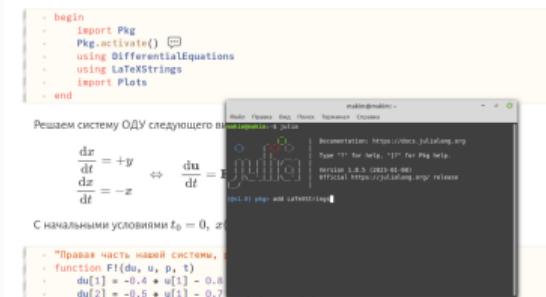
```
begin
    import Pkg
    Pkg.activate() ⚡
    using DifferentialEquations
    using LaTeXStrings
    import Plots
end
```

Решаем систему ОДУ следующего вида

$$\begin{aligned} \frac{dx}{dt} &= +y \\ \frac{dy}{dt} &= -x \end{aligned} \quad \Leftrightarrow \quad \frac{du}{dt} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} u$$

С начальными условиями $t_0 = 0$, $x_0 = 1$, $y_0 = 0$.

```
• "Правая часть нашей системы, "
• function F!(du, u, p, t)
    du[1] = -0.4 * u[1] - 0.8
    du[2] = -0.5 * u[1] - 0.7
```



Формулировка задания

Формулировка задания 1

Между страной X и страной Y идет война. Численность состава войск исчисляется от начала войны, и являются временными функциями $x(t)$ и $y(t)$. В начальный момент времени страна X имеет армию численностью **10000** человек, а в распоряжении страны Y армия численностью в **29000** человек. Для упрощения модели считаем, что коэффициенты a , b , c , h постоянны. Также считаем $P(t)$ и $Q(t)$ непрерывные функции.

Формулировка задания 2

Постройте графики изменения численности войск армии и армии для следующих случаев:

1. Модель боевых действий между регулярными войсками

$$\begin{cases} \frac{dx}{dt} = -0.333x(t) - 0.77y(t) + 1.6 \sin(t) \\ \frac{dy}{dt} = -0.5x(t) - 0.65y(t) + 1.7 \cos(t + 2) \end{cases}$$

Формулировка задания 3

2. Модель ведение боевых действий с участием регулярных войск и партизанских отрядов

$$\begin{cases} \frac{dx}{dt} = -0.343x(t) - 0.815y(t) + \sin(2t) + 1 \\ \frac{dy}{dt} = -0.227x(t)y(t) - 0.815y(t) + \cos(10t) + 1 \end{cases}$$

Написание программ

Код на Pluto.jl первой модели (1)

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

Activating project at `~/.julia/environments/v1.8`

T

Период времени

```
. begin
.     const a = 0.333
.     const b = 0.777
.     const c = 0.5
.     const h = 0.65
.
.     "Начальные условия: u₀[1] -- x₀, u₀[2] -- y₀"
.     u₀ = [10000, 29000]
.
.     "Период времени"
.     T = (0.0, 1.8)
. 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] + 1.6 * sin(t)
.     du[2] = -c * u[1] - h * u[2] + 1.7 * cos(t + 2)
. end
```

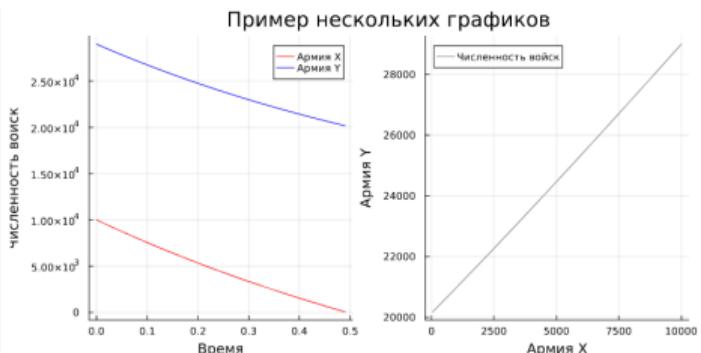
```
prob = ODEProblem with uType Vector{Int64} and tType Float64. In-place: true
timespan: (0.0, 1.8)
u₀: 2-element Vector{Int64}:
 10000
29000
prob = ODEProblem(F!, u₀, T)
```

	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	200.704	20300.0

```
. sol = solve(prob, saveat=0.01)
```

Код на Pluto.jl первой модели (2)

```
▶ [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
```



```
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
```

Код на Pluto.jl второй модели (изменения)

T

Период времени

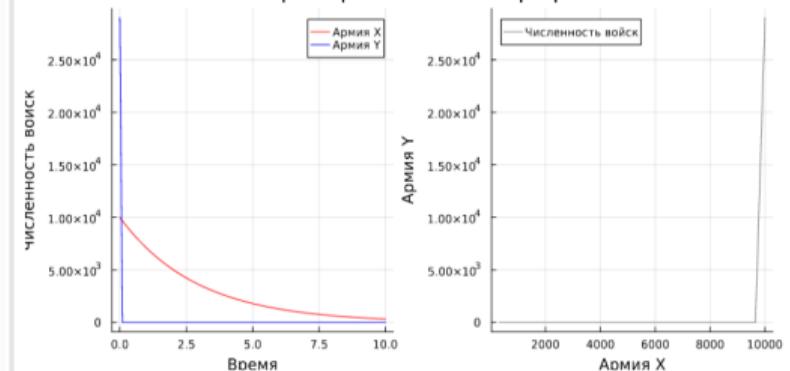
```
. begin
  const a = 0.343
  const b = 0.815
  const c = 0.227
  const h = 0.815
  "Начальные условия: u₀[1] -- x₀, u₀[2] -- y₀"
  u₀ = [10000, 29000]
  "Период времени"
  T = (0.0, 10)
end
```

F!

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

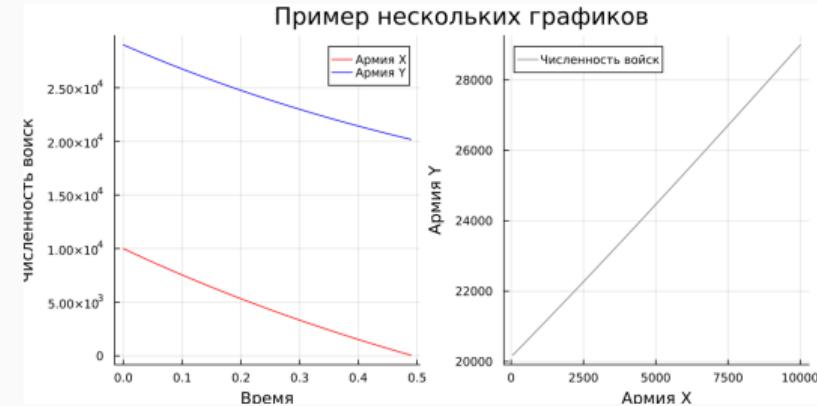
```
"Правая часть нашей системы, p, t не используются. u[1] -- x, u[2] -- y"
function F!(du, u, p, t)
  du[1] = -a * u[1] - b * u[2] + sin(2 * t) + 1
  du[2] = -c * u[1] * u[2] - h * u[2] + cos(10 * t) + 1
end
```

Пример нескольких графиков



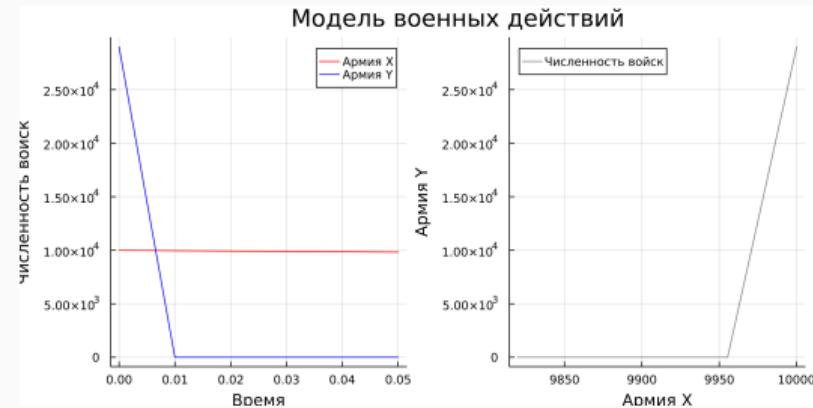
Код на Julia первой модели и результат выполнения

```
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 = {0.8, 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], colors[: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")
```



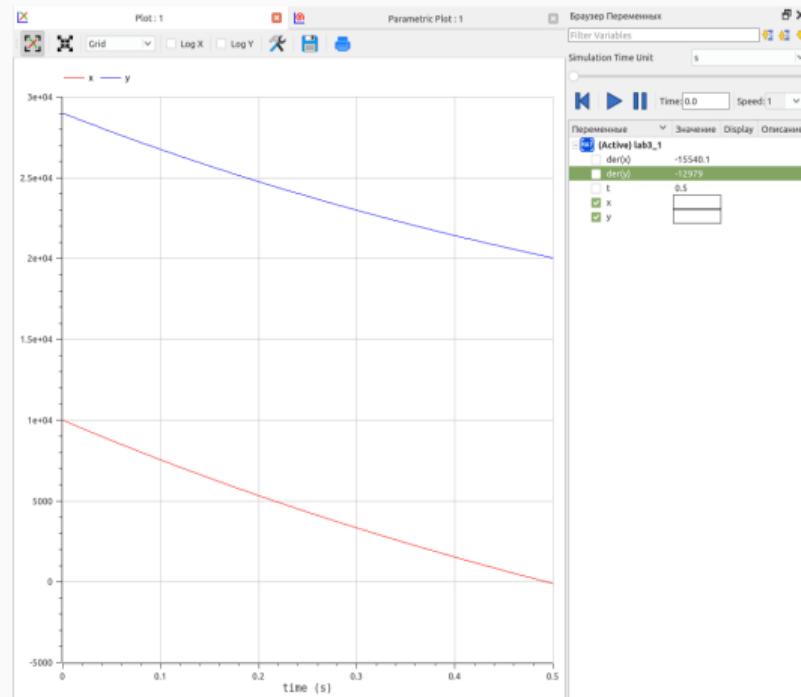
Код на Julia второй модели (изменения) и результат выполнения

```
5  const a = 0.343
6  const b = 0.815
7  const c = 0.227
8  const h = 0.815
9
10 "Начальные условия: u₀[1] - x₀, u₀[2] - y₀"
11 u₀ = [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
```

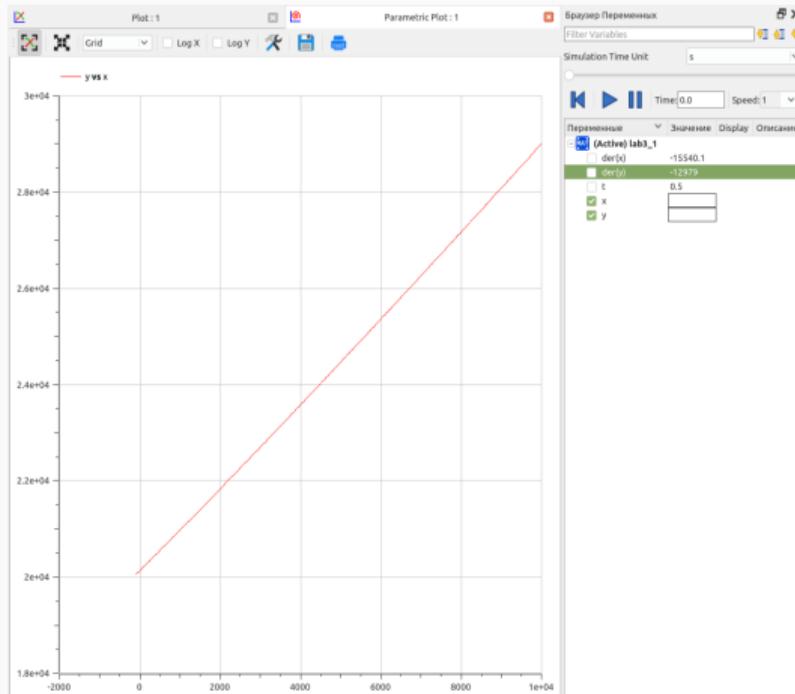


Код на Modelica первой модели и результат выполнения (1)

```
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;
```

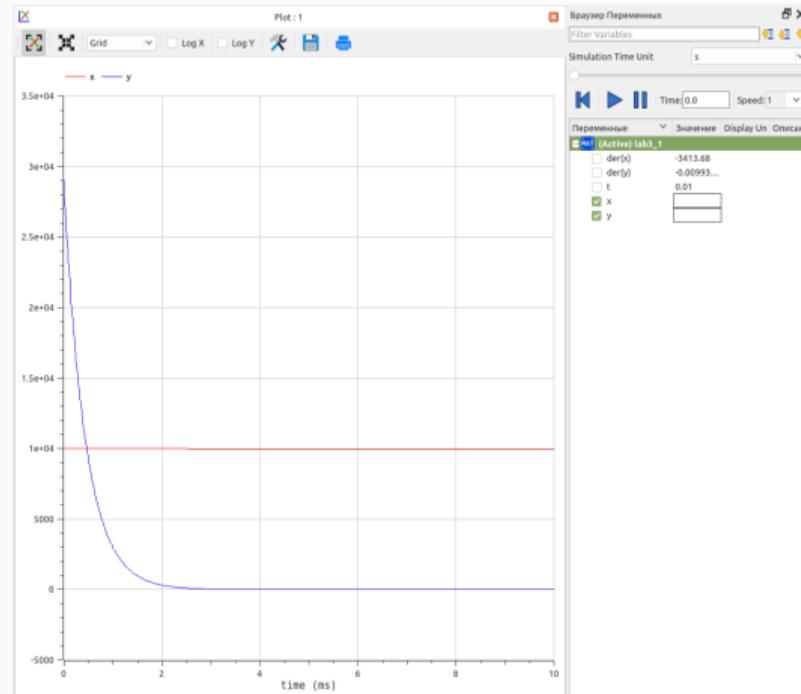


Результат выполнения (2)

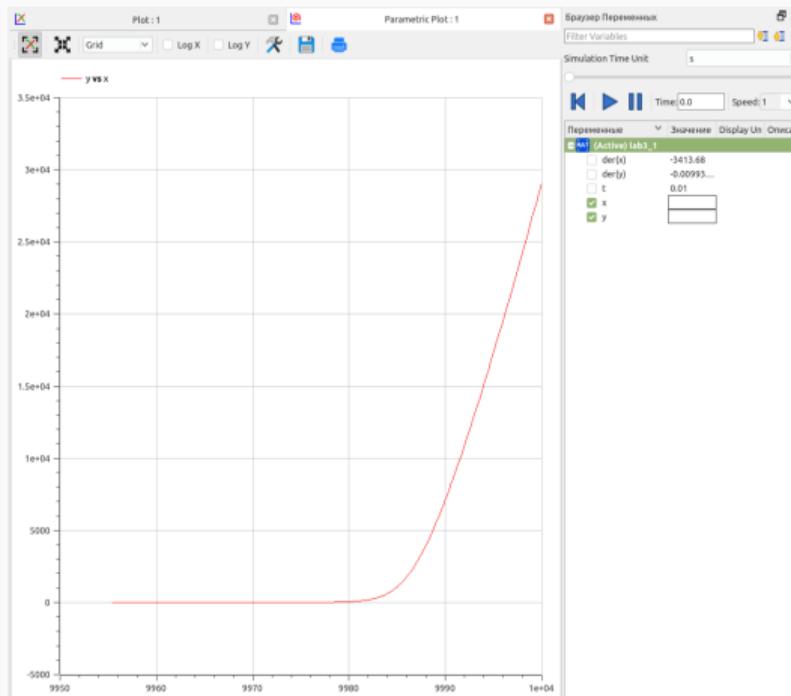


Код на Modelica второй модели и результат выполнения (1)

```
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;
```



Результат выполнения (2)



Результаты

Результаты

- Написаны программы на языках Julia и Modelica, описывающих модель боевых действий Ланчестера.
- Построены графики изменения численности армий в соответствии с поставленными задачами.

Вывод

Начал знакомство с интерактивным блокнотом Pluto.jl. Продолжил знакомство с функционалом языка программирования Julia и языка моделирования Modelica. Используя эти средства, построил математическую модель, представляющую собой модель боевых действий Ланчестера.