Отчет по лабораторной работе №2

по дисциплине: Математическое моделирование

Ким Михаил Алексеевич

Содержание

# 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. [1](#fig:01)).

* julia  
  ]  
  add Pluto

* Figure 1: Установка пакета Pluto
* Figure 1: Установка пакета Pluto

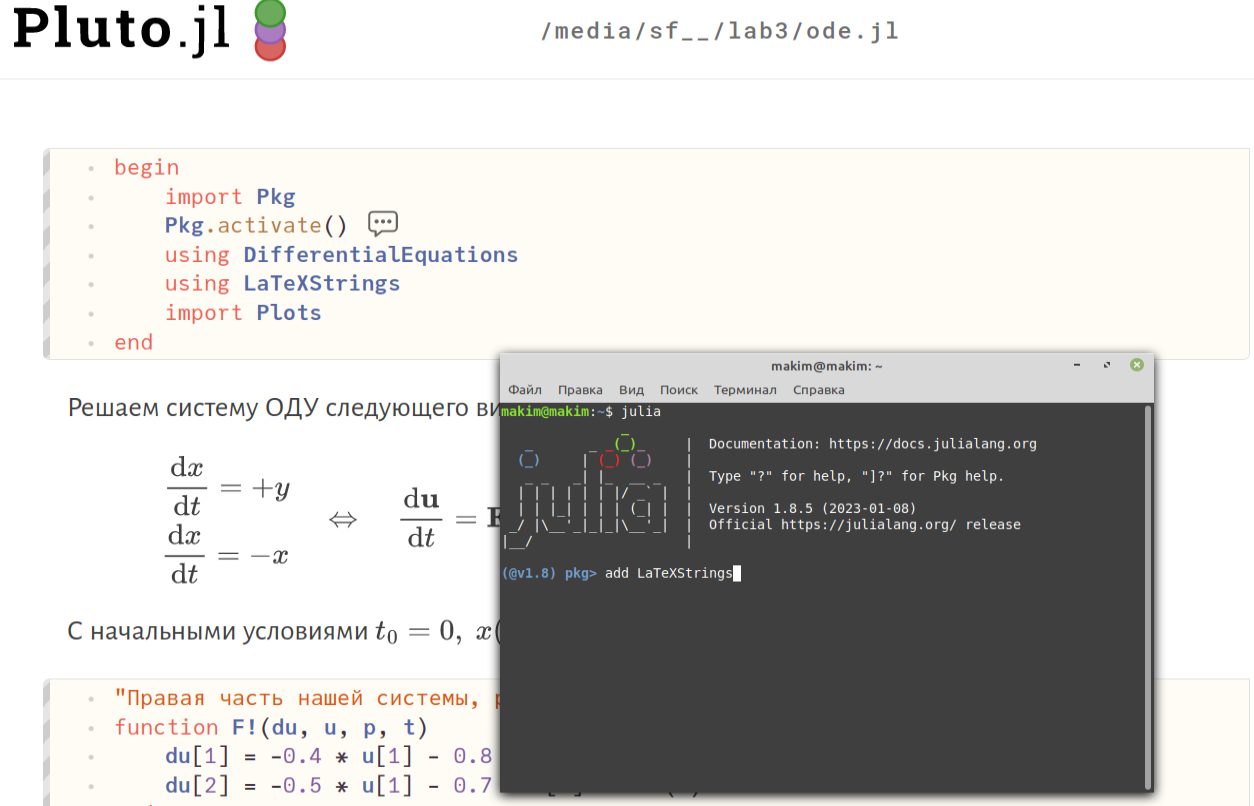
1. Запускаем интерактивный блокнот Pluto (рис. [2](#fig:02)).

* import Pluto; Pluto.run()

* Figure 2: Запуск Pluto
* Figure 2: Запуск Pluto

1. Устанавливаем дополнительно пакет **LaTeXStrings** для корректной отрисовки графиков внутри Pluto (рис. [3](#fig:03)).

* julia  
  ]  
  add LaTeXStrings

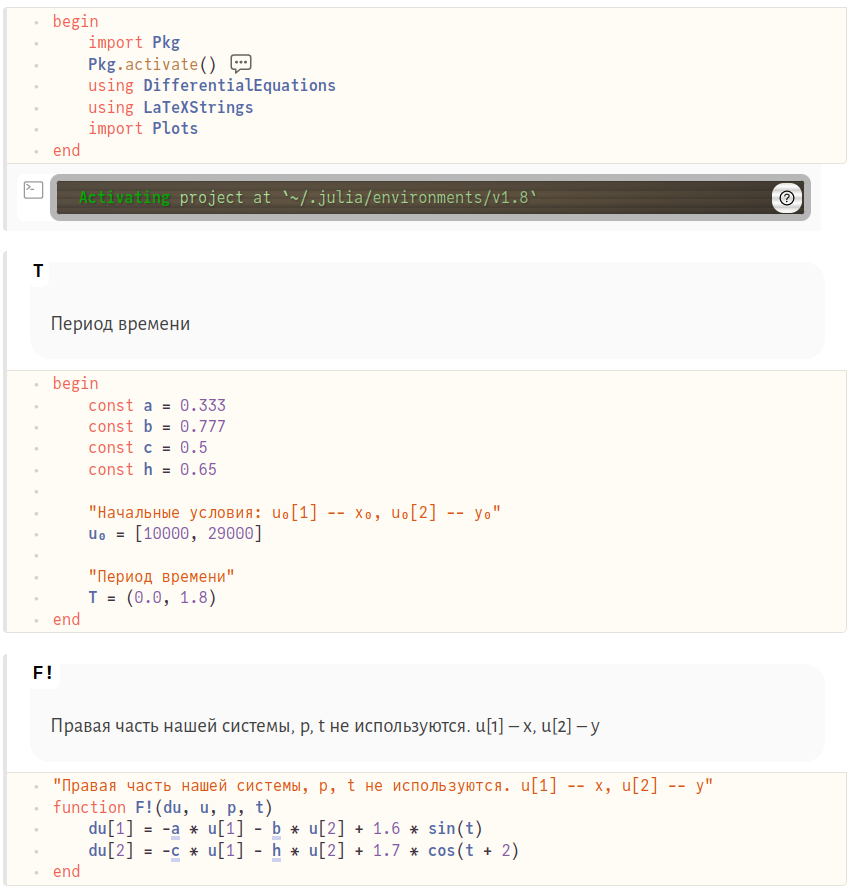
* 
* Figure 3: Установка пакета LaTeXStrings

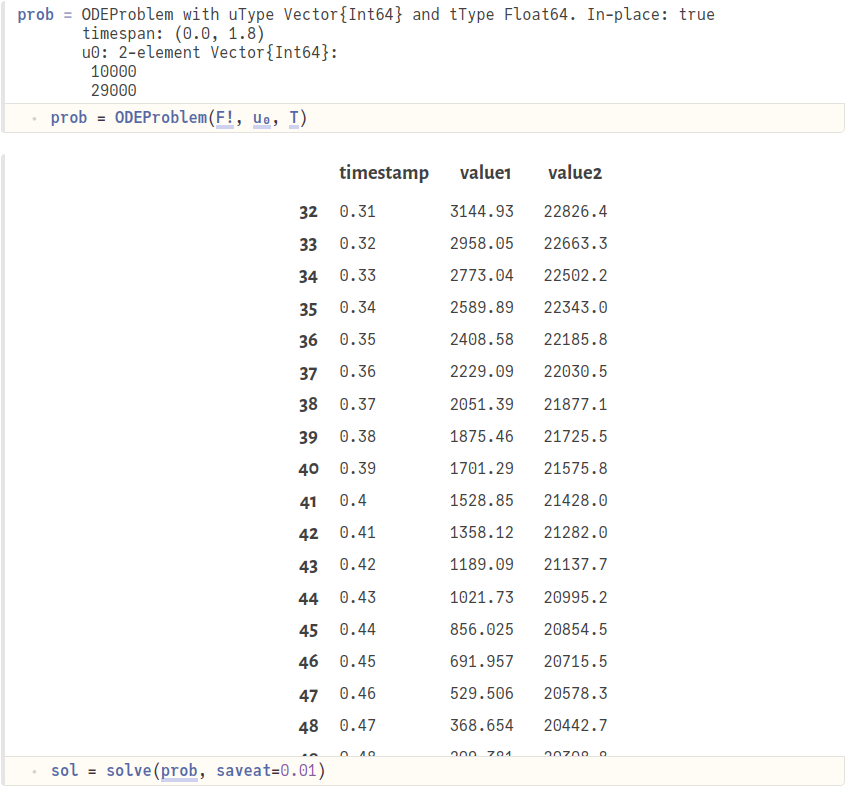
## 4.2 Выполнение лабораторной работы

### 4.2.1 Pluto.jl

1. Пишем программу, описывающую модель боевых действий между регулярными войсками (рис. [4](#fig:04), [5](#fig:05), [6](#fig:06), [7](#fig:07)).

* 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: u₀[1] -- x₀, u₀[2] -- y₀"  
   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

* 
* Figure 4: Импорт библиотек. Определение коэффициентов, начальных условий, периода времени, функции для решения ОДУ

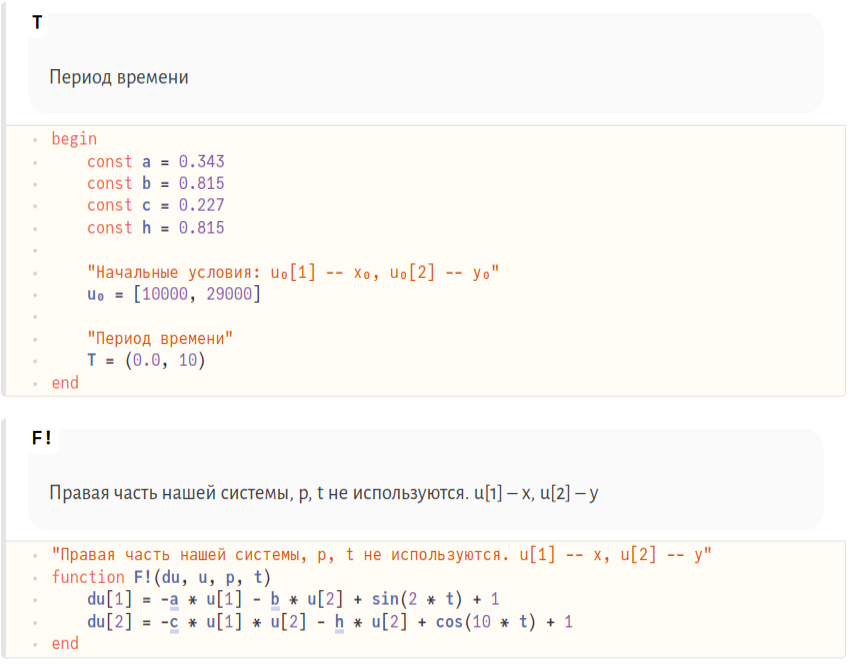
* 
* Figure 5: Формирование проблемы и численный поиск решений

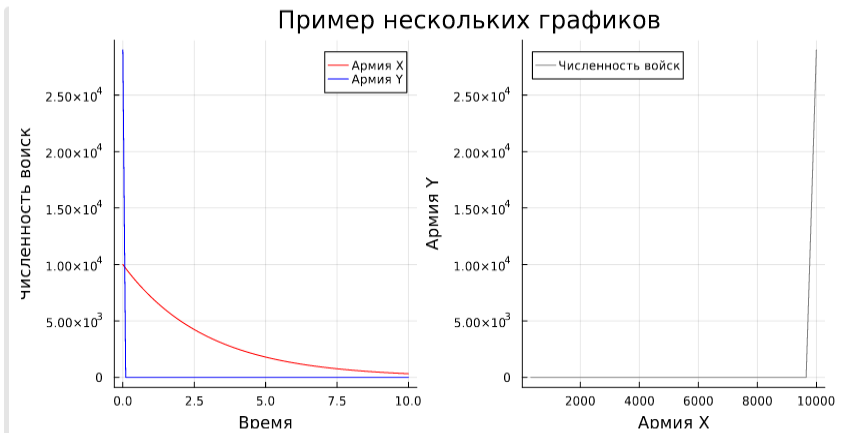
* 
* Figure 6: Исключение численности войск меньше нуля

* 
* Figure 7: Отрисовка графиков

1. 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. [8](#fig:08), [9](#fig:09)).

* 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: u₀[1] -- x₀, u₀[2] -- y₀"  
   u₀ = [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!, 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

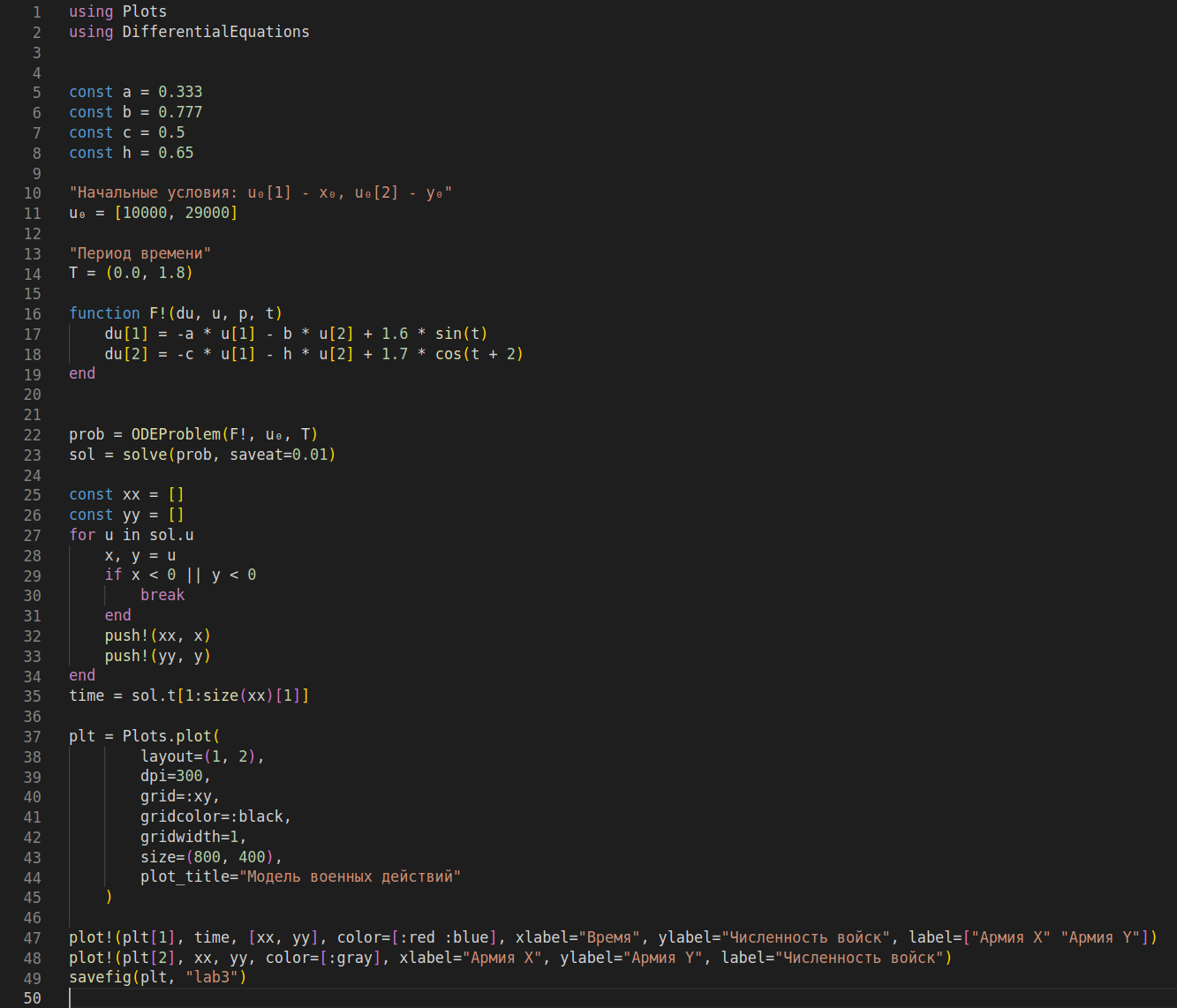
* 
* Figure 8: Измененные коэффициенты, период времени

* 
* Figure 9: Измененные графики

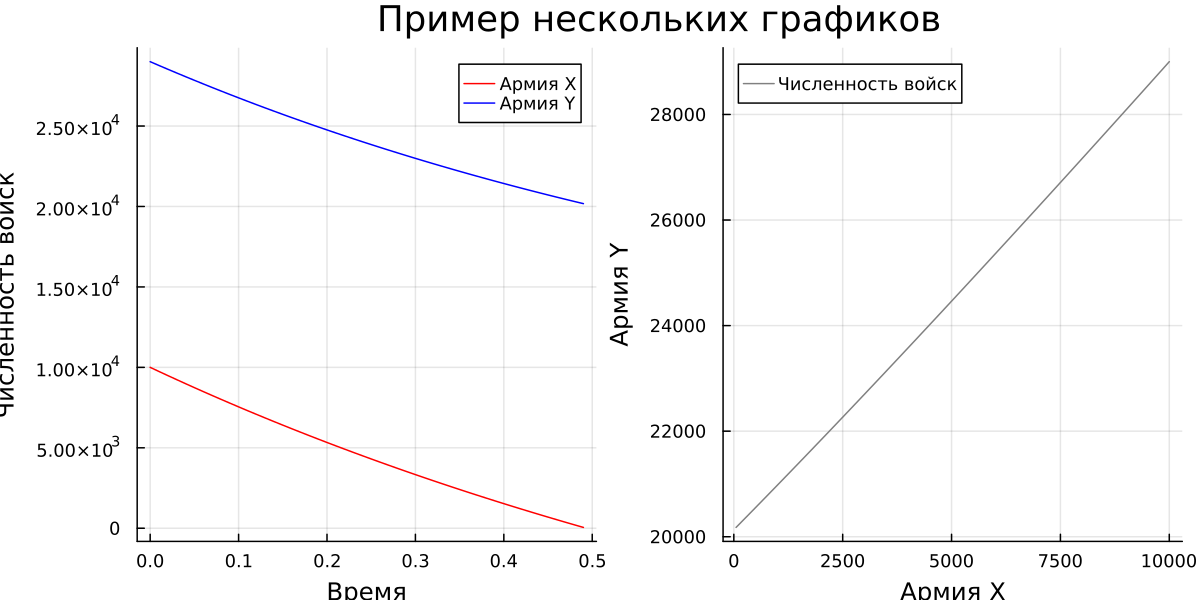
### 4.2.2 Julia

1. Пишем программу, описывающую модель боевых действий между регулярными войсками (рис. [10](#fig:10)).

* using Plots  
  using DifferentialEquations  
    
    
  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)  
    
  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")

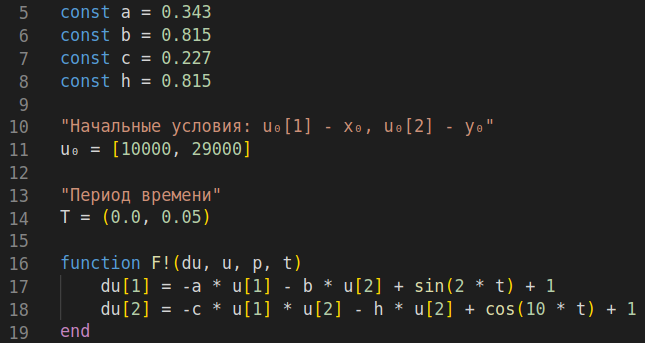
* 
* Figure 10: Код повторяет код из Pluto.jl

1. Получаем данные графики (рис. [11](#fig:11)).

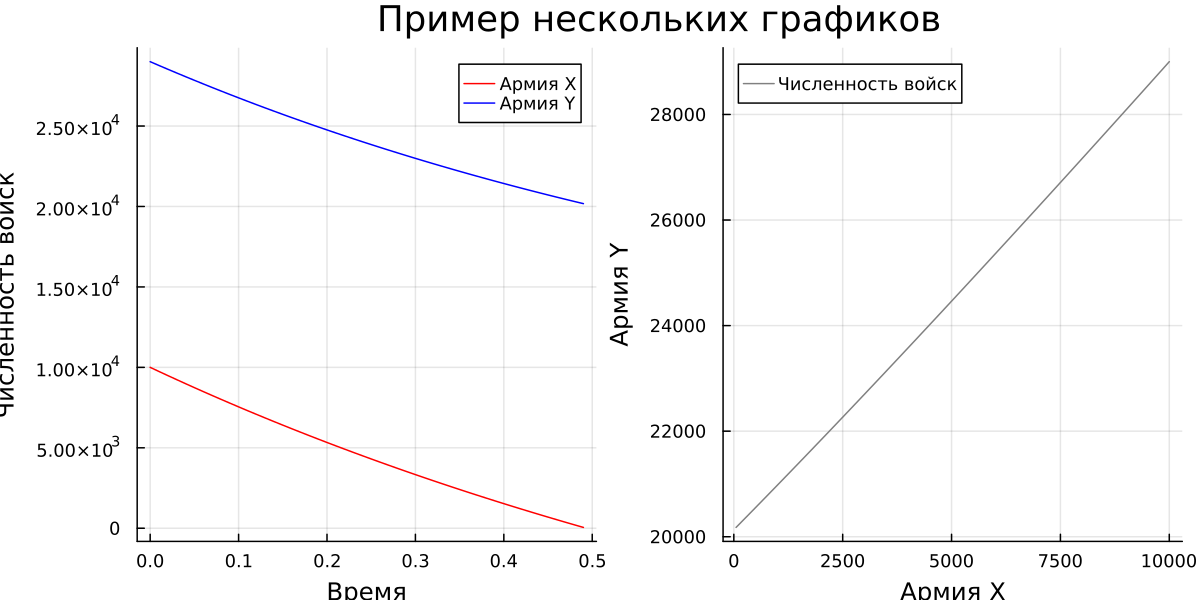
* 
* Figure 11: Результат работы программы

1. Пишем программу, описывающую модель боевых действий с участием регулярных войск и партизанских отрядов (рис. [12](#fig:12)).

* using Plots  
  using DifferentialEquations  
    
    
  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, 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="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\_2")

* 
* Figure 12: Измененные строчки кода по сравнению с предыдущим листингом

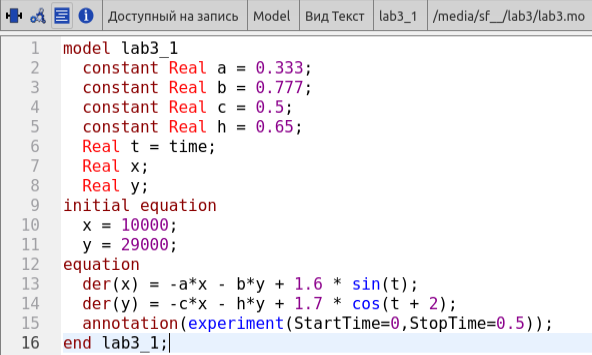
1. Получаем данные графики (рис. [13](#fig:13)).

* 
* Figure 13: Результат работы программы

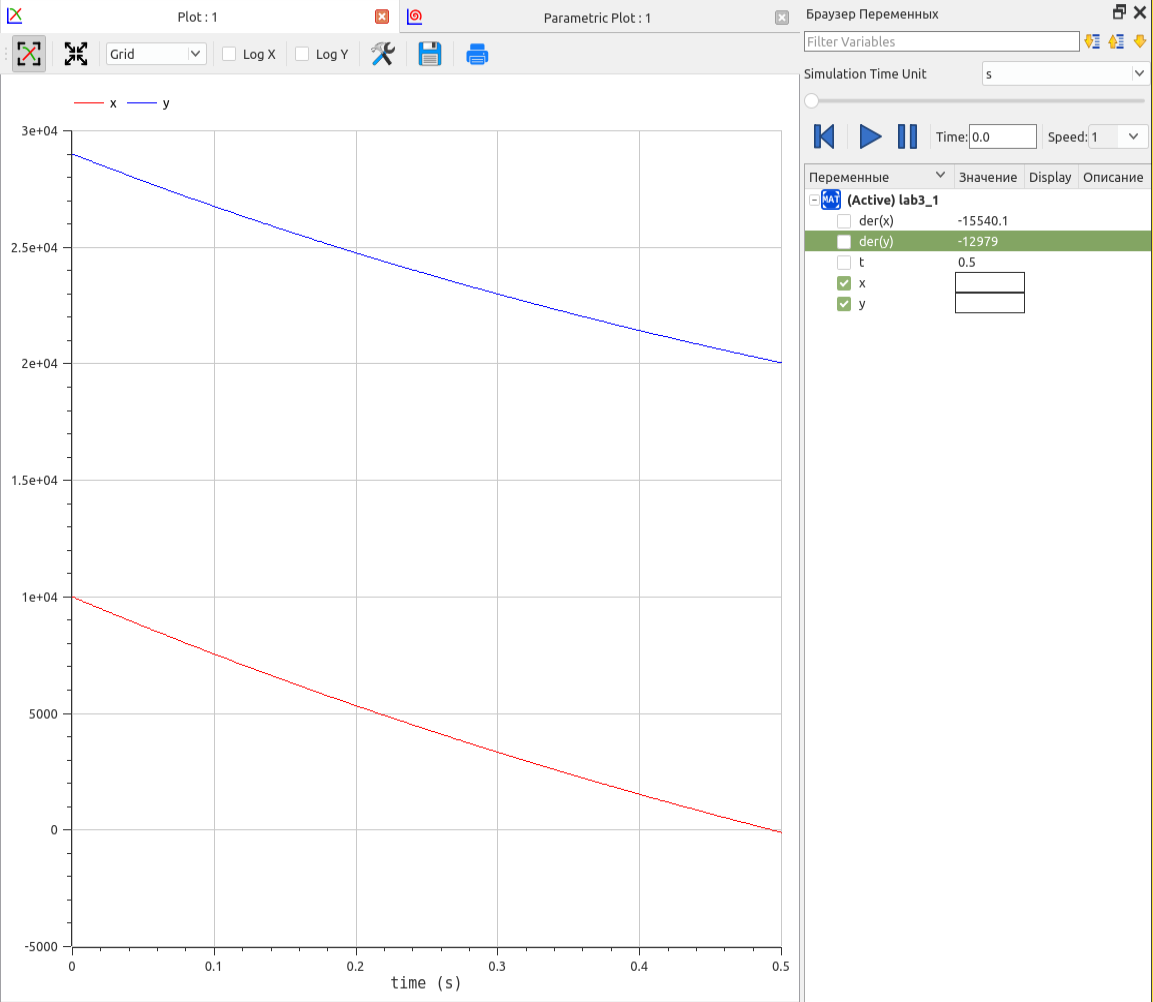
### 4.2.3 OpenModelica

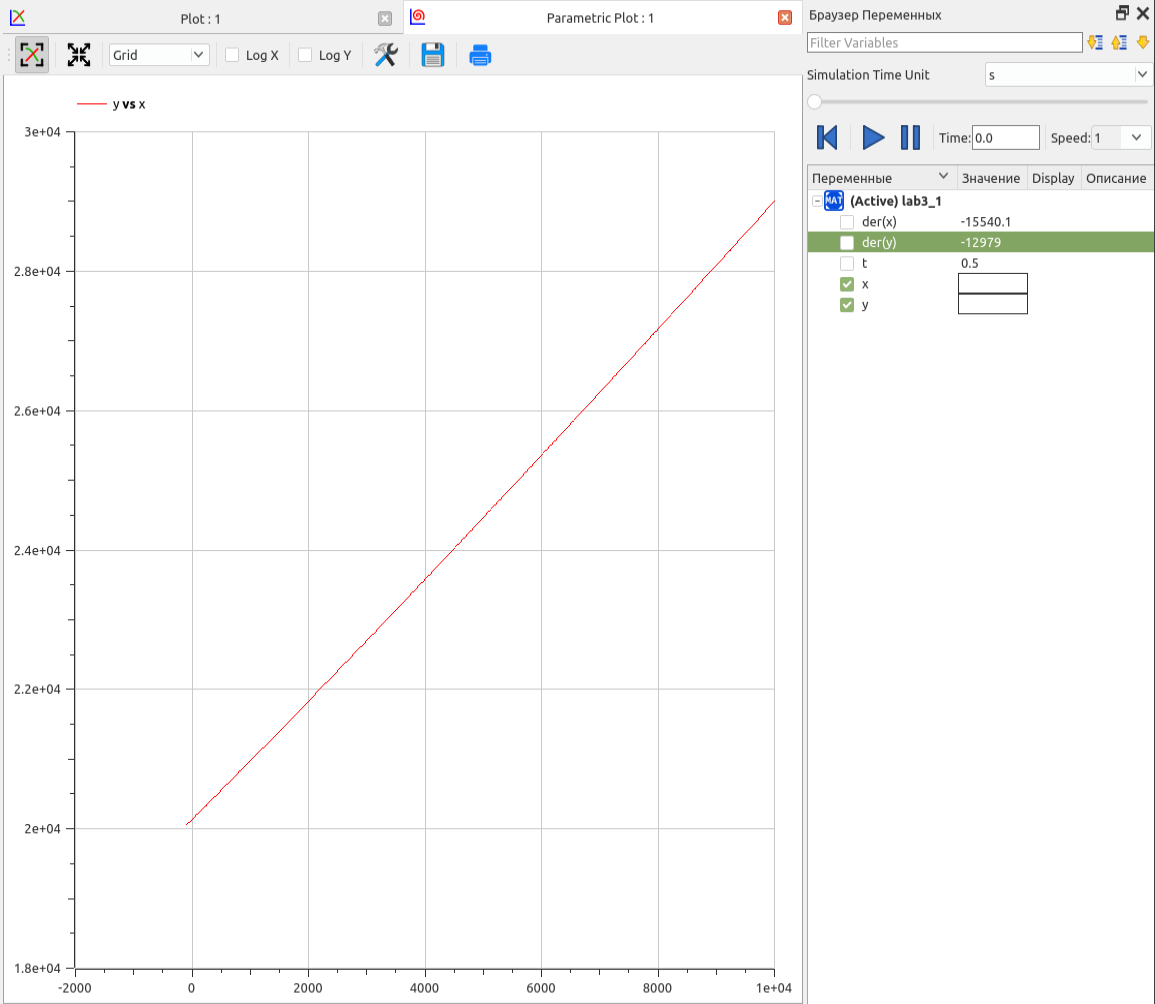
1. Пишем программу, описывающую модель боевых действий между регулярными войсками (рис. [14](#fig:14)).

* 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;

* 
* Figure 14: Задание констант, изменяющихся переменных, начальных условий, системы уравнений, времени моделирования

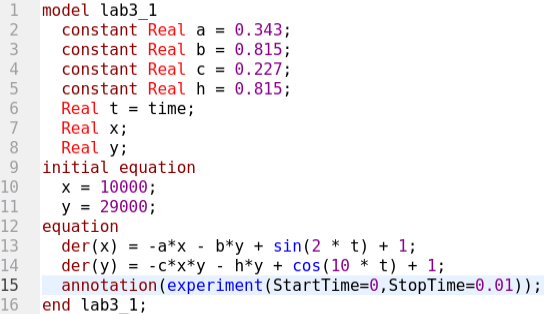
1. Получаем данные графики (рис. [15](#fig:15), [16](#fig:16)).

* 
* Figure 15: График зависимости численности армии от времени

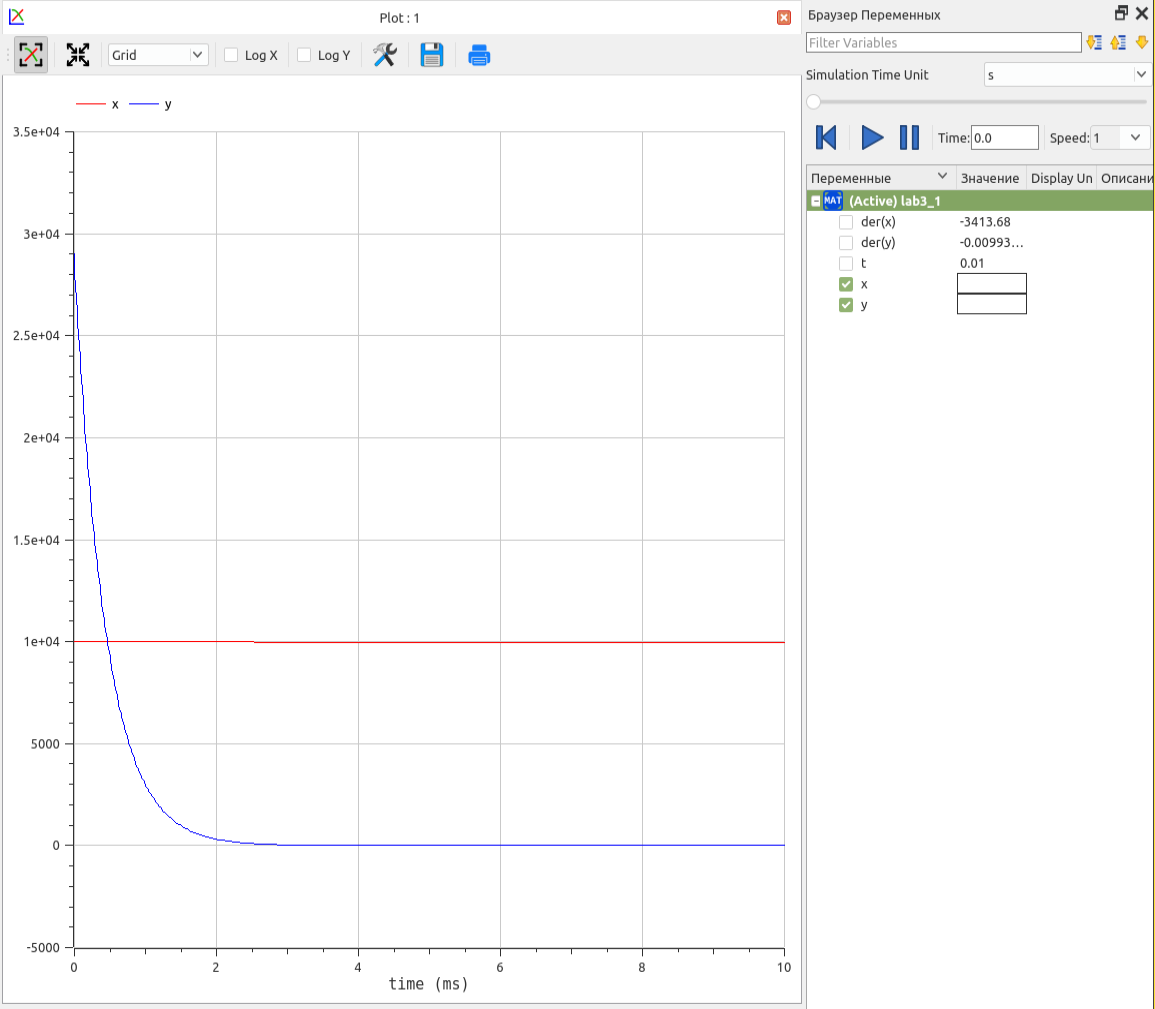
* 
* Figure 16: График зависимости численности армии X от армии Y

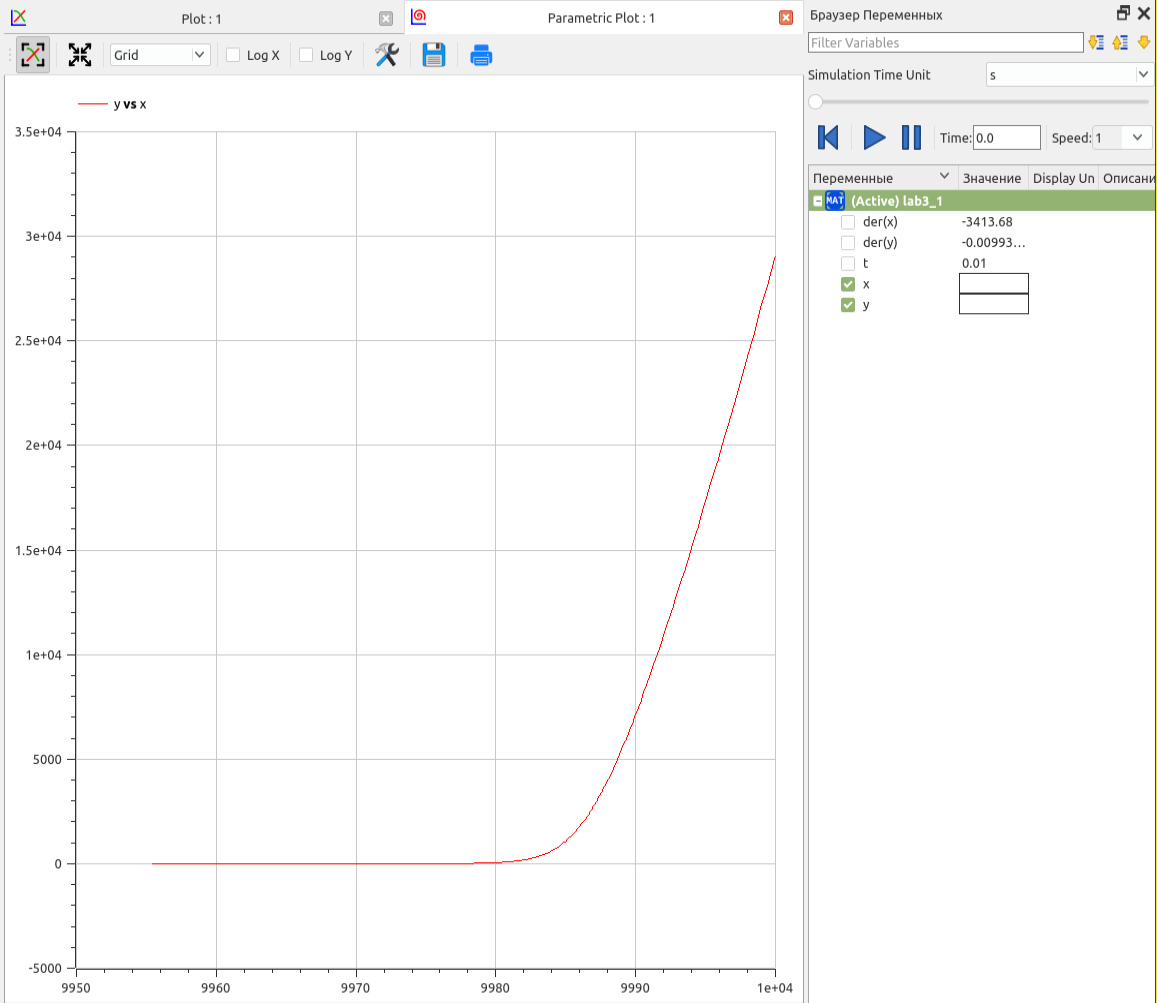
1. Пишем программу, описывающую модель боевых действий с участием регулярных войск и партизанских отрядов (рис. [17](#fig:17)).

* 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;

* 
* Figure 17: Задание иных констант, системы уравнений, времени моделирования

1. Получаем данные графики (рис. [18](#fig:18), [19](#fig:19)).

* 
* Figure 18: График зависимости численности армии от времени

* 
* Figure 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, приминительно к данной конкретной задаче было замечено, что суммарная длина кода на языке моделирование 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 интерпетируемым языков 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>.