

# Лабораторная работа №3

## "Модели Ланчестера"

Выполнил: Кармацкий Никита Сергеевич

НФИбд-01-21

## Цель работы:

Изучить модели боевых действий Ланчестера. Применить их на практике для решения задания лабораторной работы

## Теоретическая справка:

Законы Ланчестера (законы Осипова — Ланчестера) — математическая формула для расчета относительных сил пары сражающихся сторон — подразделений вооруженных сил

Уравнения Ланчестера — это дифференциальные уравнения, описывающие зависимость между силами сражающихся сторон  $A$  и  $D$  как функцию от времени, причем функция зависит только от  $A$  и  $D$ .

## Задание лабораторной работы:

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 61000 человек, а в распоряжении страны Y армия численностью в 45000 человек. Для упрощения модели считаем, что коэффициенты  $a, b, c, h$  постоянны. Также считаем  $P(t)$  и  $Q(t)$  непрерывными функциями.

# Задачи:

Построить графики изменения численности войск армии X и армии У для следующих случаев:

1. Модель боевых действий между регулярными войсками:

$$\frac{dx}{dt} = -0.22x(t) - 0.82y(t) + 2\sin(4t)$$

$$\frac{dy}{dt} = -0.45x(t) - 0.67y(t) + 2\cos(4t)$$

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

$$\frac{dx}{dt} = -0.28x(t) - 0.83y(t) + 1.5\sin(t)$$

$$\frac{dy}{dt} = -0.31x(t)y(t) - 0.75y(t) + 1.5\cos(t)$$

# **Основные этапы выполнения работы**

# 1. Математическая модель

Численность регулярных войск определяется тремя факторами:

1. Скорость уменьшения численности войск из-за причин, не связанных с боевыми действиями (болезни, травмы, дезертирство);
2. Скорость потерь, обусловленных боевыми действиями противоборствующих сторон (что связано с качеством стратегии, уровнем вооружения, профессионализмом солдат и т.п.);
3. Скорость поступления подкрепления (задаётся некоторой функцией от времени).

## 2. Скачиваем OpenModelica себе на устройство

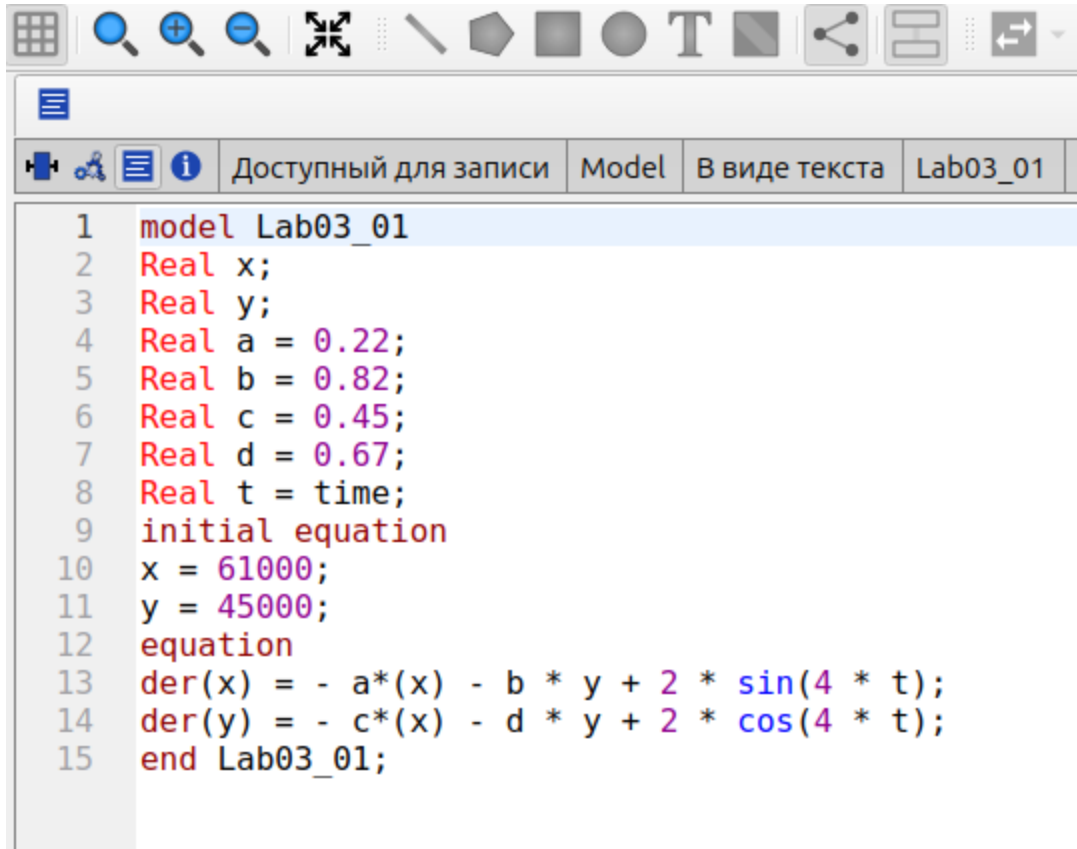
Для создания траектории движения будем использовать ЯП OpenModelica, но для начало установим все нужное для нормального функционирования.

```
nskarmatskiy@nskarmatskiy-M1050:~$ sudo apt install openmodelica
[sudo] пароль для nskarmatskiy:
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Уже установлен пакет openmodelica самой новой версии (1.22.2~3-g9f40725-1).
Обновлено 0 пакетов, установлено 0 новых пакетов, для удаления отмечено 0 пакетов
в, и 81 пакетов не обновлено.
nskarmatskiy@nskarmatskiy-M1050:~$
```

Рис.1 Установка OpenModelica



### 3. Пишем код для построения траектории на OpenModelica



The screenshot shows the OpenModelica IDE interface. At the top is a toolbar with various icons for file operations, editing, and simulation. Below the toolbar is a menu bar with options like 'Доступный для записи' (Available for recording), 'Model', 'В виде текста' (As text), and 'Lab03\_01'. The main area is a code editor displaying the following code:

```
1 model Lab03_01
2   Real x;
3   Real y;
4   Real a = 0.22;
5   Real b = 0.82;
6   Real c = 0.45;
7   Real d = 0.67;
8   Real t = time;
9   initial equation
10  x = 61000;
11  y = 45000;
12  equation
13  der(x) = - a*(x) - b * y + 2 * sin(4 * t);
14  der(y) = - c*(x) - d * y + 2 * cos(4 * t);
15 end Lab03_01;
```

Рис.2 Код для построения моделей(openModelica)

## 4. Просматриваем результаты работы программы

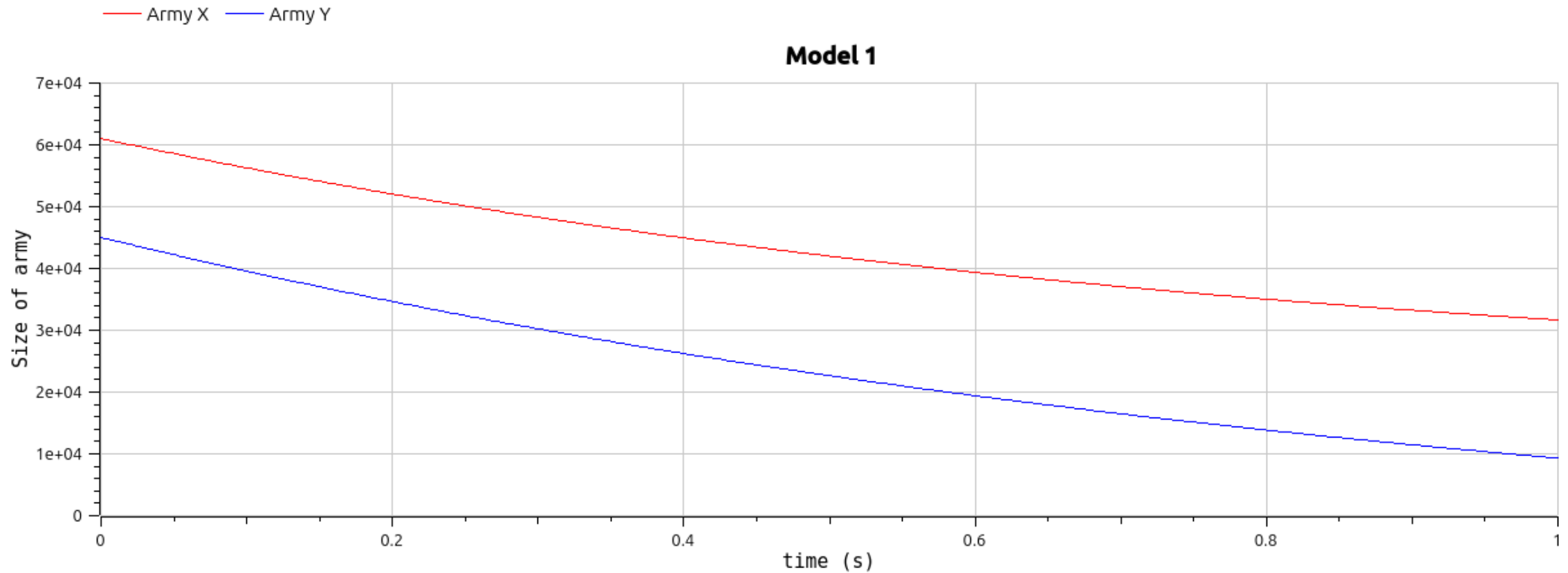


Рис.3 Первая модель(OpenModelica)

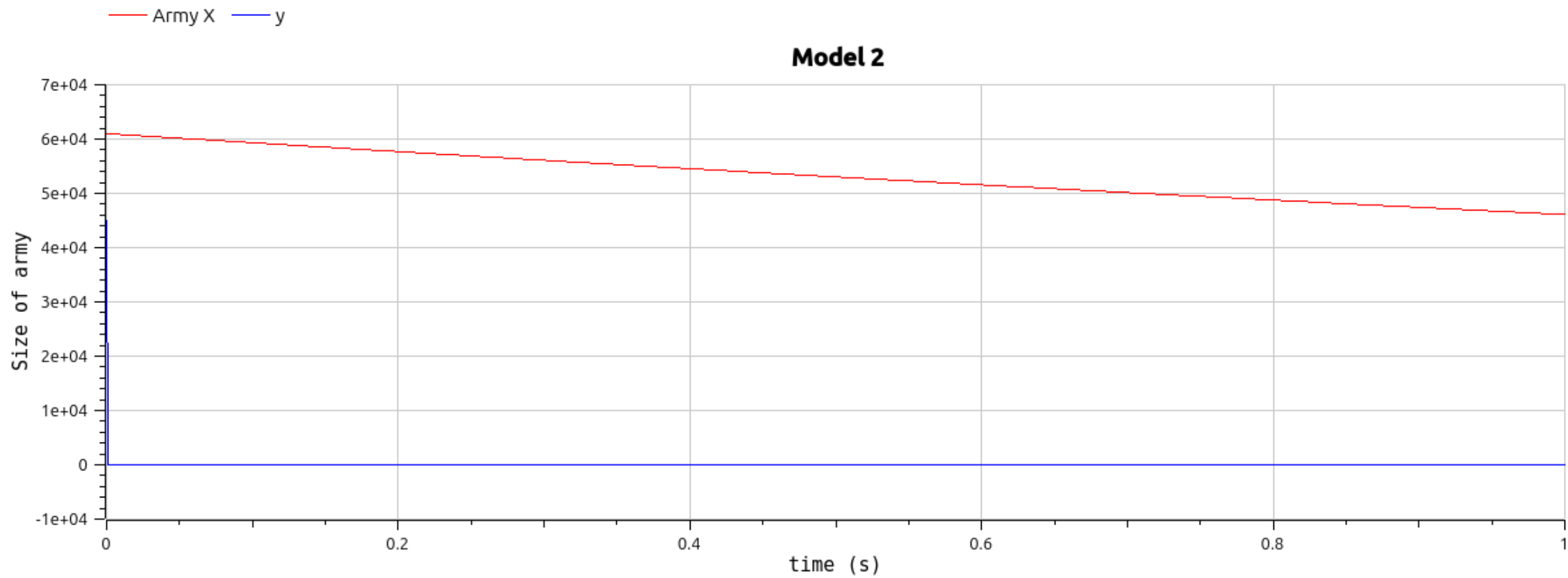


Рис.4 Вторая модель(OpenModelica)

## 5. Пишем код для построения траектории на Julia

```
using Plots;
using DifferentialEquations;

function one(du, u, p, t)
    du[1] = - 0.22*u[1] - 0.82*u[2] + 2*sin(4*t)
    du[2] = - 0.45*u[1] - 0.67*u[2] + 2*cos(4*t)
end

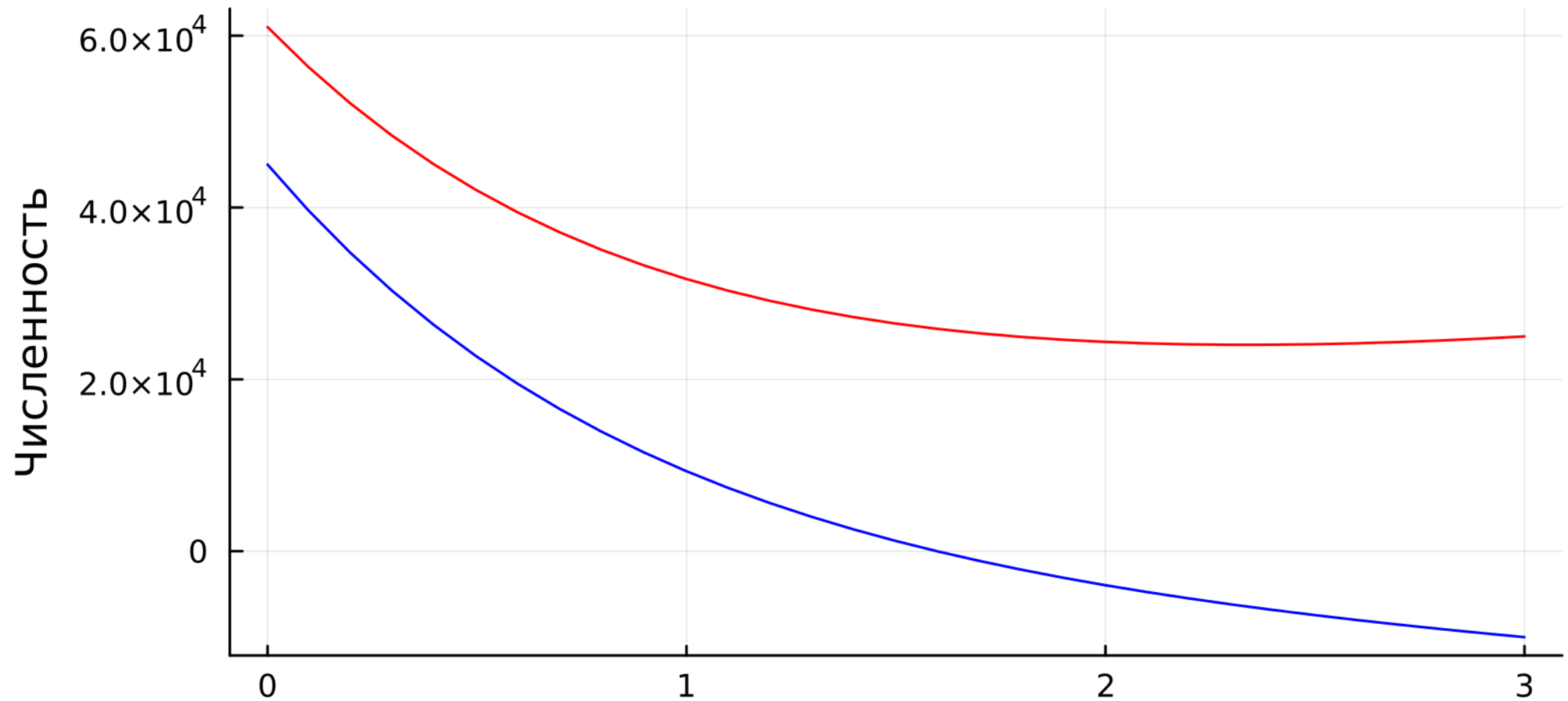
function two(du, u, p, t)
    du[1] = - 0.28*u[1] - 0.83*u[2] + 1.5*sin(t)
    du[2] = (- 0.31*u[1] - 0.75)*u[2] + 1.5*cos(t)
end

const people = Float64[61000, 45000]
const prom1 = [0.0, 3.0]
const prom2 = [0.0, 0.0007]

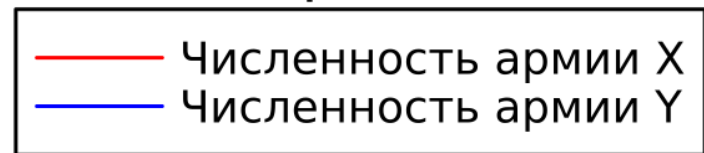
prob1 = ODEProblem(one, people, prom1)
prob2 = ODEProblem(two, people, prom2)
```

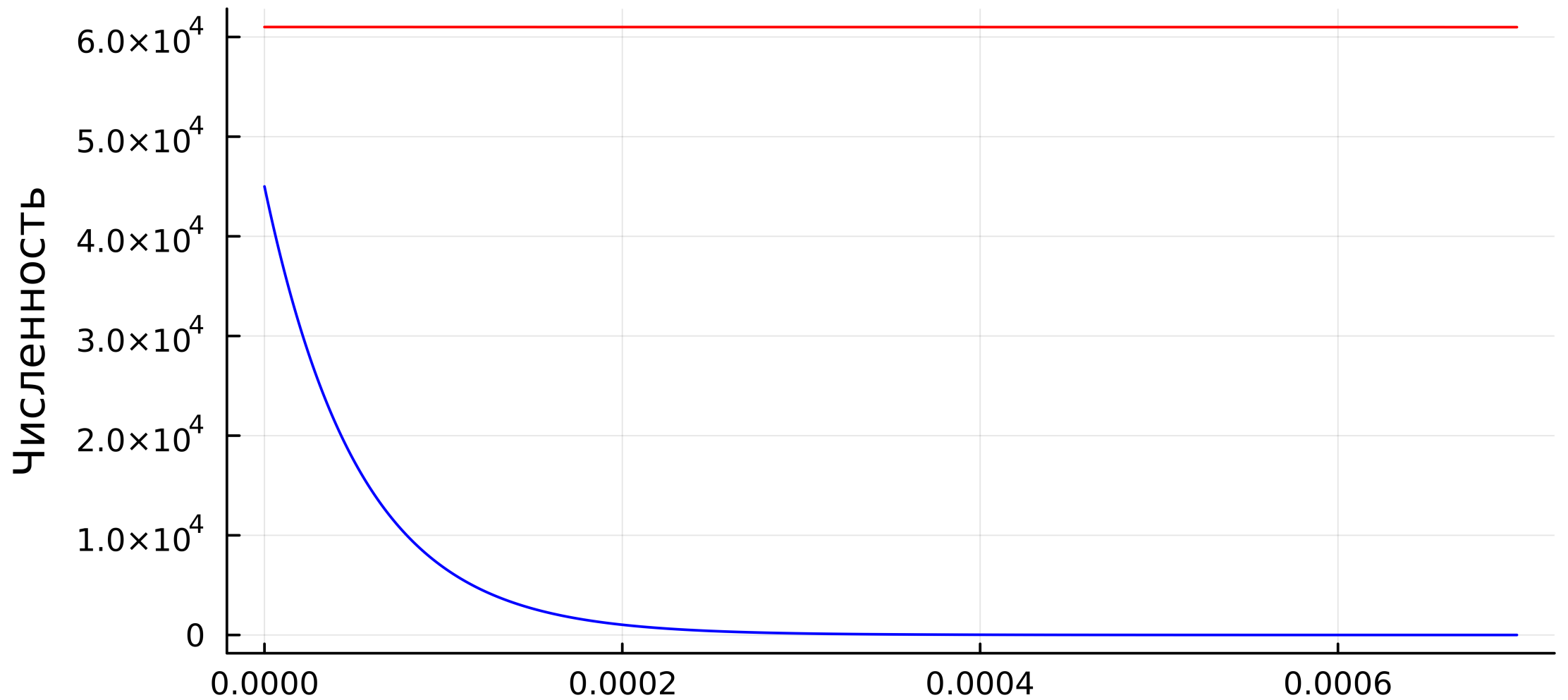
Рис. 6 Код на Julia

## **6. Просматриваем результат работы программы на Julia**

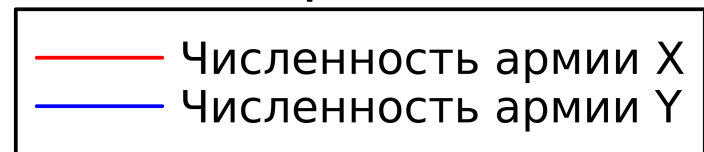


Время





Время



## **Анализ полученных результатов. Сравнение языков.**

Как видно из графиков, для первой модели, то есть двух регулярных армий, противостоящих друг другу, графики на Julia и OpenModelica идентичны (с поправкой на использование разных графических ресурсов, разный масштаб и т.д.).

Аналогичная ситуация верна и для графиков противостояния регулярной армии и армии партизанов, которые рассматривались во второй модели.



## **Вывод:**

По итогам лабораторной работы мы построили по две модели на языках Julia и OpenModelica. В ходе проделанной работы можно сделать вывод, что OpenModelica лучше приспособлен для моделирование процессов, протекающих во времени. Построение моделей боевых действий на языке OpenModelica занимает гораздо меньше строк и времени, чем аналогичное построение на языке Julia.

**Спасибо за внимание**