

# **Отчет по лабораторной работе № 2**

## **Задача о погоне**

Хусаинова Динара Айратовна

# Содержание

<b>Цель работы</b>	<b>4</b>
<b>Теоретическое введение</b>	<b>5</b>
<b>Задание</b>	<b>7</b>
<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>Моделирование с помощью Julia</b>	<b>11</b>
<b>Выводы</b>	<b>16</b>
<b>Список литературы</b>	<b>17</b>

## Список иллюстраций

1	Номер варианта . . . . .	8
1	Julia . . . . .	11
2	lab2.jl и запуск . . . . .	11
3	Процесс запуска . . . . .	12
4	Plots . . . . .	12
5	DifferentialEquations . . . . .	13
6	Using . . . . .	13
7	Запуск кода . . . . .	14
8	Случай 1 . . . . .	15
9	Случай 2 . . . . .	15

## Цель работы

Изучить основы языков программирования Julia и OpenModelica. Освоить библиотеки этих языков, которые используются для построения графиков и решения дифференциальных уравнений. Решить задачу о погоне.

# Теоретическое введение

## **Справка о языках программирования:**

Julia — высокоуровневый высокопроизводительный свободный язык программирования с динамической типизацией, созданный для математических вычислений. Эффективен также и для написания программ общего назначения. Синтаксис языка схож с синтаксисом других математических языков (например, MATLAB и Octave), однако имеет некоторые существенные отличия. Julia написан на Си, C++ и Scheme. Имеет встроенную поддержку многопоточности и распределённых вычислений, реализованные в том числе в стандартных конструкциях.

OpenModelica — свободное открытое программное обеспечение для моделирования, симуляции, оптимизации и анализа сложных динамических систем. Основано на языке Modelica. Активно развивается Open Source Modelica Consortium, некоммерческой неправительственной организацией. Open Source Modelica Consortium является совместным проектом RISE SICS East AB и Линчёпингского университета. По своим возможностям приближается к таким вычислительным средам как Matlab Simulink, Scilab xCos, имея при этом значительно более удобное представление системы уравнений исследуемого блока.

## **Математическая справка:**

Дифференциальное уравнение — уравнение, которое помимо функции содержит её производные. Порядок входящих в уравнение производных может быть различен (формально он ничем не ограничен). Производные, функции, независимые переменные и параметры могут входить в уравнение в различных комбинациях или отсутствовать вообще, кроме хотя бы одной производной. Не любое уравнение, содержащее производные

неизвестной функции, является дифференциальным.

В отличие от алгебраических уравнений, в результате решения которых ищется число (несколько чисел), при решении дифференциальных уравнений ищется функция (семейство функций).

Дифференциальное уравнение порядка выше первого можно преобразовать в систему уравнений первого порядка, в которой число уравнений равно порядку исходного дифференциального уравнения.

**Физические термины:**

- Тангенциальная скорость - составляющая вектора скорости, перпендикулярная линии, соединяющей источник и наблюдателя. Измеряется собственному движению - угловому перемещению источника.
- Радиальная скорость — проекция скорости точки на прямую, соединяющую её с выбранным началом координат.
- Полярная система координат — двумерная система координат, в которой каждая точка на плоскости определяется двумя числами — полярным углом и полярным радиусом.

## Задание

1. Запишите уравнение, описывающее движение катера, с начальными условиями для двух случаев (в зависимости от расположения катера относительно лодки в начальный момент времени).
2. Постройте траекторию движения катера и лодки для двух случаев.
3. Найдите точку пересечения траектории катера и лодки

# Выполнение лабораторной работы

Расчитываю свой вариант (рис. 1).

```
8
9  using System;
10 class HelloWorld {
11     static void Main() {
12         Console.WriteLine(((1032212283 % 70) + 1));
13     }
14 }
15
```




Рис. 1: Номер варианта

1. На море в тумане катер береговой охраны преследует лодку браконьеров. Через определенный промежуток времени туман рассеивается, и лодка обнаруживается на расстоянии 17,7 км от катера. Затем лодка снова скрывается в тумане и уходит прямолинейно в неизвестном направлении. Известно, что скорость катера в 3,8 раза больше скорости браконьерской лодки.
2. Примем за момент отсчета времени момент первого рассеивания тумана. Введем



полярные координаты с центром в точке нахождения браконьеров и осью, проходящей через катер береговой охраны. Тогда начальные координаты катера (17,7; 0). Обозначим скорость лодки  $v$ .

3. Траектория катера должна быть такой, чтобы и катер, и лодка все время были на одном расстоянии от полюса. Только в этом случае траектория катера пересечется с траекторией лодки. Поэтому для начала катер береговой охраны должен двигаться некоторое время прямолинейно, пока не окажется на том же расстоянии от полюса, что и лодка браконьеров. После этого катер береговой охраны должен двигаться вокруг полюса удаляясь от него с той же скоростью, что и лодка браконьеров.
4. Пусть время  $t$  - время, через которое катер и лодка окажутся на одном расстоянии от начальной точки.

$$t = \frac{x}{v}$$

$$t = \frac{17,7 - x}{3,8v}$$

$$t = \frac{17,7 + x}{3,8v}$$

Из этих уравнений получаем объединение двух уравнений:

$$\begin{cases} \frac{x}{v} = \frac{17,7 - x}{3,8v} \\ \frac{x}{v} = \frac{17,7 + x}{3,8v} \end{cases}$$

Решая это, получаем два значения для  $x$ :

$$x_1 = 14,0125$$

$$x_2 = 24,0214$$

$$v_\tau$$

– тангенциальная скорость

$$v$$

– радиальная скорость

$$v = \frac{dr}{dt}$$

$$v_\tau = \sqrt{((3,8 * v)^2 - v^2)} = \frac{\sqrt{70} * v}{5}$$

$$\left\{ \begin{array}{l} \frac{dr}{dt} = v \\ r \frac{d\theta}{dt} = \frac{\sqrt{70} * v}{5} \end{array} \right.$$

$$\left\{ \begin{array}{l} \theta_0 = 0 \\ r_0 = x_1 = 14,0125 \end{array} \right.$$

или

$$\left\{ \begin{array}{l} \theta_0 = -\pi \\ r_0 = x_2 = 24,0214 \end{array} \right.$$

Итоговое уравнение после того, как убрали производную по t:

$$\frac{dr}{d\theta} = \frac{5r}{\sqrt{70}}$$

# Моделирование с помощью Julia

К сожалению, OpenModelica не адаптирована к использованию полярных координат, поэтому адекватное отображение результатов данной задачи там невозможно.

## 1. Скачиваем Julia (рис. 2).



Рис. 1: Julia

## 2. Создаем файл lab2.jl и запускаем Julia (рис. 3).

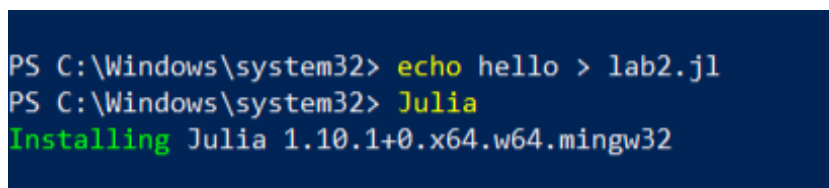


Рис. 2: lab2.jl и запуск

## 3. Запуск Julia (рис. 4).



```

julia> import Pkg; Pkg.add("DifferentialEquations")
Resolving package versions...
Installed Calculus _____ v0.5.1
Installed StatsFuns _____ v1.3.1
Installed MutableArithmetics _____ v1.4.1
Installed OffsetArrays _____ v1.13.0
Installed NonlinearSolve _____ v3.5.6
Installed DifferentialEquations _____ v7.12.0
Installed Polyester _____ v0.7.9
Installed TimerOutputs _____ v0.5.23
Installed EnumX _____ v1.0.4
Installed Sundials_jll _____ v5.2.2+0
Installed StaticArrays _____ v1.9.2
Installed RecursiveArrayTools _____ v3.8.1
Installed MaybeInplace _____ v0.1.1
Installed PDMats _____ v0.11.31
Installed CEnum _____ v0.5.0
Installed FunctionWrappers _____ v1.1.3
Installed TriangularSolve _____ v0.1.20
Installed IntelOpenMP_jll _____ v2024.0.2+0
Installed Static _____ v0.8.10
Installed SLEEFPirates _____ v0.6.42

```

Рис. 5: DifferentialEquations

```

julia> using DifferentialEquations

julia> using Plots

julia>

```

Рис. 6: Using

5. Исходный код в файле lab2.jl:

```
using Plots using DifferentialEquations
```

```
const a = 17.7 const n = 3.8
```

```
const r0 = a/(n + 1) const r0_2 = a/(n - 1)
```

```
const T = (0, 2*pi) const T_2 = (-pi, pi)
```

```
function F(u, p, t) return u / sqrt(n*n - 1) end
```

```
problem = ODEProblem(F, r0, T)
```

```
result = solve(problem, abstol=1e-8, reltol=1e-8) @show result.u @show result.t
```

```

dxR = rand(1:size(result.t)[1]) rAngles = [result.t[dxR] for i in 1:size(result.t)[1]]
plt = plot(proj=:polar, aspect_ratio=:equal, dpi = 1000, legend=true, bg=:white)
plot!(plt, xlabel="theta", ylabel="r(t)", title="Хусаинова Задача о погоне. Случай номер
1", legend=:outerbottom) plot!(plt, [rAngles[1], rAngles[2]], [0.0, result.u[size(result.u)[1]]],
label="Путь лодки", color=:blue, lw=1) scatter!(plt, rAngles, result.u, label="", mc=:blue,
ms=0.0005) plot!(plt, result.t, result.u, xlabel="theta", ylabel="r(t)", label="Путь катера",
color=:green, lw=1) scatter!(plt, result.t, result.u, label="", mc=:green, ms=0.0005)
savefig(plt, "lab2_01.png")

problem = ODEProblem(F, r0_2, T_2) result = solve(problem, abstol=1e-8, reltol=1e-8)
dxR = rand(1:size(result.t)[1]) rAngles = [result.t[dxR] for i in 1:size(result.t)[1]]
plt1 = plot(proj=:polar, aspect_ratio=:equal, dpi = 1000, legend=true, bg=:white)
plot!(plt1, xlabel="theta", ylabel="r(t)", title="Хусаинова Задача о погоне. Случай номер
2", legend=:outerbottom) plot!(plt1, [rAngles[1], rAngles[2]], [0.0, result.u[size(result.u)[1]]],
label="Путь лодки", color=:blue, lw=1) scatter!(plt1, rAngles, result.u, label="", mc=:blue,
ms=0.0005) plot!(plt1, result.t, result.u, xlabel="theta", ylabel="r(t)", label="Путь катера",
color=:green, lw=1) scatter!(plt1, result.t, result.u, label="", mc=:green, ms=0.0005)
savefig(plt1, "lab2_02.png")

```

6. Запуск кода ( рис. 8).

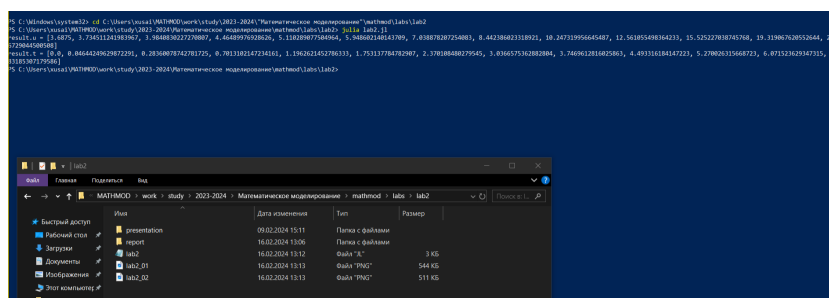


Рис. 7: Запуск кода

7. Просмотр результата работы ( рис. 9-10).

### Хусаинова Задача о погоне. Случай номер 1

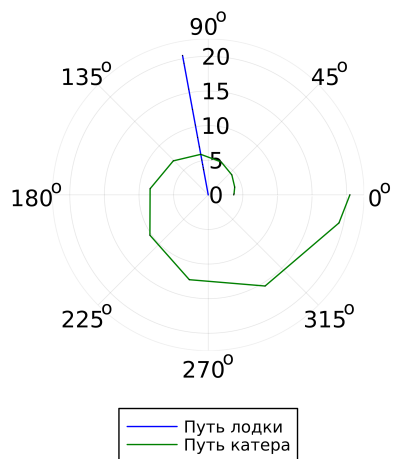


Рис. 8: Случай 1

### Хусаинова Задача о погоне. Случай номер 2

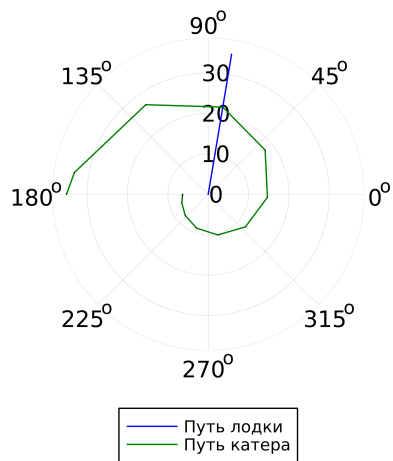


Рис. 9: Случай 2

## **Выводы**

Были изучены основы языков программирования Julia и OpenModelica. Освоены библиотеки этих языков, которые используются для построения графиков и решения дифференциальных уравнений. Поскольку OpenModelica не работает с полярными координатами, она пока что не была использована в данной лабораторной работе.



## Список литературы

- [1] Документация по Julia: <https://docs.julialang.org/en/v1/>
- [2] Документация по OpenModelica: <https://openmodelica.org/>
- [3] Решение дифференциальных уравнений: <https://www.wolframalpha.com/>