Отчёт по лабораторной работе №2

Задача о погоне

Желдакова Виктория Алексеевна

Содержание

# 1 Цель работы

Изучить основы языков Julia и OpenModelica. Познакомиться с библиотеками Plots и DifferentialEquations для построения графиков и решения дифференциальных уравнений. Решить задачу о погоне с использованием обоих языков.

# 2 Задание

## 2.1 Вариант 16

На море в тумане катер береговой охраны преследует лодку браконьеров. Через определенный промежуток времени туман рассеивается, и лодка обнаруживается на расстоянии 9,5 км от катера. Затем лодка снова скрывается в тумане и уходит прямолинейно в неизвестном направлении. Известно, что скорость катера в 3,3 раза больше скорости браконьерской лодки.

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

# 3 Теоретическое введение

## 3.1 Справка о языках программирования

Julia — высокоуровневый 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. OpenModelica используется в академической среде и на производстве. В промышленности используется в области оптимизации энергоснабжения,автомобилестроении и водоочистке.

## 3.2 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 вовсе, кроме хотя бы одной производной. Не любое уравнение, содержащее производные неизвестной функции, является дифференциальным.

Дифференциальные уравнения являются частным случаем функциональных уравнений. В отличие от алгебраических уравнений, в результате решения которых ищется число (несколько чисел), при решении дифференциальных уравнений ищется функция (семейство функций).

# 4 Выполнение лабораторной работы

## 4.1 Математическая модель

Согласно варианту расстояние между лодкой и катером равно 9,5 км, а отношение скорости катера в 3,3 раза больше скорости браконьерской лодки.

Введём полярные координаты с центром в точке нахождения браконьеров и осью, проходящей через катер береговой охраны. 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.

Чтобы найти расстояние x (расстояние после которого катер начнет двигаться вокруг полюса), необходимо составить простое уравнение. Пусть через время t катер и лодка окажутся на одном расстоянии x от полюса. За это время лодка пройдет , а катер (или , в зависимости от начального положения катера относительно полюса). Время, за которое они пройдут это расстояние, вычисляется как или (во втором случае ). Так как время одно и то же, то эти величины одинаковы. Мы получаем объединение из двух уравнений для двух различных начальных положений катера:

Из данных уравнений можно найти расстояние, после которого катер начнёт раскручиваться по спирали. Для данных уравнений решения будут следующими: , . Задачу будем решать для двух случаев. После того, как катер береговой охраны окажется на одном расстоянии от полюса, что и лодка, он должен сменить прямолинейную траекторию и начать двигаться вокруг полюса удаляясь от него со скоростью лодки v. Для этого скорость катера раскладываем на две составляющие: - радиальная скорость и - тангенциальная скорость.

1. Решение исходной задачи сводится к решению системы из двух дифференциальных уравнений:

с начальными условиями

или

Исключая из полученной системы производную по t, можно перейти к следующему уравнению (с неизменными начальными условиями):

Решением этого уравнения с заданными начальными условиями и будет являться траектория движения катера в полярных координатах.

## 4.2 Решение с помощью языков программирования

### 4.2.1 OpenModelica

Реализация решения данной задачи невозможна с помощью OpenModelica, т.к. в ней не поддерживаются полярные координаты. [1]

### 4.2.2 Julia

Установим Julia (рис. 1).



Рис. 1: Установка Julia

Установим пакеты Plots и DifferentialEquations для создания графиков и решения лифференциальных уравнений соответственно и проверим их установку (рис. 2). [2]

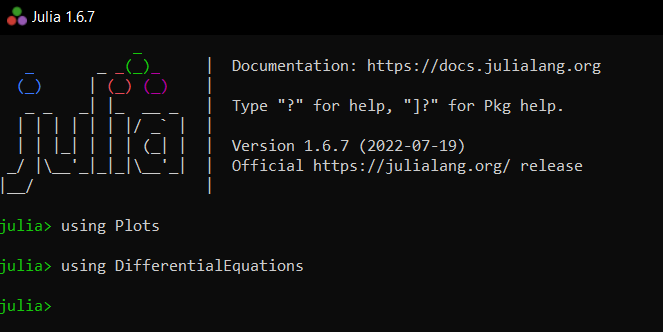


Рис. 2: Проверка установки пакетов

Напишем программу для решения нашей задачи. Код программы:

using Plots  
using DifferentialEquations  
  
const k = 12.2  
const n = 4.1  
  
const r0 = k/(n + 1)  
const r02 = k/(n - 1)  
  
const T = (0, 2\*pi)  
const T2 = (-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="Задача о погоне - случай 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, "lab02\_01.png")  
  
problem = ODEProblem(F, r02 , T2)  
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="Задача о погоне - случай 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, "lab02\_02.png")

Запустим программу (рис. 3).

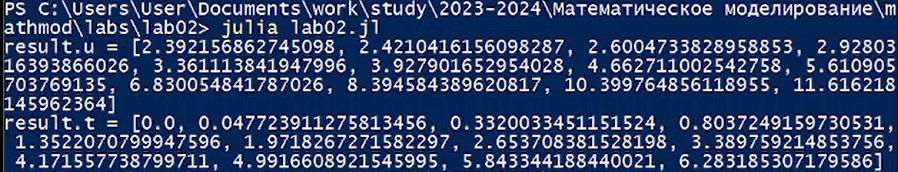


Рис. 3: Запуск программы и её вывод

В результате работы программы получаем графики для обоих случаев начального положения катера относительно полюса (рис. 4) (рис. 5).



Рис. 4: График для первого случая

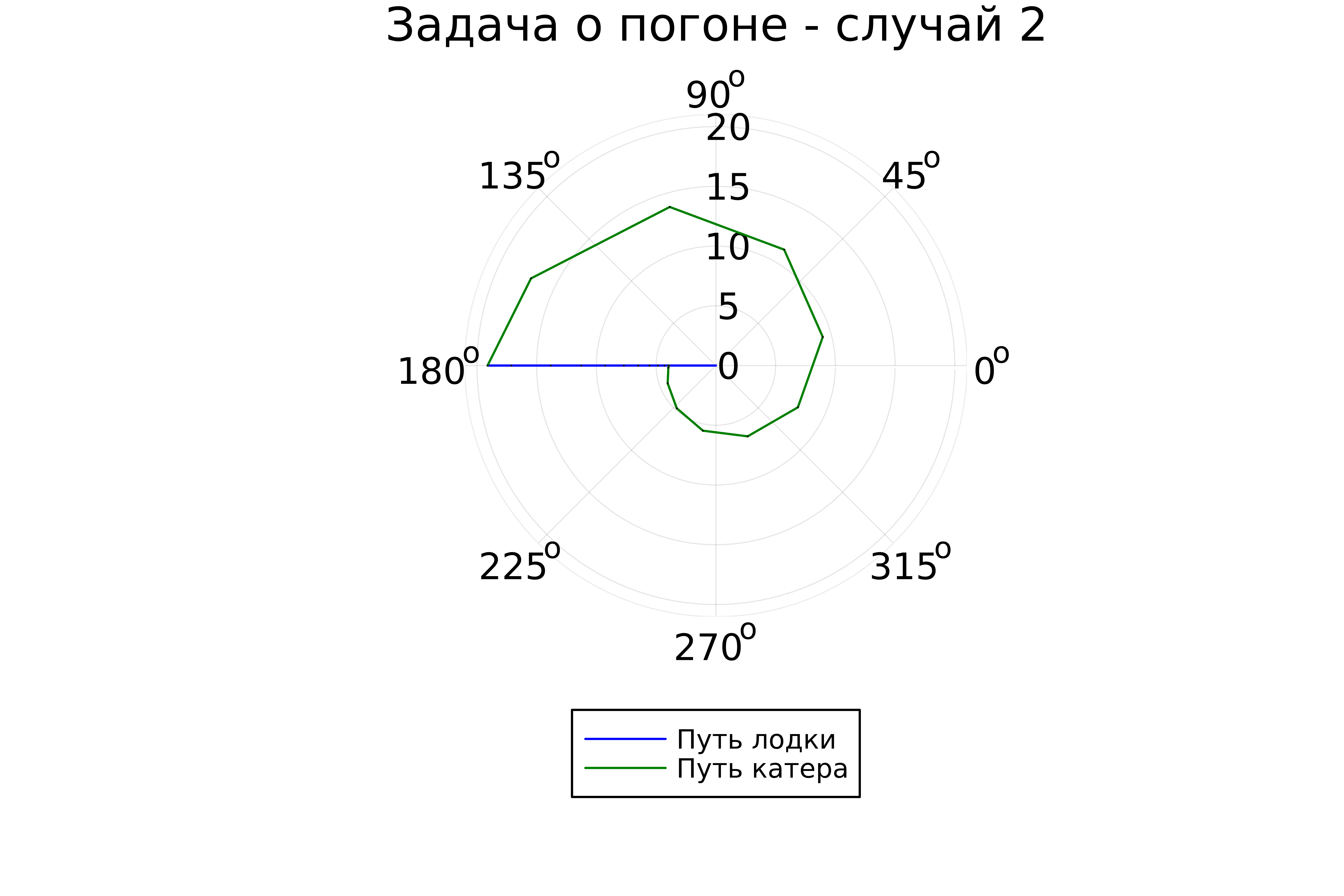


Рис. 5: График для второго случая

# 5 Выводы

Изучила основы языков Julia и OpenModelica. Познакомилась с библиотеками Plots и DifferentialEquations для построения графиков и решения дифференциальных уравнений. Решила задачу о погоне только с использованием языка Julia, т.к. OpenModelica не поддерживает работу с полярными координатами.

# Список литературы

[1] Документация по OpenModelica: https://openmodelica.org/

[2] Документация по Julia: https://docs.julialang.org/en/v1/