Отчет по лабораторной работе №2

Задача о погоне

Легиньких Галина Андреевна

Содержание

Список иллюстраций

Список таблиц

1 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.

2 Теоретическое введение

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. [1]

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. [2]

3 Задание

Задания лабораторной работы разделены по вариантам. **Мой вариант 18** (исходя из формулы $N_{student} mod K_{ofvariants} + 1$).

На море в тумане катер береговой охраны преследует лодку браконьеров. Через определенный промежуток времени туман рассеивается, и лодка обнаруживается на расстоянии 7,7 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

4 Выполнение работы

1. Принимает за $t_0 = 0$, $x_0 = 0$ - место нахождения лодки браконьеров в момент обнаружения, $x_0 = 7,7$ - место нахождения катера береговой охраны относительно лодки браконьеров в момент обнаружения лодки.
2. Введем полярные координаты. Считаем, что полюс - это точка обнаружения лодки браконьеров $x0(\theta_0 = x_0 = 0)$, а полярная ось r проходит через точку нахождения катера береговой охраны
3. Траектория катера должна быть такой, чтобы и катер, и лодка все время были на одном расстоянии от полюса θ. 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. Чтобы найти расстояние x (расстояние после которого катер начнет двигаться вокруг полюса), необходимо составить следующие уравнение. Пусть через время t катер и лодка окажутся на одном расстоянии x от полюса. За это время лодка пройдет x, а катер $7,7 - x$ (или $7,7 + x$, в зависимости от начального положения катера относительно полюса). Время, за которое они пройдут это расстояние, вычисляется как $\frac{x}{v}$ или $\frac{7,7-x}{3,3v}$ (во втором случае $\frac{7,7+x}{3,3v}$). Так как время должно быть одинаковым, эти величины тоже будут друг другу равны. Тогда неизвестное расстояние x можно найти из следующего уравнения:

$$\left[\begin{array}{l} \frac{x}{v} = \frac{7{,}7-x}{3{,}3v} \\ \frac{x}{v} = \frac{7{,}7+x}{3{,}3v} \end{array}\right.$$

Отсюда мы найдем два значения $x_1 = \frac{7{,}7}{4{,}3}$, $x_2 = \frac{7{,}7}{2{,}3}$, задачу будем решать для двух случаев.

5. 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: v_r - радиальная скорость и v_τ - тангенциальная скорость. Радиальная скорость - это скорость, с которой катер удаляется от полюса, $v_r = \frac{dr}{dt}$. Нам нужно, чтобы эта скорость была равна скорости лодки, поэтому полагаем $\frac{dr}{dt} = v$. Тангенциальная скорость – это линейная скорость вращения катера относительно полюса. Она равна произведению угловой скорости $\frac{d\theta}{dt}$ на радиус r, $v_\tau = r\frac{d\theta}{dt}$

$$v_\tau = \frac{\sqrt{989}v}{10}$$

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

$$\left\{\begin{array}{c} \frac{dr}{dt} = v \\ r\frac{d\theta}{dt} = \frac{\sqrt{989}v}{10} \end{array}\right.$$

с начальными условиями

$$\left\{\begin{array}{c} \theta_0 = 0 \\ r_0 = x_1 = \frac{7{,}7}{4{,}3} \end{array}\right.$$

или

$$\begin{cases} \theta_0 = -\pi \\ r_0 = x_2 = \frac{7{,}7}{2{,}3} \end{cases}$$

Исключая из полученной системы производную по t, можно перейти к следующему уравнению:

$$\frac{dr}{d\theta} = \frac{10r}{\sqrt{989}}$$

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

4.1 Julia

Решить дифференциальное уравнение, расписанное в постановке задачи лабораторной работы, поможет библиотека **DifferentialEquations**. Итоговые изображения в полярных координатах будут строиться через библиотеку **Plots**.

Установим Julia: (рис. 4.1)

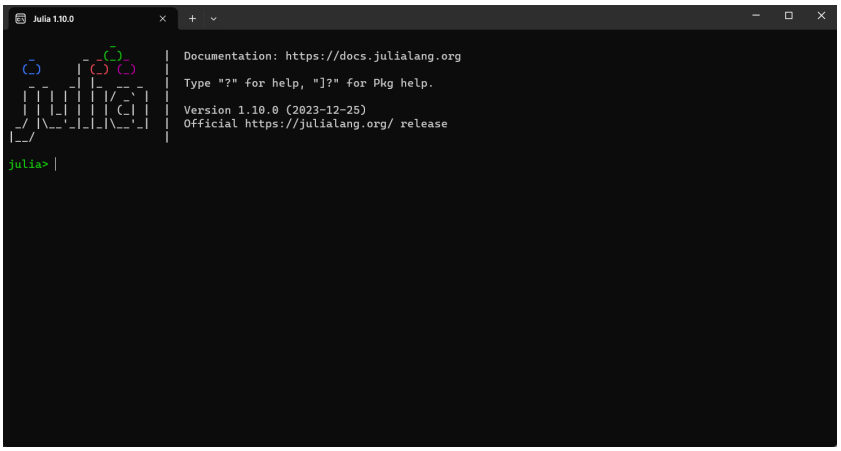

Рис. 4.1: Julia

Установим нужные библиотеки, проверим их установку:

```
import Pkg
```

```
Pkg.add("Plots")
Using Plots
Pkg.add("DifferentialEquations")
using DifferentialEquations
```

Код программы:

```
using Plots
using DifferentialEquations

# расстояние от лодки до катера
const a = 7.7
const n = 3.3

# расстояние начала спирали
const r0 = a/(n + 1)
const r0_2 = a/(n - 1)
# интервал
const T = (0, 2*pi)
const T_2 = (-pi, pi)

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

# задача ОДУ
problem = ODEProblem(F, r0, T)

#решение
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]]

#холст1
plt = plot(proj=:polar, aspect_ratio=:equal, dpi = 1000, legend=true, bg=:white)

#параметры для холста
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="Пу
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",
scatter!(plt, result.t, result.u, label="", mc=:green, ms=0.0005)

savefig(plt, "lab02_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]]

#холст2
plt1 = plot(proj=:polar, aspect_ratio=:equal, dpi = 1000, legend=true, bg=:white)

#параметры для холста
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="П
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",
scatter!(plt1, result.t, result.u, label="", mc=:green, ms=0.0005)

savefig(plt1, "lab02_02.png")
```

Скомпилируем файл командной в PShell: (рис. 4.2)

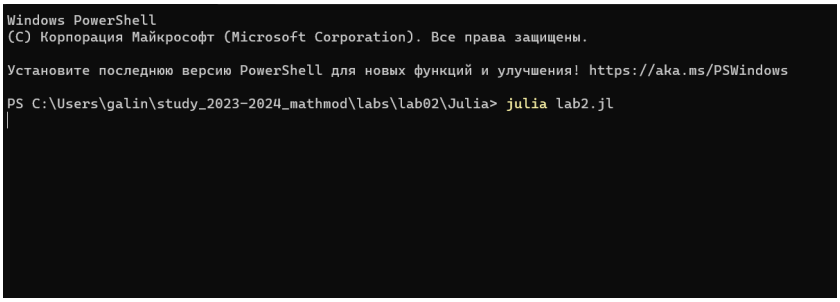

Рис. 4.2: PShell

Результат для первого случая: (рис. 4.3)

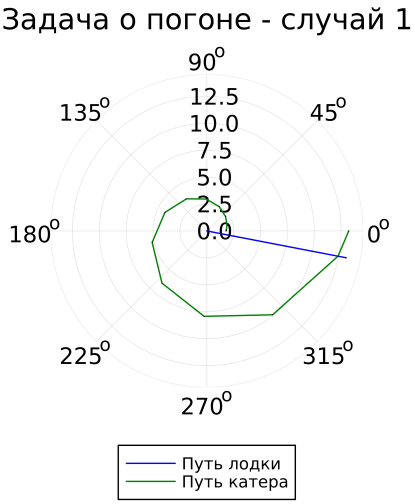

Рис. 4.3: 1 случай

Результат для второго случая: (рис. 4.4)

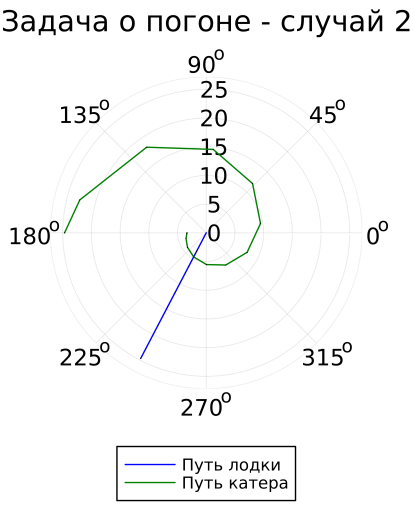

Рис. 4.4: 2 случай

4.2 OpenModelica

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

5 Анализ полученных результатов

Мною были построены графики для обоих случаев. На них получилось отрисовать трактерию катера, траекторию лодки и получилось наглядно найти их точки пересечения. Мы успешно решили задачу о погоне.

6 Вывод

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.

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

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