

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ им. Патриса Лулумбы

Факультет физико-математических и
естественных наук

Кафедра теории вероятности и
кибербезопасности

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

дисциплина: Математическое моделирование

Студент: Кармацкий Никита Сергеевич

Номер студ.билета: 1032210091

Группа: НФИбд-01-21

Москва

2024 г.

Цель работы:

Изучить основы языков программирования Julia и OpenModelica. Освоить библиотеки этих ЯП, которые используются для построения графиков и решения ДУ. Так же надо 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 вовсе, 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

Задания лабораторной работы разделены по вариантам. **Мой вариант 32**

(исходя из формулы $N_{student} \bmod K_{of variants} + 1$).

Этот же вариант будет использоваться для всех последующих лабораторных работ.

Задача о погоне. Вариант 32:

На море в тумане катер береговой охраны преследует лодку браконьеров. Через определенный промежуток времени туман рассеивается, и лодка обнаруживается на расстоянии 11,5 км от катера. Затем лодка снова скрывается в тумане и уходит прямолинейно в неизвестном направлении. Известно, что скорость катера в 3,5 раза больше скорости браконьерской лодки.

Задачи:

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

Основные этапы выполнения работы

Математическая модель

1. 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 $(11,5; 0)$. Обозначим скорость лодки v .
2. 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.
3. 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 $11,5 + x$ (или $11,5 - x$, в зависимости от начального положения катера относительно полюса). Время, за которое они пройдут это расстояние, вычисляется как $\frac{x}{v}$ или $\frac{11,5-x}{3,5v}$ ($\frac{11,5+x}{3,5v}$). Так как время должно быть одинаковым, эти величины тоже будут друг другу равны. Из этого получаем объединение из двух уравнений (двух из-за двух разных изначальных позиций катера относительно полюса):

$$\begin{cases} \frac{x}{v} = \frac{11,5-x}{3,5v} \\ \frac{x}{v} = \frac{11,5+x}{3,5v} \end{cases}$$

Из данных уравнений можно найти расстояние, после которого катер начнёт раскручиваться по спирали. Для данных уравнений решения будут следующими: $x_1 = \frac{23}{9}$, $x_2 = \frac{23}{5}$. 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: $v_r = \frac{dr}{dt} = v$ - радиальная скорость и $v_t = r \frac{d\theta}{dt}$ - тангенциальная скорость.

$$v_t = \frac{3v * \sqrt{5}}{10}$$

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

$$\begin{cases} \frac{dr}{dt} = v \\ r \frac{d\theta}{dt} = \frac{3v * \sqrt{5}}{10} \end{cases}$$

с начальными условиями

$$\begin{cases} \theta_0 = 0 \\ r_0 = x_1 = \frac{23}{9} \end{cases}$$

или

$$\begin{cases} \theta_0 = -\pi \\ r_0 = x_2 = \frac{23}{5} \end{cases}$$

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


$$\frac{dr}{d\theta} = \frac{2r}{3 * \sqrt{5}}$$

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

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

1. Скачиваем Julia себе на устройство

```
nskarmatskiy@nskarmatskiy-M1050:~$ sudo snap install julia --classic
julia 1.10.1 or The Julia Language (julialang✓) установлен
nskarmatskiy@nskarmatskiy-M1050:~$ julia
```



Documentation: <https://docs.julialang.org>

Type "?" for help, "]??" for Pkg help.

Version 1.10.1 (2024-02-13)

Official <https://julialang.org/> release

```
julia> 1+1
\2

julia> 1+2
3
```

Рис.1 Установка Julia

2. Проводим расчеты для своего варианта:

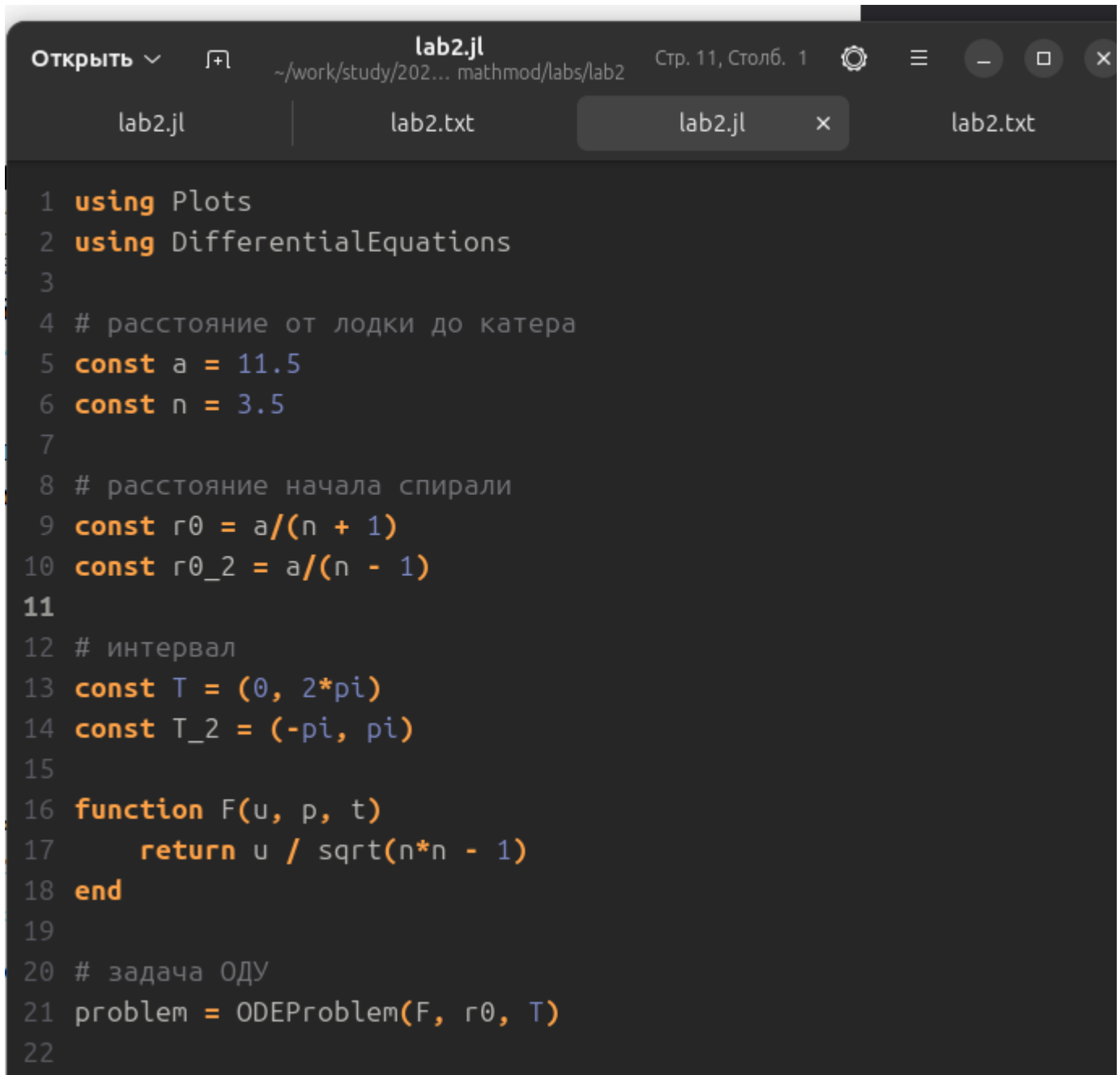
```

1 {x/v = (11,5 - x)/3,5v -> x1 = 23/9
2 {x/v = (11,5 + x)/3,5v -> x2 = 23/5
3
4 Vt = sqrt(3,5v^2 - v^2) = (3v*sqrt(5))/2
5
6 dr/dθ = 2r/3*sqrt(5)
7

```

Рис.2 Расчеты

3. Создаем файл и пишем код для решения задачи:



```
1 using Plots
2 using DifferentialEquations
3
4 # расстояние от лодки до катера
5 const a = 11.5
6 const n = 3.5
7
8 # расстояние начала спирали
9 const r0 = a/(n + 1)
10 const r0_2 = a/(n - 1)
11
12 # интервал
13 const T = (0, 2*pi)
14 const T_2 = (-pi, pi)
15
16 function F(u, p, t)
17     return u / sqrt(n*n - 1)
18 end
19
20 # задача ОДУ
21 problem = ODEProblem(F, r0, T)
22
```

Рис.3 Рабочий файл

```
using Plots
using DifferentialEquations

# расстояние от лодки до катера
const a = 11.5
const n = 3.5

# 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="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, 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="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")

```

4. Компилируем файл в командной строке:

Для компиляции файла будем использовать команду: **julia lab2.jl**


```
ирование/mathmod/labs/lab2/  
nskarmatskiy@nskarmatskiy-M1050:~/work/study/2023-2024/Математическое моделирова  
ние/mathmod/labs/lab2$ julia lab2.jl
```

Рис.4 Компиляция файла

5. Результаты работы код на Julia

На рисунках ниже показына графики траектории движения катера и лодки в двух случаях

Задача о погоне - случай 1

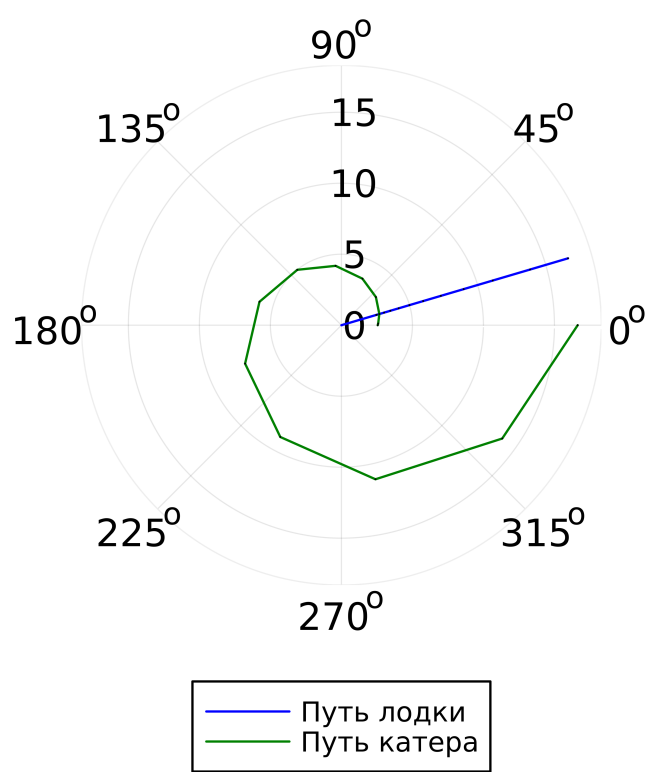


Рис.5 Случай один

Задача о погоне - случай 2

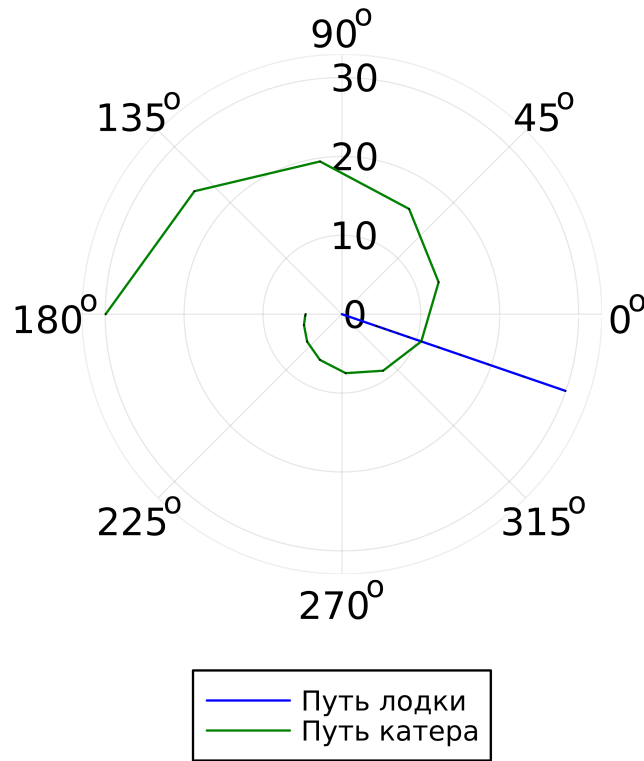


Рис.6 Случай два

Вывод:

Мы изучили основы языков программирования Julia и OpenModelica. Освоили библиотеки этих ЯП, которые используются для построения графиков и решения ДУ. Так же надо решили задачу о погоне.

Список литературы. Библиография

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