Отчёт по лабораторной работе №2

Задача о погоне

Низамова Альфия Айдаровна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Разобраться в алгоритме построения математической модели на примере задачи о погоне. Также необходимо провести теоритические рассуждение и вывести дифференциальные уравнения, с помощью которых мы сможем определить точку пересечения лодки и катера.

2 Библиография

1. Git - система контроля версий
2. Дифференциальные уравнения
3. Язык программирования - Julia
4. Установка Julia
5. Создание Plot в Julia

3 Задачи лабораторной работы

1. Изучить условие задачи о погоне
2. Провести рассуждения и вывести дифференциальные уравнения
3. Построить траекторию движение катера и лодки для двух случаев
4. Определить по графику точку пересечения катера и лодки

4 Выполнение лабораторной работы

1. Определила номер своего варинта по формуле: ((ст_билет)mod70) + 1 = (1032201670 mod 70) + 1 = 2

2. Учитывая, что лодка обнаруживается на расстоянии 12 км от катера, а скорость катера в 4 раза больше скорости браконьерской лодки (вариант 2), провела вычисления и получила начальные значения:

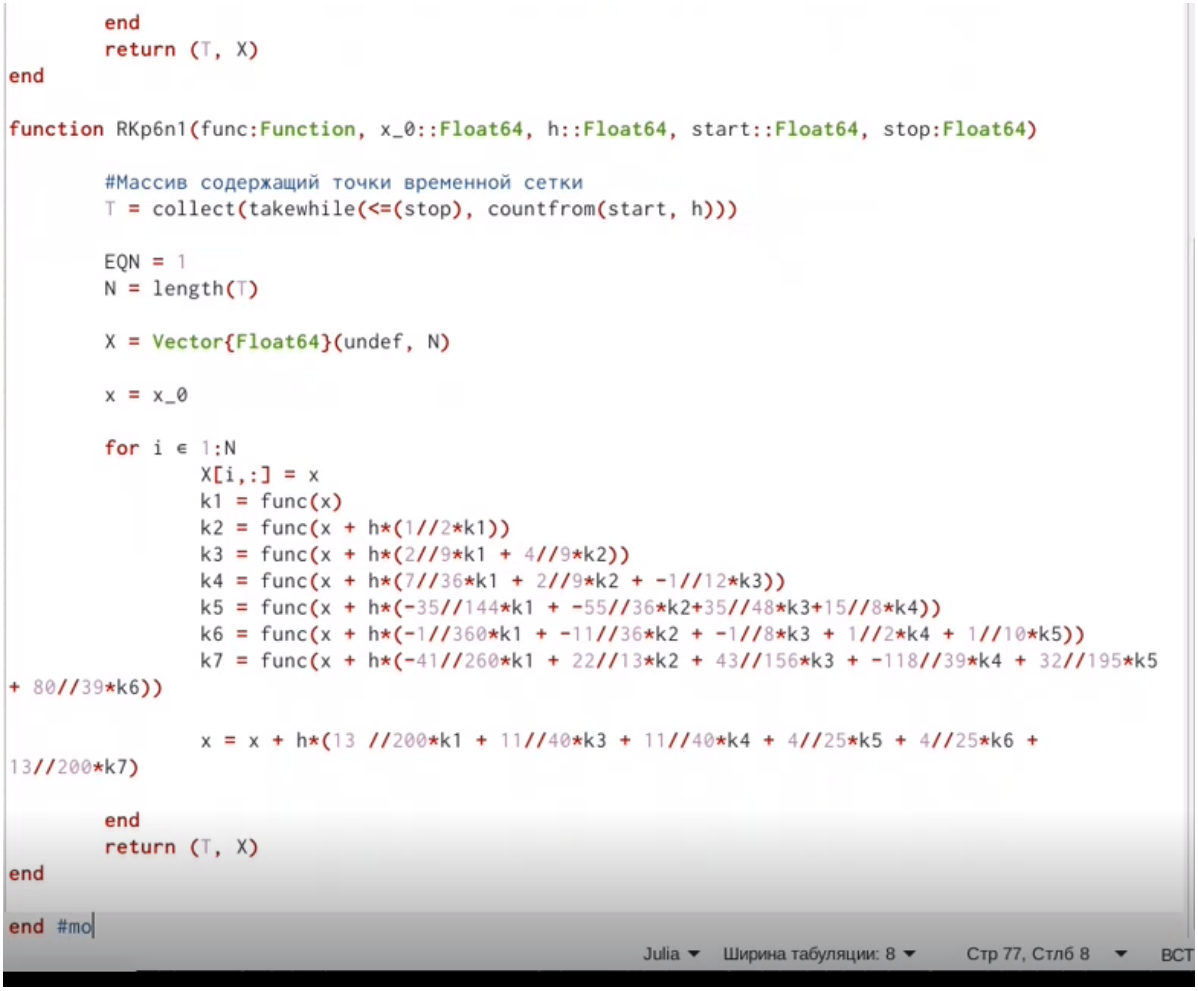
r0 = 0 x1 = 2.4
r0 = -pi x2 = 4

3. Переходим к написанию кода на языке программирования Julia. Написала код для метода Рунгк-Кутты с постоянным шагом интегрирования

```
#Метод Рунге-Кутты с постоянным шагом интегрирования. Результаты вычислений и моменты времени
записываются в массивы и возвращаются

module RK

using Base.Iterators: countfrom takewhile
using StaticArrays: SVector

function RKp6n1(func::Function, x_0::SVector, h::Float64, start::Float64, stop::Float64)

        #Массив содержащий точки временной сетки
        T = collect(takewhile(<=(stop), countfrom(start, h)))

        EQN = length(x_0)
        N = length(T)

        X = Matrix{Float}(undef, N, EQN)

        #В массиве X N строк и EQN столбцов

        x = SVector{EQN}(x_0)

        for i ∈ 1:N
        X[i,:] = x
        k1 = func(x)
        k2 = func(x + h*(1//2*k1))
        k3 = func(x + h*(2//9*k1 + 4//9*k2))
        k4 = func(x + h*(7//36*k1 + 2//9*k2 + -1//12*k3))
        k5 = func(x + h*(-35//144*k1 + -55//36*k2+35//48*k3+15//8*k4))
        k6 = func(x + h*(-1//360*k1 + -11//36*k2 + -1//8*k3 + 1//2*k4 + 1//10*k5))
        k7 = func(x + h*(-41//260*k1 + 22//13*k2 + 43//156*k3 + -118//39*k4 + 32//195*k5 +
80//39*k6))

        x = x + h*(13 //200*k1 + 11//40*k3 + 11//40*k4 + 4//25*k5 + 4//25*k6 + 13//200*k7)

        X[i,:] = x
        k1 = func(x)
        k2 = func(x + h*(1//2*k1))
        k3 = func(x + h*(2//9*k1 + 4//9*k2))
        k4 = func(x + h*(7//36*k1 + 2//9*k2 + -1//12*k3))
        k5 = func(x + h*(-35//144*k1 + -55//36*k2+35//48*k3+15//8*k4))
        k6 = func(x + h*(-1//360*k1 + -11//36*k2 + -1//8*k3 + 1//2*k4 + 1//10*k5))
        k7 = func(x + h*(-41//260*k1 + 22//13*k2 + 43//156*k3 + -118//39*k4 + 32//195*k5 +
80//39*k6))

        x = x + h*(13 //200*k1 + 11//40*k3 + 11//40*k4 + 4//25*k5 + 4//25*k6 + 13//200*k7)
        |
Julia ▾  Ширина табуляции: 8 ▾  Стр 44, Стлб 9 ▾  ВСТ
```

Рис. 4.1: RK_1

Рис.1 “RK_1”



```
        end
        return (T, X)
end

function RKp6n1(func:Function, x_0::Float64, h::Float64, start::Float64, stop:Float64)

        #Массив содержащий точки временной сетки
        T = collect(takewhile(<=(stop), countfrom(start, h)))

        EQN = 1
        N = length(T)

        X = Vector{Float64}(undef, N)

        x = x_0

        for i ∈ 1:N
                X[i,:] = x
                k1 = func(x)
                k2 = func(x + h*(1//2*k1))
                k3 = func(x + h*(2//9*k1 + 4//9*k2))
                k4 = func(x + h*(7//36*k1 + 2//9*k2 + -1//12*k3))
                k5 = func(x + h*(-35//144*k1 + -55//36*k2+35//48*k3+15//8*k4))
                k6 = func(x + h*(-1//360*k1 + -11//36*k2 + -1//8*k3 + 1//2*k4 + 1//10*k5))
                k7 = func(x + h*(-41//260*k1 + 22//13*k2 + 43//156*k3 + -118//39*k4 + 32//195*k5
+ 80//39*k6))

                x = x + h*(13 //200*k1 + 11//40*k3 + 11//40*k4 + 4//25*k5 + 4//25*k6 +
13//200*k7)

        end
        return (T, X)
end

end #mo
```


Рис. 4.2: RK_2

Рис.2 “RK_1”

4. Рассматриваем два случая.

   Первый: начальное значение: 2.4 сохраняем изображение: lab2_1.png

```julia
using Plots

include("RK.jl")

function F(r)
  k = 4.0
  return r / sqrt(k^2-1)
end

const r₀ = 2.4
const h = 0.01
const θ₀ = 0.0
const θ₁ = 4π

θ, R = RK.RKp6n1(F, r₀, h, θ₀, θ₁)


plt = plot(
  proj = :polar,
  aspect_ratio=:equal,
  dpi=300,
  title="lab2",
  legend=true)



plot!(
  plt,
  θ,
  R,
  label="Траектория катера",
  color=:red)

plot!(
  plt,
  [1, 1]*19/25*pi,
  [0, 100],
  label = "Траектория лодки",
  color=:blue)


savefig(plt, "lab2.png")
```

Рис. 4.3: Code_1

Рис.3 “Code_1”

Получаем следующий график:

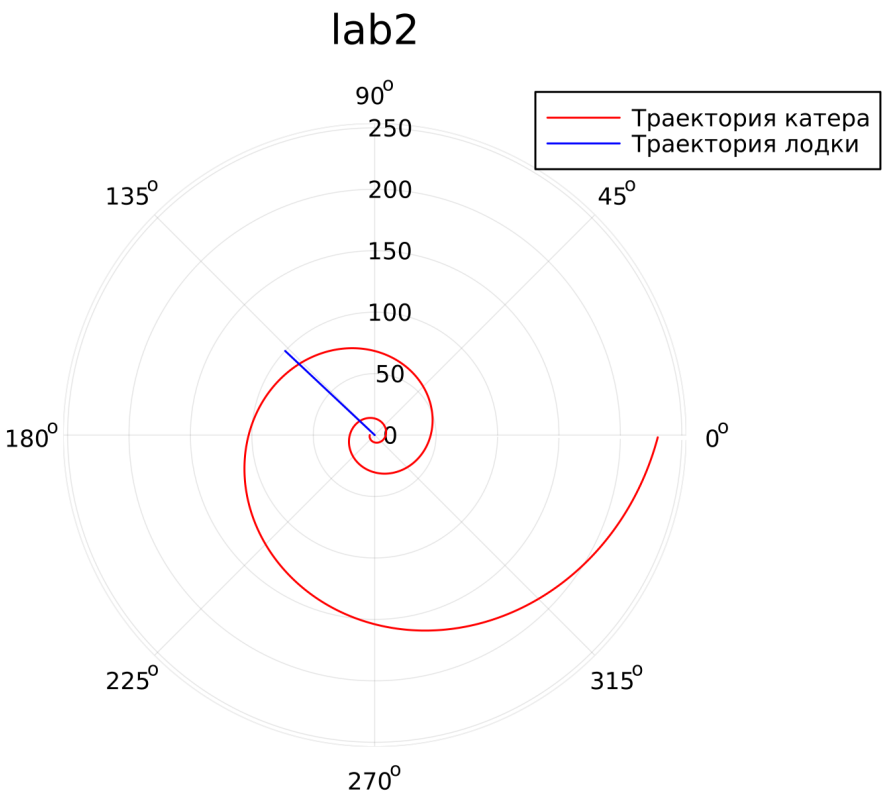


Рис. 4.4: График для первого случая

Рис.4 “График для первого случая”

```
Второй:
начальное значение: 4
сохраняем изображение: lab2.png
```

```julia
using Plots

include("RK.jl")

function F(r)
  k = 4.0
  return r / sqrt(k^2-1)
end

const r₀ = 4.0
const h = 0.01
const θ₀ = -π
const θ₁ = 4π

θ, R = RK.RKp6n1(F, r₀, h, θ₀, θ₁)


plt = plot(
  proj = :polar,
  aspect_ratio=:equal,
  dpi=300,
  title="lab2",
  legend=true)



plot!(
  plt,
  θ,
  R,
  label="Траектория катера",
  color=:red)

plot!(
  plt,
  [1, 1]*19/25*pi,
  [0, 100],
  label = "Траектория лодки",
  color=:blue)


savefig(plt, "lab2.png")
```

Рис. 4.5: Code_2

Рис.5 “Code_2”

Получаем следующий график:

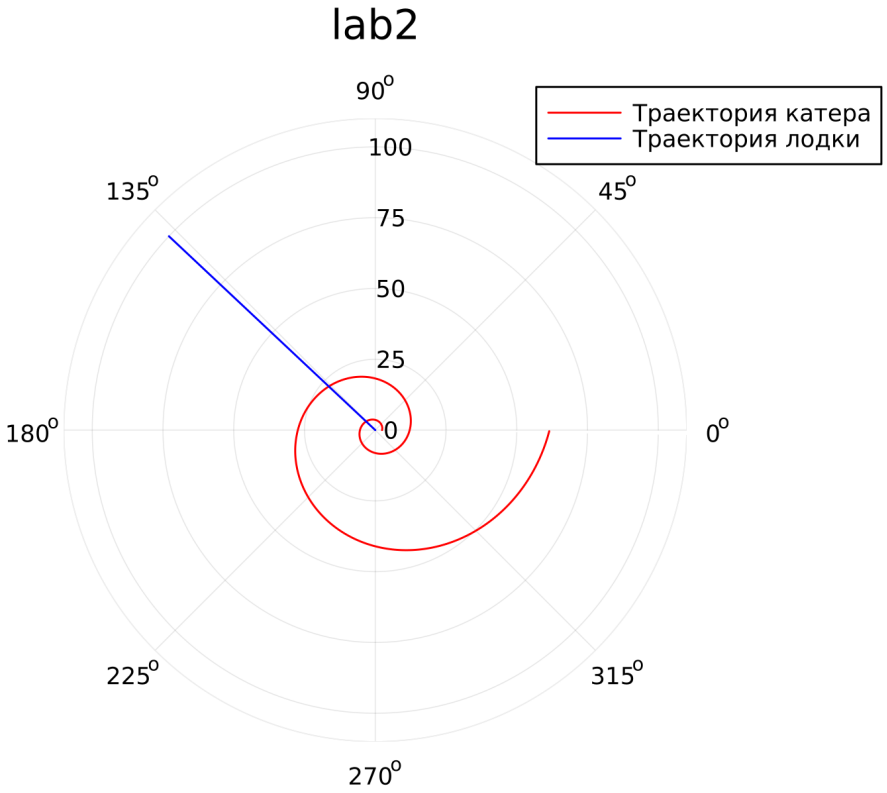


Рис. 4.6: График для второго случая

Рис.4 “График для второго случая”

6. Исходя из полученных графиков, мы видим, что в первом случае катер быстрее догонит лодку с браконьерами (при начальном значении 2,4). Точка пересечения красного и синего графиков и есть ответ.

5 Выводы

В ходе лабораторной работы нам удалось рассмотреть задачу о погоне,составить и решить дифференциальные уравнения. Смоделировать ситуацию и сделать вывод о том, что в первом случае погоня завершиться раньше.