

Лабораторная работа №2

Задача о погоне

Ильин Андрей Владимирович

Содержание

Цель работы	4
Задачи	5
Термины	6
Теоретическая справка	7
Выполнение лабораторной работы.	8
Листинг	14
Анализ результатов	16
Выводы	17
Библиография	18

Список иллюстраций

1	Установка пакета ‘Plots’	8
2	Начало написания скрипта	9
3	Уравнение кривой	9
4	Открытие цикла	10
5	Выбор начального условия	10
6	Выбор начального условия	11
7	Скрипт для построения графиков	11
8	Кривая погони №1	12
9	Кривая погони №2	13

Цель работы

Решить задачу о погоне. Смоделировать кривую погони средствами Julia и OpenModelica.

Задачи

1. Провести аналогичные рассуждения и вывод дифференциальных уравнений, если скорость катера больше скорости лодки в n раз.
2. Построить траекторию движения катера и лодки для двух случаев.
3. Определить по графику точку пересечения катера и лодки

Термины

- Задача о погоне — классическая задача из области дифференциальных уравнений. [1]
- Кривая погони — кривая, представляющая собой решение задачи о «погоне». [2]
- Julia — это открытый свободный высокопроизводительный динамический язык высокого уровня, созданный специально для технических (математических) вычислений. Его синтаксис близок к синтаксису других сред технических вычислений, таких как Matlab и Octave. [3]
- OpenModelica — свободное открытое программное обеспечение для моделирования, симуляции, оптимизации и анализа сложных динамических систем. Основано на языке Modelica. [4]

Теоретическая справка

Для построения кривой погони, нам необходимо знать начальные условия и уравнения кривой. Ориентируясь на рассуждения из пособия к лабораторной работы[1] можно вывести общие формулы. Благодаря общим формулам можно будет написать программу, которая будет строить разные кривые погони в зависимости от исходных данных (расстояния и разницы в скорости).

Пусть

- n - разница в скорости, то есть скорость катера в n раз больше лодки;
- a - расстояние между катером и лодкой в момент рассеивания тумана.

Тогда общие начальные условия для первого случая выглядят следующим образом:

$$\begin{cases} \theta_0 = 0 \\ r_0 = \frac{a}{(n+1)} \end{cases}$$

Для второго случая:

$$\begin{cases} \theta_0 = -\pi \\ r_0 = \frac{a}{(n-1)} \end{cases}$$

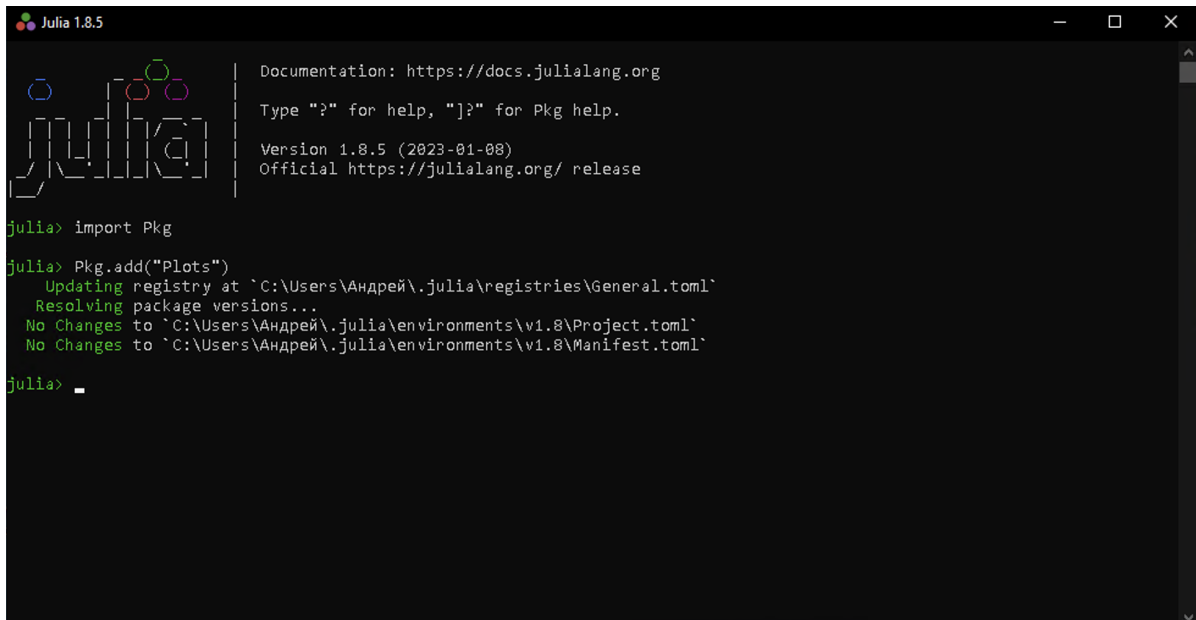
Уравнение кривой в общем случае выглядит следующим образом:

$$r(\theta) = r_0 e^{\frac{\theta}{\sqrt{n^2-1}}}$$

Выполнение лабораторной работы.

1. Установим пакет в Julia необходимый для построения графика. (Рис. 1)

```
import Pkg  
Pkg.add("Plots")
```

A screenshot of the Julia 1.8.5 REPL window. The window title is 'Julia 1.8.5'. On the left, there is a small logo consisting of a grid of colored squares. On the right, there is a sidebar with text: 'Documentation: https://docs.julialang.org', 'Type "?" for help, "]?" for Pkg help.', 'Version 1.8.5 (2023-01-08)', and 'Official https://julialang.org/ release'. The main area shows the following commands and output:

```
julia> import Pkg  
julia> Pkg.add("Plots")  
Updating registry at `C:\Users\Андрей\.julia\registries\General.toml`  
Resolving package versions...  
No Changes to `C:\Users\Андрей\.julia\environments\v1.8\Project.toml`  
No Changes to `C:\Users\Андрей\.julia\environments\v1.8\Manifest.toml`  
julia> _
```

Рис. 1: Установка пакета 'Plots'

2. Приступим к написанию скрипта. Первым делом подключим пакет "Plots" и объявим необходимые константы. Константа a и n - являются входными данными, $\theta_{PrayDeg}$ - сторона в которую поплыла лодка, $d\theta$ - шаг для

равномерного разбиения периода, maxTheta - длина периода построения, `cases` - содержит два кейса, по которым мы будем итерироваться. (Рис. 2)

`using ...` # используется для подключения пакета
`const ...` # используется для объявления константы

```
1 using Plots
2
3 const a = 19.1
4 const n = 5.2
5 const thetaPrayDeg = 320
6 const dTheta = 0.01
7 const maxTheta = 4π
8 const cases = ["First", "Second"]
```

Рис. 2: Начало написания скрипта

3. Напишем функцию, которая будет являться уравнением нашей кривой. (Рис. 3)

синтаксис функции
`function funcName(args)`
 #script
`end`

```
10 function F(theta)
11     return r0 * exp.(theta / sqrt.(n^2 - 1))
12 end
13
```



The screenshot shows the Julia REPL output for package management: `julia> import Pkg`, `julia> Pkg.add("Plots")`, `Updating registry`, `Resolving package`, `No Changes to 'C:\...`

Рис. 3: Уравнение кривой

4. Далее откроем цикл, внутри которого будем итерироваться по двум кейсам (в данных кейсах разные начальные условия). Для написания цикла используем следующую конструкцию (Рис. 4):

```
# синтаксис цикла
for item in list
    #script
end
```

```
14 for case in cases
```

Рис. 4: Открытие цикла

5. Внутри цикла сначала определим начальные условия в зависимости от кейса. Для этого воспользуемся ветвлением (Рис. 5):

```
# синтаксис ветвления
if condition
    ...
else
    ...
end
```

```
15 global r0 = -1
16 theta0 = -1
17
18 if case=="First"
19     r0 = a / (n + 1)
20     theta0 = 0
21 else
22     r0 = a / (n - 1)
23     theta0 = -π
24 end
```

Рис. 5: Выбор начального условия

6. После этого нам необходимо равномерно разбить наш период на точки (чем меньше расстояние между точками, тем кривая погони). Также необходимо перевести градус побега лодки в радианы с учетом начального условия. (Рис. 6)

```
# синтаксис разбиения
```

```
linspace = start:step:end
```

```
26 theta1 = theta0 + maxTheta
27 thetaHunt = theta0:dTheta:theta1
28 thetaPray = thetaPrayDeg * π / 180 + 2 * theta0
```

Рис. 6: Выбор начального условия

7. Построим необходимые графики и сохраним “полотно”. (Рис. 7)

```
30 plt = plot(proj=:polar, aspect_ratio=:equal, dpi=500, title="Lab02" * case * "Case", legend=true)
31 plot!(plt, [theta0, theta0], [a, F(theta0)], label=false, color=:red)
32 plot!(plt, thetaHunt, F, label="Траектория охраны", color=:red)
33 plot!(plt, [0, thetaPray], [0, F(thetaPray) + 20], label="Траектория браконьеров", color=:green)
34
35 plot!(plt, [theta0], [a], seriestype=:scatter, label="Точка начала (охрана)", color=:red)
36 plot!(plt, [0], [0], seriestype=:scatter, label="Точка начала (браконьеры)", color=:green)
37 plot!(plt, [thetaPray], [F(thetaPray)], seriestype=:scatter, label="Точка пересечения", color=:blue)
38
39 savefig(plt, "Lab02" * case * "Case.png")
```

Рис. 7: Скрипт для построения графиков

8. Просмотрим вывод программы (Рис. 8 - 9)

Lab02FirstCase

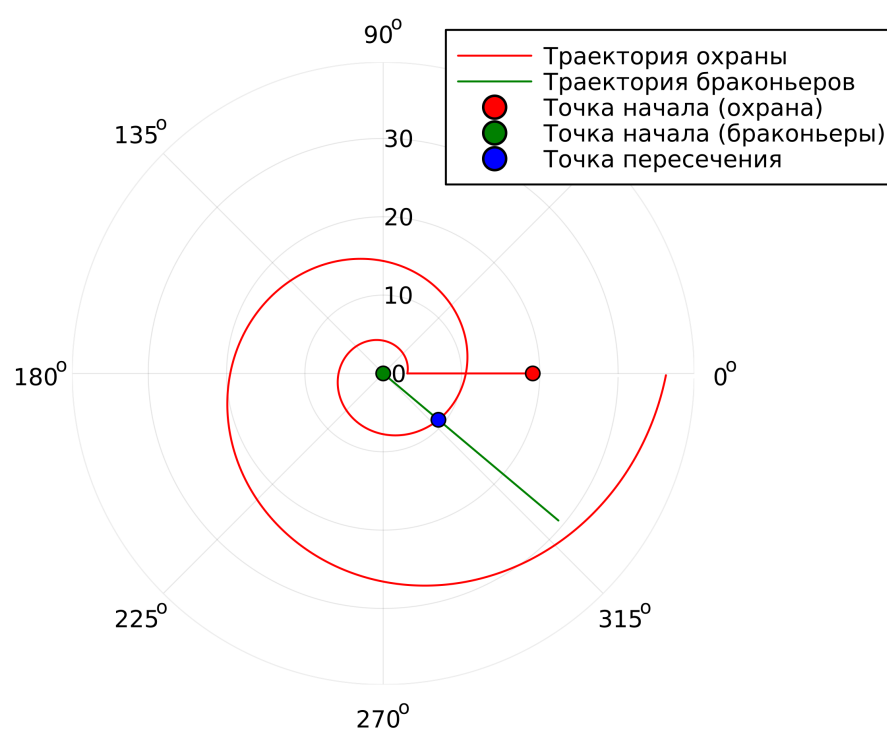


Рис. 8: Кривая погони №1

Lab02SecondCase

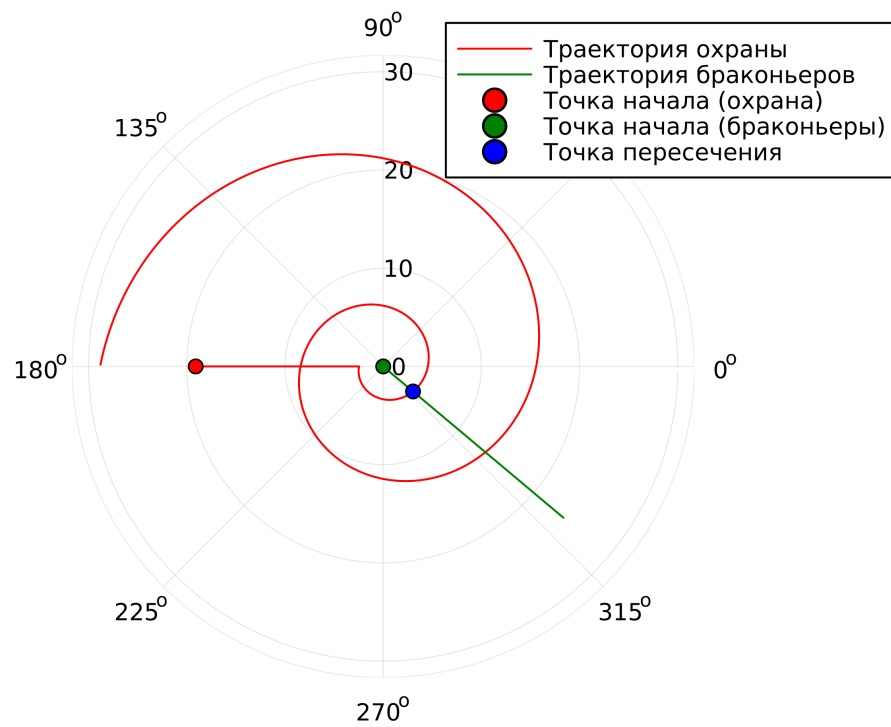


Рис. 9: Кривая погони №2

9. Построение кривой погони для данной задачи выполнять на OpenModelica стандартными инструментами невозможно. Так как в стандартные инструменты не выходит построение графиков в полярных координатах, в связи с этим данную лабораторную работу выполнять на OpenModelica нет необходимости.

ЛИСТИНГ

```
using Plots

const a = 19.1
const n = 5.2
const thetaPrayDeg = 320
const dTheta = 0.01
const maxTheta = 4pi
const cases = ["First", "Second"]

function F(theta)
    return r0 * exp.(theta / sqrt.(n^2 - 1))
end

for case in cases
    global r0 = -1
    theta0 = -1

    if case=="First"
        r0 = a / (n + 1)
        theta0 = 0
    else
        r0 = a / (n - 1)
```

```

    theta0 = -pi
end

theta1 = theta0 + maxTheta
thetaHunt = theta0:dTheta:theta1
thetaPray = thetaPrayDeg * pi / 180 + 2 * theta0

plt = plot(proj=:polar, aspect_ratio=:equal, dpi=500, title="Lab02" * case * "Case", legend=true)
plot!(plt, [theta0, theta0], [a, F(theta0)], label=false, color=:red)
plot!(plt, thetaHunt, F, label="Траектория охраны", color=:red)
plot!(plt, [0, thetaPray], [0, F(thetaPray) + 20], label="Траектория браконьеров", color=:green)

plot!(plt, [theta0], [a], seriestype=:scatter, label="Точка начала (охрана)", color=:red)
plot!(plt, [0], [0], seriestype=:scatter, label="Точка начала (браконьеры)", color=:green)
plot!(plt, [thetaPray], [F(thetaPray)], seriestype=:scatter, label="Точка пересечения", color=:blue)

savefig(plt, "Lab02" * case * "Case.png")
end

```

Анализ результатов

Работа выполнена без непредвиденных проблем в соответствии с руководством.

Ошибок и сбоев не произошло.

Выводы

Мы улучшили практические навыки в области дифференциальных уравнений, а также приобрели навыки моделирования на Julia.

Библиография

- [illegible]