

Отчет по лабораторной работе №2

Задача о погоне

Кузнецова София Вадимовна

Содержание

Выполнение лабораторной работы	5
Цель работы	6
Задание	7
Выполнение лабораторной работы	8
Моделирование с помощью Julia	11
Выводы	16
Список литературы	17

Список иллюстраций

0.1	Получение нужного номера варианта	8
0.1	Скачивание Julia	11
0.2	Процесс запуска Julia	11
0.3	Скачивание необходимых для работы пакетов	12
0.4	Скачивание необходимых для работы пакетов	12
0.5	Запуск кода	13
0.6	Случай №1	14
0.7	Случай №2	15

Список таблиц

Выполнение лабораторной работы

Цель работы

Решить задачу о погоне и изучить основы языка программирования Julia.

Задание

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.

Выполнение лабораторной работы

Расчитаем свой вариант по формуле и получаем наш вариант №59.

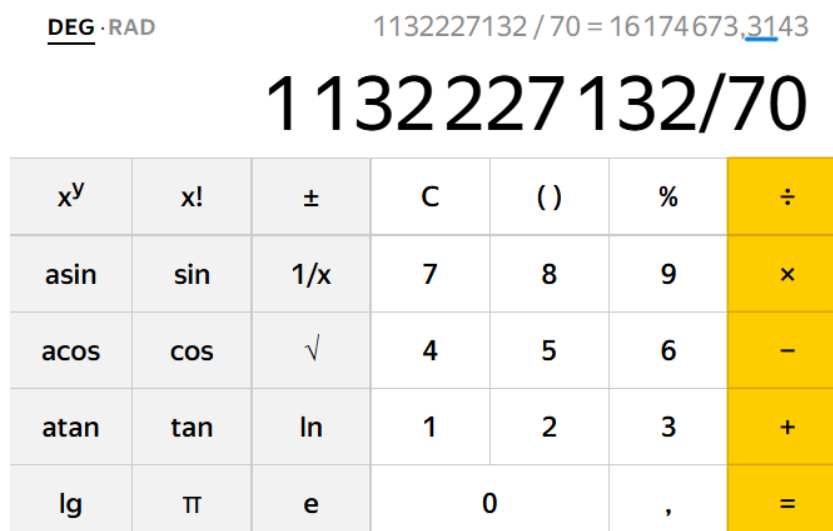


Рис. 0.1: Получение нужного номера варианта

1. На море в тумане катер береговой охраны преследует лодку браконьеров. Через определенный промежуток времени туман рассеивается, и лодка обнаруживается на расстоянии 20,3 км от катера. Затем лодка снова скрывается в тумане и уходит прямолинейно в неизвестном направлении. Известно, что скорость катера в 5,2 раза больше скорости браконьерской лодки.
2. 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 (20,3; 0). Обозначим скорость лодки v .

3. 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.
4. Пусть время t - время, через которое катер и лодка окажутся на одном расстоянии от начальной точки.

$$t = \frac{x}{v}$$

$$t = \frac{20,3 - x}{5,2v}$$

$$t = \frac{20,3 + x}{5,2v}$$

Из этих уравнений получаем объединение двух уравнений:

$$\begin{cases} \frac{x}{v} = \frac{20,3-x}{5,2v} \\ \frac{x}{v} = \frac{20,3+x}{5,2v} \end{cases}$$

Решая это, получаем два значения для x :

$$x_1 = 3,27419355$$

$$x_2 = 4,83333333$$

$$v_\tau$$

– тангенциальная скорость

$$v$$

– радиальная скорость

$$v = \frac{dr}{dt}$$

$$v_\tau = \sqrt{((5,2 * v)^2 - v^2)} = \frac{\sqrt{651} * v}{5}$$

$$\left\{ \begin{array}{l} \frac{dr}{dt} = v \\ r \frac{d\theta}{dt} = \frac{\sqrt{651} * v}{5} \end{array} \right.$$

$$\left\{ \begin{array}{l} \theta_0 = 0 \\ r_0 = x_1 = 3,27419355 \end{array} \right.$$

или

$$\left\{ \begin{array}{l} \theta_0 = -\pi \\ r_0 = x_2 = 4,83333333 \end{array} \right.$$

Итоговое уравнение после того, как убрали производную по t:

$$\frac{dr}{d\theta} = \frac{5r}{\sqrt{651}}$$

Моделирование с помощью Julia

1. Скачиваем Julia.

```
C:\Users\sofik>winget install julia -s msstore
Найдено Julia [9NJNMW8PVKMN] Версия Unknown
Этот пакет предоставляется через Microsoft Store. Программе winget может потребоваться получить пакет
в Microsoft Store от имени текущего пользователя.
Соглашения для Julia [9NJNMW8PVKMN] Версия Unknown
Версия: Unknown
Издатель: Julia Computing, Inc.
URL-адрес издателя: https://julialang.org/
Лицензия: ms-windows-store://pdp/?ProductId=9NJNMW8PVKMN
URL-адрес заявления о конфиденциальности: https://juliacomputing.com/privacy/
Соглашения:
  Category: Developer tools
  Pricing: Free
  Free Trial: No
  Terms of Transaction: https://aka.ms/microsoft-store-terms-of-transaction
  Seizure Warning: https://aka.ms/microsoft-store-seizure-warning
  Store License Terms: https://aka.ms/microsoft-store-license

Издатель требует, чтобы вы просмотрели указанную выше информацию и приняли соглашения перед установкой
.
Вы согласны с условиями?
[Y] Да [N] Нет: Y
Запуск установки пакета... 100%
Успешно установлено
```

Рис. 0.1: Скачивание Julia

2. Запускаем Julia.

```
Documentation: https://docs.julialang.org
Type "?" for help, "]"? for Pkg help.

Version 1.11.3 (2025-01-21)
Official https://julialang.org/ release

julia> |
```

Рис. 0.2: Процесс запуска Julia

3. Скачиваем необходимые для работы пакеты.

```

Downloaded artifact: Qt6ShaderTools
Downloaded artifact: Qt6Declarative
Downloaded artifact: Wayland_protocols
Downloaded artifact: Graphite2
Downloaded artifact: libass
Downloaded artifact: Pixman
Downloaded artifact: XNL2
Downloaded artifact: Gettext
Downloaded artifact: OpenSSL
Downloaded artifact: FFmpeg
Downloaded artifact: libgcrypt
Downloaded artifact: Xorg_keyboard_config
Downloaded artifact: LLVMOpenMP
Downloaded artifact: LAME
Downloaded artifact: Libiconv
Downloaded artifact: Qt6Base
Downloaded artifact: Glib
Downloaded artifact: libvorbis
Updating 'C:\Users\sofik\.julia\environments\v1.11\Project.toml'
[91a5bcd] + Plots v1.40.0
Updating 'C:\Users\sofik\.julia\environments\v1.11\Manifest.toml'
[66dad0bd] + AliasTables v1.1.3
[d1d4a3ce] + BitFlags v0.1.9
[944b1d66] + CodecZlib v0.7.8
[35d6a980] + ColorSchemes v3.29.0

```

Рис. 0.3: Скачивание необходимых для работы пакетов

```

Precompiling project...
262 dependencies successfully precompiled in 434 seconds. 191 already precompiled.
2 dependencies had output during precompilation:
LinearSolve
  Downloading artifact: MKL
MKL_jll
  Downloading artifact: IntelOpenMP
julia> using Plots

```

Рис. 0.4: Скачивание необходимых для работы пакетов

4. Код для файла lab2.jl:

```

import Pkg Pkg.add("Plots") Pkg.add("DifferentialEquations")
using Plots using DifferentialEquations

const a = 10.5 const n = 4.3 const r0 = a / (n + 1) const r0_2 = a / (n - 1) const T
= (0, 2 * pi) const T_2 = (-pi, pi)

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

function F2(u, p, t) # Function for converging spiral damping = 0.01 # Adjust this
value to control the damping strength return (u / sqrt(n * n - 1)) - damping * u # Apply
dampening to radius end

problem = ODEProblem(F, r0, T) result = solve(problem, abstol=1e-8, reltol=1e-8)
dxR = rand(1:size(result.t)[1]) rAngels = [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="Случай №1", legend=:outerbottom)

```

```

plot!(plt, [rAngels[1], rAngels[2]], [0.0, result.u[end]], label="Путь лодки", color=:red,
lw=1) scatter!(plt, rAngels, result.u, label="", mc=:blue, ms=0.0005) plot!(plt, result.t,
result.u, xlabel="theta", ylabel="r(t)", label="Путь катера", color=:blue, lw=1)
scatter!(plt, result.t, result.u, label="", mc=:green, ms=0.0005)

savefig(plt, "lab2_01.png")

problem = ODEProblem(F2, r0_2, T_2) result = solve(problem, abstol=1e-8,
reitol=1e-8)

dxR = rand(1:size(result.t)[1]) rAngels = [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="Случай №2", legend=:outerbottom)
plot!(plt1, [rAngels[1], rAngels[2]], [0.0, result.u[end]], label="Путь лодки", color=:red,
lw=1) scatter!(plt1, rAngels, result.u, label="", mc=:blue, ms=0.0005) plot!(plt1,
result.t, result.u, xlabel="theta", ylabel="r(t)", label="Путь катера", color=:blue, lw=1)
scatter!(plt1, result.t, result.u, label="", mc=:green, ms=0.0005)

savefig(plt1, "lab2_02.png")

```

6. Запускаем код.

```

Result 1 - Time (theta) values (result.t):
[0.0, 0.00047791009751875, 0.33600002675000003, 0.629783597783517, 1.0087617156689782, 2.008773945870792, 2.7861783641758966, 3.5562140838520566, 4.388978215658318, 5.245997657230685, 6.3403389940833126, 6.2
03332672795069]

Result 1 - Radius values (result.u):
[1.0811320934716881, 2.000238982772992, 2.1478956505590717, 2.4159228214083904, 2.7748861493948897, 3.2427717445898979, 3.8514366397518837, 4.636729571819248, 5.6475395228958795, 6.041115884395885, 6.6898888477
7003, 8.000339850000000]

Result 2 - Time (theta) values (result.t):
[-3.151924631889792, -3.0900021688377345, -2.8156185888099983, -2.3330039986099588, -1.765183382166897, -1.1251884768657564, -0.41731263897991566, 0.34714613511328283, 1.1618669365060587, 2.015705839189326, 2.
800781535000000, 3.515928633897923]

Result 2 - Radius values (result.u):
[2.1818218181818182, 2.2352821880218806, 3.439713840019726, 3.886697676881348, 4.02169027981493, 5.153233927861971, 6.183234632879863, 7.276422136076403, 8.061046376476006, 10.02083132476407, 13.08081183000
59, 18.206181009782338]

```

Рис. 0.5: Запуск кода

7. Просмотр результата работы.

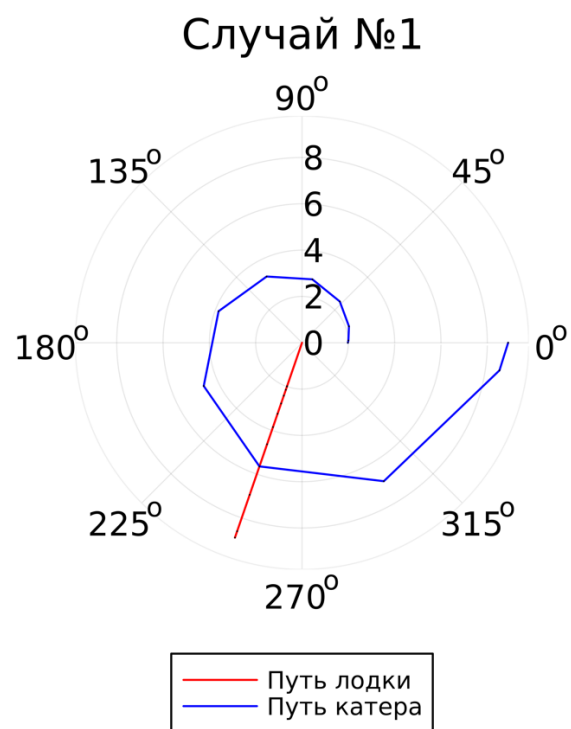


Рис. 0.6: Случай №1

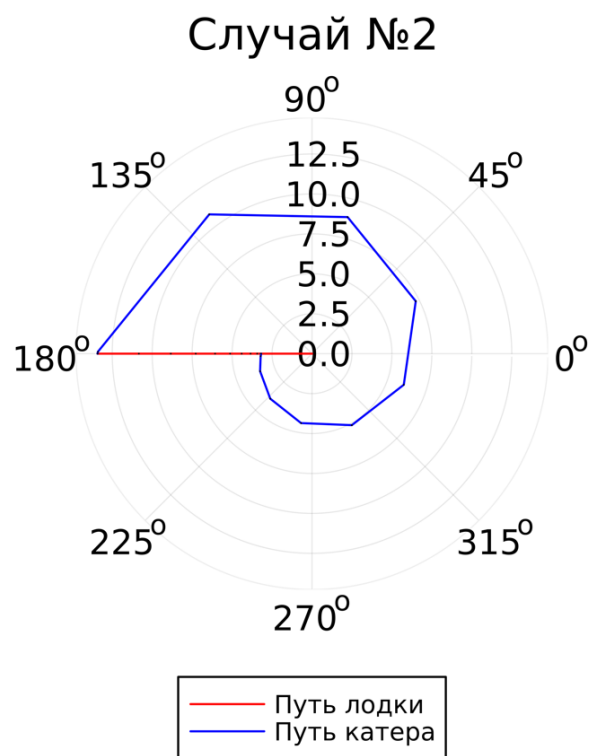


Рис. 0.7: Случай №2

Выводы

Были изучены основы языка программирования Julia и его библиотеки, которые используются для построения графиков и решения дифференциальных уравнений. А также решили задачу о погоне.

Список литературы

- [1] Документация по Julia: <https://docs.julialang.org/en/v1/>
- [2] Решение дифференциальных уравнений: <https://www.wolframalpha.com/>