

# **Математическое моделирование**

**Лабораторная работа № 2**

Сергей Павленко

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>6</b>
<b>2</b>	<b>Задание</b>	<b>7</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
3.1	Определение радиуса перехода к «обходу полюса» . . . . .	8
3.2	Разложение скорости катера . . . . .	9
3.3	Система дифференциальных уравнений . . . . .	9
3.4	Условие задачи . . . . .	10
<b>4</b>	<b>Полярные траектории: катер (ODE) и лодка (аналитически)</b>	<b>11</b>
4.1	Инициализация проекта и загрузка пакетов . . . . .	11
4.2	Параметры задачи . . . . .	12
4.3	Определение модели . . . . .	12
4.4	Запуск 1: $r_0 = s/(n+1)$ , $t \in (1e-9, 8)$ . . . . .	14
4.5	Запуск 2: $r_0 = s/(n-1)$ , $t \in (1e-9, 15)$ . . . . .	14
<b>5</b>	<b>Параметрическое исследование: катер (ODE) и лодка (аналитически) в полярных координатах</b>	<b>16</b>
5.1	Активация проекта и загрузка пакетов . . . . .	16
5.2	Определение модели . . . . .	17
5.3	Определение параметров в Dict . . . . .	17
5.4	Функция-обертка для запуска одного эксперимента . . . . .	18
5.5	Запуск базового эксперимента (кэширование) . . . . .	20
5.6	Визуализация базового эксперимента . . . . .	21
5.7	Второй базовый эксперимент (как у тебя во 2-й части): case=:minus, tmax=15 . . . . .	22
5.8	Параметрическое сканирование . . . . .	23
5.9	Запуск всех экспериментов и сбор результатов . . . . .	24
5.10	Анализ и визуализация результатов сканирования . . . . .	26
5.11	Бенчмаркинг с разными параметрами . . . . .	28
5.12	Сохранение всех результатов . . . . .	30
5.13	Анализ результатов моделирования . . . . .	31
5.14	Базовые эксперименты . . . . .	32
5.15	Параметрическое сканирование по $n$ . . . . .	34
5.16	Анализ метрики scale_ratio . . . . .	35

5.17 Время вычислений . . . . .	36
<b>6 Выводы</b>	<b>37</b>
<b>Список литературы</b>	<b>38</b>

## Список иллюстраций

5.1	Базовый эксперимент (case=plus) . . . . .	32
5.2	Базовый эксперимент (case=minus) . . . . .	33
5.3	Сканирование траекторий катера . . . . .	34
5.4	Зависимость scale_ratio от n . . . . .	35
5.5	Зависимость времени вычисления от n . . . . .	36

## **Список таблиц**

# 1 Цель работы

Рассмотрим пример построения математической модели, которая помогает выбрать оптимальную стратегию в задачах поиска и преследования.

В качестве иллюстрации возьмём ситуацию: в тумане катер береговой охраны ведёт погоню за лодкой браконьеров. Через некоторое время туман рассеивается, и лодка фиксируется на расстоянии  $k$  км от катера. Далее лодка вновь исчезает в тумане и продолжает движение прямолинейно в неизвестном направлении. Известно, что скорость катера в  $n$  раз больше скорости лодки. Требуется определить траекторию движения катера, обеспечивающую перехват лодки.

## 2 Задание

1. Выполнить рассуждения и вывести дифференциальные уравнения при условии, что скорость катера превышает скорость лодки в  $n$  раз.
2. Построить траектории движения катера и лодки для двух вариантов начального положения.
3. По построенным графикам определить точку встречи (пересечения траекторий).

## 3 Выполнение лабораторной работы

Положим момент обнаружения за начало отсчёта:  $t_0 = 0$ . Точку обнаружения лодки принимаем за начало координат. В этот момент лодка находится в точке  $X_0 = 0$ , а катер — на расстоянии  $k$  относительно лодки.

Перейдём к полярной системе координат. Пусть полюс — это точка обнаружения лодки ( $r = 0$ ), а полярная ось  $r$  направлена в сторону положения катера в момент обнаружения.

### 3.1 Определение радиуса перехода к «обходу полюса»

Найдём расстояние  $x$  от полюса, при достижении которого катер и лодка окажутся на одном радиусе. За время  $t$  лодка проходит путь  $x$  со скоростью  $v$ , а катер — путь  $x \pm k$  со скоростью  $nv$  (знак зависит от того, по какую сторону полюса был катер).

Приравниваем времена движения: - для варианта «plus»:  $\frac{x}{v} = \frac{x+k}{nv}$ , - для варианта «minus»:  $\frac{x}{v} = \frac{x-k}{nv}$ .

Отсюда получаем два стартовых радиуса:

- $x_1 = \frac{k}{n+1}$  при  $\theta = 0$ ,
- $x_2 = \frac{k}{n-1}$  при  $\theta = -\pi$ .

Далее, когда катер оказывается на том же радиусе, что и лодка, он должен перейти от прямолинейного движения к движению вокруг полюса, одновре-



менно удаляясь от него с той же радиальной скоростью, что и лодка (чтобы не «проигрывать» по удалению).

### 3.2 Разложение скорости катера

Скорость катера разложим на две компоненты: - радиальная:  $v_r = \frac{dr}{dt}$ , - тангенциальная:  $v_t = r \frac{d\theta}{dt}$ .

Требуемое условие по радиальной скорости:

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

Полная скорость катера равна  $nv$ , а компоненты образуют прямоугольный треугольник, поэтому:

$$(nv)^2 = v_r^2 + v_t^2.$$

Подставляя  $v_r = v$ , получаем:

$$v_t = v\sqrt{n^2 - 1}.$$

То есть:

$$r \frac{d\theta}{dt} = v\sqrt{n^2 - 1}.$$

### 3.3 Система дифференциальных уравнений

Таким образом, задача сводится к решению системы:

$$\begin{cases} \frac{dr}{dt} = v, \\ r \frac{d\theta}{dt} = v\sqrt{n^2 - 1}. \end{cases}$$

Начальные условия для двух сценариев:

### Сценарий 1

$$\begin{cases} \theta_0 = 0, \\ r_0 = \frac{k}{n+1}. \end{cases}$$

### Сценарий 2

$$\begin{cases} \theta_0 = -\pi, \\ r_0 = \frac{k}{n-1}. \end{cases}$$

Исключая параметр  $t$ , получаем уравнение траектории катера:

$$\frac{dr}{d\theta} = \frac{r}{\sqrt{n^2 - 1}}.$$

Решив его с соответствующим начальным условием, получаем искомую траекторию катера в полярных координатах.

## 3.4 Условие задачи

На море в тумане катер береговой охраны преследует лодку браконьеров. После рассеивания тумана лодка обнаружена на расстоянии 20 км от катера. Затем лодка снова скрывается и движется прямолинейно в неизвестном направлении.

Скорость катера в 5 раз больше скорости лодки, то есть  $k = 20$ ,  $n = 5$ .

Для моделирования процесса и построения графиков использовались внешние файлы с программным кодом:

## 4 Полярные траектории: катер (ODE) и лодка (аналитически)

**Цель:** построить траектории в полярных координатах для «катера» (решение ОДУ) и «лодки» (заданная прямая в декартовых координатах, переведённая в полярные).

### 4.1 Инициализация проекта и загрузка пакетов

```
using DrWatson
@quickactivate "project"

using DifferentialEquations
using Plots
using DataFrames
using JLD2

script_name = isempty(PROGRAM_FILE) ? "interactive" : splitext(basename(PROGRAM_FILE))
mkpath(plotsdir(script_name))
mkpath(datadir(script_name))
```

## 4.2 Параметры задачи

```
n = 5
s = 20
fi = 3/4 * pi

p = (n = n, s = s, fi = fi)
```

## 4.3 Определение модели

Катер:  $dr/d\theta = r / \sqrt{n^2 - 1}$  Здесь независимая переменная —  $\theta$  (мы используем  $t$  как  $\theta$ , как принято в ODEProblem).

```
function cutter_ode!(dr, r, p, □)
    dr[1] = r[1] / sqrt(p.n^2 - 1)
end
```

Лодка: линия  $x(t)=t$ ,  $y(t)=\tan(fi+\pi)*t$ , затем перевод в  $(r, \theta)$  В Julia правильнее использовать  $\text{atan}(y, x)$ , чтобы угол был в верном квадранте.

```
function boat_polar(t, p)
    k = tan(p.fi + pi)
    x = t
    y = k * t
    r = hypot(x, y)          # sqrt(x^2 + y^2)
    □ = atan(y, x)
    return r, □
end
```

Утилита: генерация траектории лодки для набора  $t$

```

function make_boat_curve(tgrid, p)
    r = VectorFloat64(undef, length(tgrid))
    ϕ = VectorFloat64(undef, length(tgrid))
    for (i, tt) in pairs(tgrid)
        ri, ϕi = boat_polar(tt, p)
        r[i] = ri
        ϕ[i] = ϕi
    end
    return r, ϕ
end

```

Утилита: один прогон (и график, и таблицы, и сохранение)

```

function run_case(case_name; r0, ϕspan=(0.0, 2pi), nϕ=10_000, tmin=1e-9, tmax=8.0, nt

```

— Катер (ODE) —

```

ϕgrid = collect(LinRange(ϕspan[1], ϕspan[2], nϕ))
prob = ODEProblem(cutter_ode!, [r0], ϕspan, p)
sol = solve(prob, Tsit5(), saveat=ϕgrid)

df_cutter = DataFrame(ϕ = sol.t, r = first.(sol.u))

```

— Лодка (аналитически) —

```

tgrid = collect(LinRange(tmin, tmax, nt))
r_boat, ϕ_boat = make_boat_curve(tgrid, p)
df_boat = DataFrame(t = tgrid, ϕ = ϕ_boat, r = r_boat)

```

— Визуализация —

```
plt = plot(sol, proj=:polar, label="катер", xlabel="", ylabel="r",
           title="Полярные траектории – $case_name", lw=2, legend=:topleft)
plot!(plt, r_boat, proj=:polar, label="лодка", lw=2)
```

— Сохранение —

```
savefig(plt, plotsdir(script_name, "polar_$case_name.png"))
@save datadir(script_name, "data_$case_name.jld2") df_cutter df_boat p r0 rspan n

return (plt=plt, df_cutter=df_cutter, df_boat=df_boat)
end
```

#### 4.4 Запуск 1: $r_0 = s/(n+1)$ , $t \in (1e-9, 8)$

```
case1_r0 = p.s / (p.n + 1)
res1 = run_case("r0=s_div_(n+1)"; r0=case1_r0, tmax=8.0, p=p)

println("Кейс 1 – первые 5 строк (катер):")
println(first(res1.df_cutter, 5))
println("\nКейс 1 – первые 5 строк (лодка):")
println(first(res1.df_boat, 5))
```

#### 4.5 Запуск 2: $r_0 = s/(n-1)$ , $t \in (1e-9, 15)$

```
case2_r0 = p.s / (p.n - 1)
res2 = run_case("r0=s_div_(n-1)"; r0=case2_r0, tmax=15.0, p=p)
```

```
println("\nКейс 2 – первые 5 строк (катер):")
println(first(res2.df_cutter, 5))
println("\nКейс 2 – первые 5 строк (лодка):")
println(first(res2.df_boat, 5))
```

(опционально) показать последний график в интерактивной среде

```
res2.plt
```

## 5 Параметрическое исследование: катер (ODE) и лодка (аналитически) в полярных координатах

### 5.1 Активация проекта и загрузка пакетов

```
using DrWatson
@quickactivate "project"

using DifferentialEquations
using DataFrames
using Plots
using JLD2
using BenchmarkTools
```

Установка каталогов

```
script_name = splitext(basename(PROGRAM_FILE))[1]
mkpath(plotsdir(script_name))
mkpath(datadir(script_name))
```



## 5.2 Определение модели

Катер:  $dr/d\theta = r / \sqrt{n^2 - 1}$

```
function cutter_ode!(dr, r, p, θ)
    dr[1] = r[1] / sqrt(p.n^2 - 1)
end
```

Лодка:  $x=t, y=\tan(\theta+\pi)*t \rightarrow (r, \theta)$  Важно: используем  $\text{atan}(y, x)$ , чтобы угол был корректен по квадрантам

```
function boat_polar(t, p)
    k = tan(p.fi + pi)
    x = t
    y = k * t
    r = hypot(x, y)
    θ = atan(y, x)
    return r, θ
end
```

## 5.3 Определение параметров в Dict

Все параметры — в одном Dict, как в шаблоне. Здесь «case» управляет выбором  $r_0$ :  $:plus(s/(n+1))$  или  $:minus(s/(n-1))$ .

```
base_params = Dict(
    :n => 5,
    :s => 20,
    :fi => 3/4*pi,
```

```

:case => :plus,                # :plus или :minus
:ϕspan => (0.0, 2*pi),         # интервал по ϕ
:nϕ => 10_000,                # число точек для сохранения решения катера

:tspan_boat => (1e-9, 8.0),    # интервал по t для лодки
:nt_boat => 1_000,             # число точек для лодки

:solver => Tsit5(),
:experiment_name => "base_experiment"
)

println("Базовые параметры эксперимента:")
for (key, value) in base_params
    println(" $key = $value")
end

```

## 5.4 Функция-обертка для запуска одного эксперимента

Возвращаем Dict со строковыми ключами (как в твоём шаблоне).

```

function run_single_experiment(params::Dict)
    @unpack n, s, fi, case, ϕspan, nϕ, tspan_boat, nt_boat, solver = params

```

Параметры для ODE в виде именованного кортежа

```
p = (n=n, s=s, fi=fi)
```

Начальное условие  $r_0$  зависит от кейса

```
r0 = case == :plus ? s/(n+1) : s/(n-1)
```

— Катер (ODE) —

```
ϕgrid = collect(LinRange(ϕspan[1], ϕspan[2], nϕ))  
prob = ODEProblem(cutter_ode!, [r0], ϕspan, p)  
sol = solve(prob, solver; saveat=ϕgrid)  
  
r_cutter = first.(sol.u)
```

— Лодка (аналитика) —

```
tgrid = collect(LinRange(tspan_boat[1], tspan_boat[2], nt_boat))  
r_boat = VectorFloat64(undef, length(tgrid))  
ϕ_boat = VectorFloat64(undef, length(tgrid))  
for (i, tt) in pairs(tgrid)  
    ri, ϕi = boat_polar(tt, p)  
    r_boat[i] = ri  
    ϕ_boat[i] = ϕi  
end
```

— Мини-анализ (несколько метрик, чтобы было что сравнивать в сканировании) — 1) финальный радиус катера на  $\theta=\text{end}$

```
r_cutter_final = r_cutter[end]
```

2) максимальный радиус лодки на её сетке

```
r_boat_max = maximum(r_boat)
```

3) простая «разница масштабов» (без строгого физического смысла, но для сравнения кейсов полезно)

```

scale_ratio = r_cutter_final / r_boat_max

return Dict(
    "solution" => sol,
    "□_points" => sol.t,
    "r_cutter" => r_cutter,

    "t_points_boat" => tgrid,
    "□_boat" => □_boat,
    "r_boat" => r_boat,

    "r0" => r0,
    "r_cutter_final" => r_cutter_final,
    "r_boat_max" => r_boat_max,
    "scale_ratio" => scale_ratio,

    "parameters" => params
)
end

```

## 5.5 Запуск базового эксперимента (кэширование)

```

data_base, path_base = produce_or_load(
    datadir(script_name, "single"),
    base_params,
    run_single_experiment;
    prefix = "polar",

```

```

    tag = false,
    verbose = true
)

println("\nРезультаты базового эксперимента:")
println(" r0: ", data_base["r0"])
println(" r_cutter_final: ", data_base["r_cutter_final"])
println(" r_boat_max: ", data_base["r_boat_max"])
println(" scale_ratio: ", round(data_base["scale_ratio"]; digits=4))
println(" Файл результатов: ", path_base)

```

## 5.6 Визуализация базового эксперимента

```

p1 = plot(
    data_base["□_points"], data_base["r_cutter"],
    proj=:polar,
    label="катер",
    xlabel="□",
    ylabel="r",
    title="Базовый эксперимент (case=$(base_params[:case]))",
    lw=2,
    legend=:topleft,
    grid=true
)
plot!(
    p1,
    data_base["□_boat"], data_base["r_boat"],

```

```

    proj=:polar,
    label="лодка",
    lw=2
)

savefig(p1, plotsdir(script_name, "single_experiment.png"))

```

## 5.7 Второй базовый эксперимент (как у тебя во 2-й части): case=:minus, tmax=15

```

base_params2 = copy(base_params)
base_params2[:case] = :minus
base_params2[:tspan_boat] = (1e-9, 15.0)
base_params2[:experiment_name] = "base_experiment_minus"

data_base2, path_base2 = produce_or_load(
    datadir(script_name, "single"),
    base_params2,
    run_single_experiment;
    prefix = "polar",
    tag = false,
    verbose = true
)

p1b = plot(
    data_base2["□_points"], data_base2["r_cutter"],
    proj=:polar,

```

```

    label="катер",
    xlabel="",
    ylabel="r",
    title="Базовый эксперимент (case=$(base_params2[:case]))",
    lw=2,
    legend=:topleft,
    grid=true
)
plot!(
    p1b,
    data_base2["_boat"], data_base2["r_boat"],
    proj=:polar,
    label="лодка",
    lw=2
)
savefig(p1b, plotsdir(script_name, "single_experiment_minus.png"))

```

## 5.8 Параметрическое сканирование

В твоей задаче естественно сканировать  $n$  (оно влияет и на ODE, и на  $r_0$ ). При желании можно заменить на  $:fi$  или  $:s$  (или сделать сетку по двум параметрам).

```

param_grid = Dict(
    :n => [3, 4, 5, 6, 8, 10],      # сканируем n
    :s => [20],                    # фиксируем
    :fi => [3/4*pi],               # фиксируем

    :case => [:plus, :minus],      # сканируем оба кейса

```

```

:span => [(0.0, 2*pi)],
:n => [10_000],

:tspan_boat => [(1e-9, 8.0)], # можно тоже сканировать, но обычно фиксируют
:nt_boat => [1_000],

:solver => [Tsit5()],
:experiment_name => ["parametric_scan"]
)

all_params = dict_list(param_grid)

println("\n" * "="^60)
println("ПАРАМЕТРИЧЕСКОЕ СКАНИРОВАНИЕ")
println("Всего комбинаций параметров: ", length(all_params))
println("Исследуемые n: ", param_grid[:n])
println("Исследуемые case: ", param_grid[:case])
println("="^60)

```

## 5.9 Запуск всех экспериментов и сбор результатов

```

all_results = []
all_dfs = []

for (i, params) in enumerate(all_params)
    println("Порядок: $i/$(length(all_params)) | n=$(params[:n]) | case=$(params[:case])")
    # ... (code for running experiments) ...
end

```



```

data, path = produce_or_load(
    datadir(script_name, "parametric_scan"),
    params,
    run_single_experiment;
    prefix = "scan",
    tag = false,
    verbose = false
)

```

Сводка по эксперименту

```

result_summary = merge(
    params,
    Dict(
        :r0 => data["r0"],
        :r_cutter_final => data["r_cutter_final"],
        :r_boat_max => data["r_boat_max"],
        :scale_ratio => data["scale_ratio"],
        :filepath => path
    )
)
push!(all_results, result_summary)

```

Полные данные (катер) — удобно для дальнейших графиков/анализа

```

df = DataFrame(
    □ = data["□_points"],
    r = data["r_cutter"],
    n = fill(params[:n], length(data["□_points"])),
    case = fill(string(params[:case]), length(data["□_points"]))
)

```

```

    )
    push!(all_dfs, df)
end

```

## 5.10 Анализ и визуализация результатов сканирования

```

results_df = DataFrame(all_results)
println("\nСводная таблица результатов (первые строки):")
println(first(results_df, 10))

```

Сравнительный график траекторий катера для всех комбинаций (по  $\theta$ )

```

p2 = plot(size=(900, 520), dpi=150)
for params in all_params
    data, _ = produce_or_load(
        datadir(script_name, "parametric_scan"),
        params,
        run_single_experiment;
        prefix = "scan",
        tag = false,
        verbose = false
    )

    plot!(
        p2,
        data["□_points"], data["r_cutter"],
        label="n=$(params[:n]), case=$(params[:case])",

```

```

        lw=2,
        alpha=0.8
    )
end
plot!(
    p2,
    xlabel="□",
    ylabel="r(□)",
    title="Сканирование: траектории катера (ODE) при разных n и case",
    legend=:outerright,
    grid=true
)
savefig(p2, plotsdir(script_name, "parametric_scan_cutter_comparison.png"))

```

График метрики `scale_ratio` по `n` (раздельно для `case`) (`scale_ratio = r_cutter_final / r_boat_max`)

```

p3 = plot(size=(900, 520), dpi=150)
for cs in unique(results_df.case)
    sub = results_df[results_df.case .== cs, :]
    plot!(
        p3,
        sub.n, sub.scale_ratio,
        seriestype=:scatter,
        label="case=$cs"
    )
end
plot!(
    p3,

```

```

xlabel="n",
ylabel="scale_ratio",
title="Зависимость scale_ratio от n (для разных case)",
legend=:topleft,
grid=true
)
savefig(p3, plotsdir(script_name, "scale_ratio_vs_n.png"))

```

## 5.11 Бенчмаркинг с разными параметрами

```

println("\n" * "="^60)
println("Бенчмаркинг для разных n (оба case)")
println("="^60)

benchmark_results = []

```

Возьмём сетку n из param\_grid и оба case (как в сканировании)

```

for n_value in param_grid[:n], case_value in param_grid[:case]
    bench_params = Dict(
        :n => n_value,
        :s => base_params[:s],
        :fi => base_params[:fi],
        :case => case_value,
        :span => base_params[:span],
        :n[] => base_params[:n[]],
        :tspan_boat => base_params[:tspan_boat],
        :nt_boat => base_params[:nt_boat],
    )

```

```

        :solver => base_params[:solver]
    )

```

```

function benchmark_run()

```

Только катер (ODE) — это и есть «вычислительная часть»

```

    p = (n=bench_params[:n], s=bench_params[:s], fi=bench_params[:fi])
    r0 = bench_params[:case] == :plus ? bench_params[:s]/(bench_params[:n]+1) : b
    prob = ODEProblem(cutter_ode!, [r0], bench_params[:tspan], p)
    return solve(prob, bench_params[:solver]; saveat=LinRange(bench_params[:tspan],
end

```

```

println("\nБенчмарк для n = $n_value, case = $case_value:")
b = @benchmark $benchmark_run() samples=80 evals=1
tsec = median(b).time / 1e9
println(" Медианное время: ", round(tsec; digits=4), " сек")

```

```

    push!(benchmark_results, (n=n_value, case=string(case_value), time=tsec))
end

```

```

bench_df = DataFrame(benchmark_results)

```

График времени вычисления от n (раздельно по case)

```

p4 = plot(size=(900, 520), dpi=150)
for cs in unique(bench_df.case)
    sub = bench_df[bench_df.case .== cs, :]
    plot!(
        p4,

```

```

        sub.n, sub.time,
        seriestype=:scatter,
        label="case=$cs"
    )
end
plot!(
    p4,
    xlabel="n",
    ylabel="Время вычисления, сек",
    title="Зависимость времени решения ODE от n (для разных case)",
    legend=:topleft,
    grid=true
)
savefig(p4, plotsdir(script_name, "computation_time_vs_n.png"))

```

## 5.12 Сохранение всех результатов

```

@save datadir(script_name, "all_results.jld2") base_params base_params2 param_grid al
@save datadir(script_name, "all_plots.jld2") p1 p1b p2 p3 p4

println("\n" * "="^60)
println("ЛАБОРАТОРНАЯ РАБОТА ЗАВЕРШЕНА")
println("="^60)
println("\nРезультаты сохранены в:")
println(" • data/$(script_name)/single/ - базовые эксперименты")
println(" • data/$(script_name)/parametric_scan/ - параметрическое сканирование")
println(" • data/$(script_name)/all_results.jld2 - сводные данные")

```

```
println(" • plots/${script_name}/ - все графики")
println(" • data/${script_name}/all_plots.jld2 - объекты графиков")
println("\nДля анализа результатов используйте:")
println(" using JLD2, DataFrames")
println(" @load \"data/${script_name}/all_results.jld2\"")
println(" println(results_df)")
```

## 5.13 Анализ результатов моделирования

В ходе работы выполнено численное исследование движения катера, описываемого уравнением:

$$\frac{dr}{d\theta} = \frac{r}{\sqrt{n^2 - 1}},$$

а также сопоставление полученной траектории со траекторией лодки, заданной аналитически и представленной в полярных координатах.

## 5.14 Базовые эксперименты

### 5.14.1 1. Случай (case = plus)

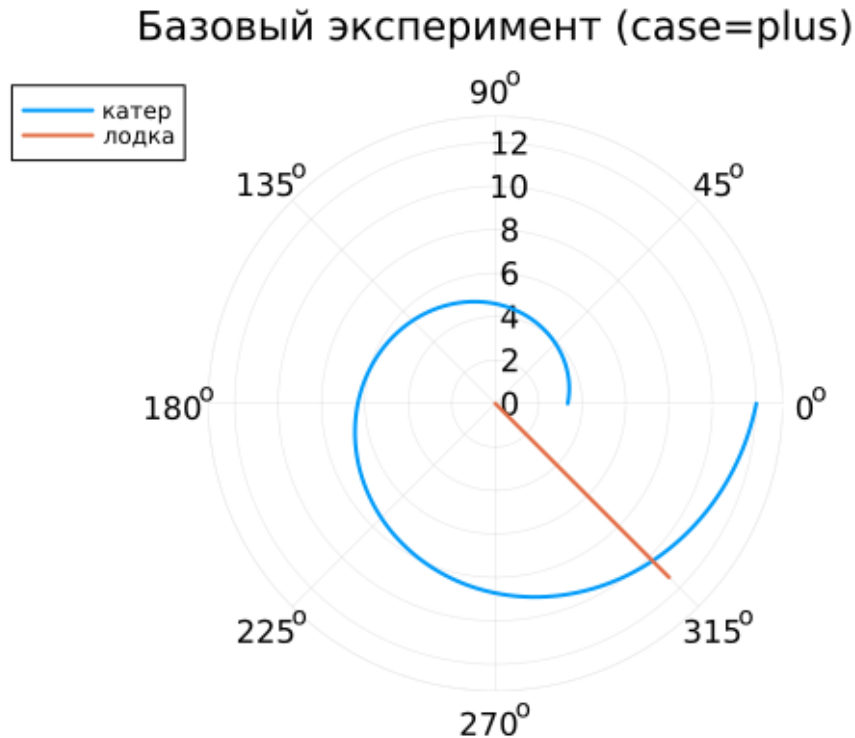


Рисунок 5.1: Базовый эксперимент (case=plus)

По полярному графику видно, что траектория катера представляет собой расходящуюся спираль: при росте угла  $\theta$  радиус  $r$  увеличивается монотонно. Причина в том, что  $dr/d\theta$  пропорциональна  $r$ , из-за чего рост по  $\theta$  приобретает экспоненциальный характер.

Траектория лодки в полярной системе выражается лучом, поскольку в декартовой системе она движется по прямой.

По рисунку заметно, что катер увеличивает радиус быстрее, чем лодка, и постепенно «обходит» её по траектории, создавая условия для перехвата.



### 5.14.2 2. Случай (case = minus)



Рисунок 5.2: Базовый эксперимент (case=minus)

Во втором сценарии стартовый радиус больше, поэтому катер начинает движение дальше от полюса. Форма траектории остаётся той же (спираль), однако весь график оказывается сдвинут наружу, то есть отличается масштабом.

Таким образом, отличие case=plus и case=minus проявляется главным образом в начальных условиях, а не в типе траектории.

## 5.15 Параметрическое сканирование по $n$

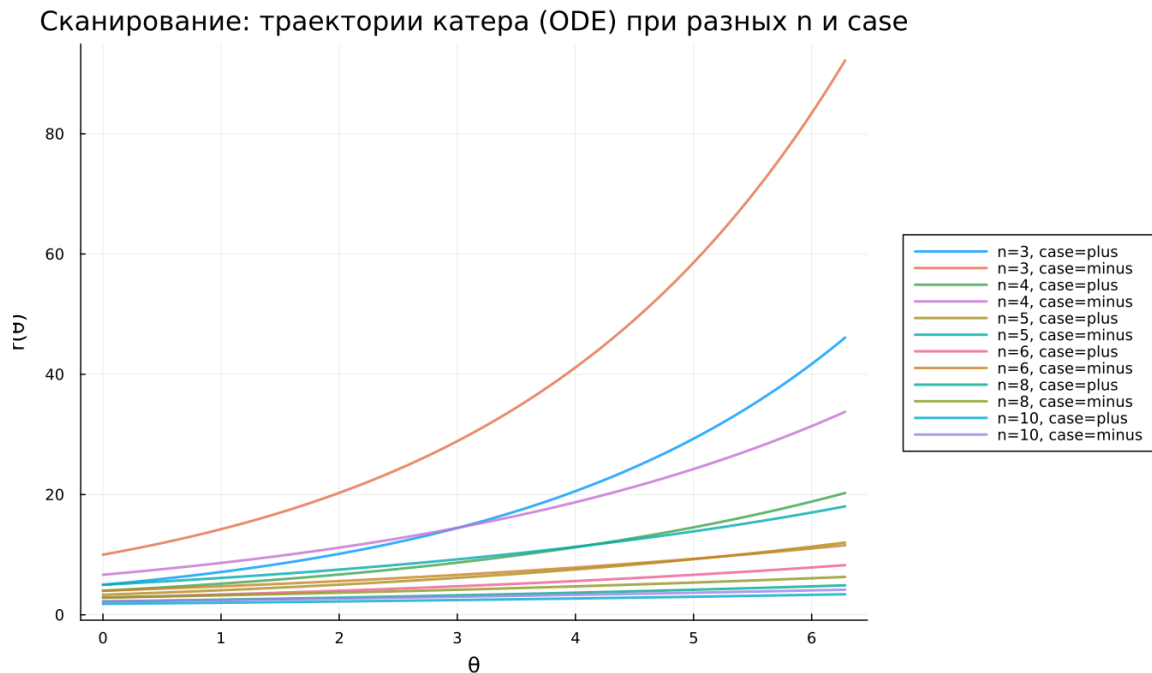


Рисунок 5.3: Сканирование траекторий катера

Проверено влияние параметра  $n$  на динамику системы для обоих режимов.

Из уравнения следует, что «крутизна» спирали определяется коэффициентом

$$\frac{1}{\sqrt{n^2 - 1}}.$$

Следовательно: - при меньших  $n$  спираль расходится быстрее; - при больших  $n$  увеличение радиуса происходит медленнее; - траектории становятся более «пологими» и менее быстро расходящимися.

По графику видно: при  $n = 3$  рост наиболее интенсивный, а при  $n = 10$  — наиболее медленный.

## 5.16 Анализ метрики scale\_ratio

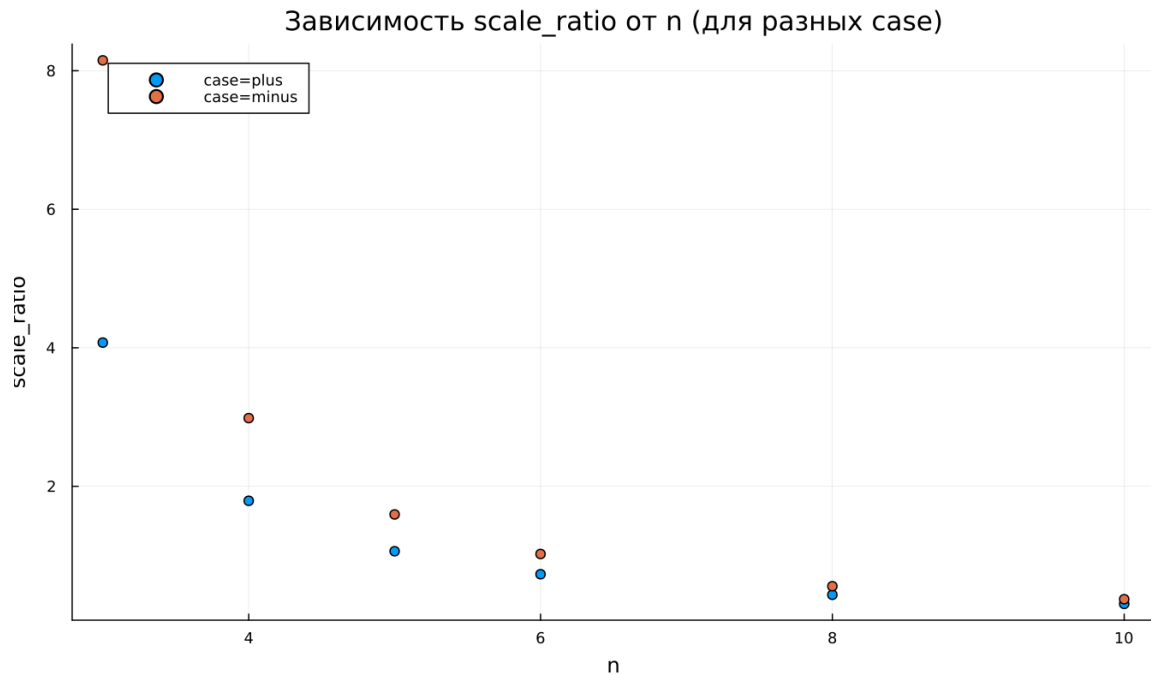


Рисунок 5.4: Зависимость scale\_ratio от n

Для сравнения масштабов введена метрика:

$$\text{scale\_ratio} = \frac{r_{\text{final}}}{\max(r_{\text{boat}})}.$$

Наблюдения по графику: - при малых  $n$  значение существенно превышает 1, то есть катер уходит по радиусу заметно дальше лодки; - с ростом  $n$  метрика быстро уменьшается; - при больших  $n$  масштабы траекторий становятся ближе друг к другу.

В режиме case=minus метрика закономерно больше, так как стартовый радиус выше.

## 5.17 Время вычислений

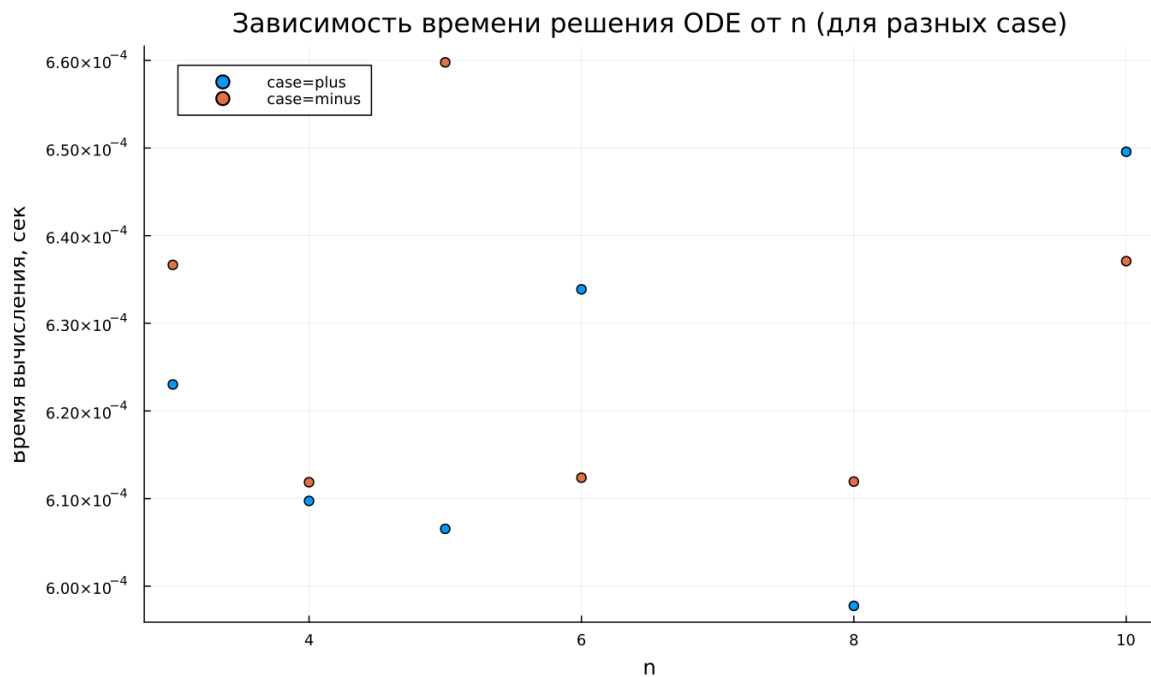


Рисунок 5.5: Зависимость времени вычисления от  $n$

Проведён бенчмаркинг численного решения ОДУ для различных значений  $n$ .

Из результата следует: - время вычислений находится на уровне  $\sim 6 \times 10^{-4}$  секунды; - заметной зависимости от  $n$  не выявлено; - небольшие флуктуации связаны с особенностями адаптивного шага и внутренней численной обработкой.

## 6 Выводы

1. Траектория катера в полярной системе координат описывается расходящейся (экспоненциальной по углу) спиралью.
2. Параметр  $n$  влияет на скорость радиального роста: чем больше  $n$ , тем медленнее «раскручивается» спираль.
3. Выбор начального условия (case) изменяет масштаб траектории, но не изменяет её качественную форму.
4. Численное решение устойчиво, а вычислительные затраты практически не зависят от параметров модели.

Итоги моделирования согласуются с аналитическим видом полученного дифференциального уравнения.

# Список литературы

1. Задача о погоне