Математическое моделирование

Лабораторная работа № 2

Максим Новичков

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Рассматривается пример построения математической модели для выбора рациональной стратегии в задаче преследования.

Пусть в условиях ограниченной видимости катер береговой охраны преследует лодку нарушителей. В момент, когда туман рассеивается, лодка фиксируется на расстоянии k км от катера. Далее лодка продолжает движение по прямой в неизвестном направлении и снова исчезает из поля зрения.

Известно, что скорость катера превышает скорость лодки в n раз. Требуется определить такую траекторию движения катера, которая гарантирует перехват цели.

2 Задание

1. Выполнить аналитические рассуждения и получить систему дифференциальных уравнений при условии, что скорость катера в n раз больше скорости лодки.

2. Построить траектории движения катера и лодки для двух различных начальных ситуаций.

3. По результатам моделирования определить точку встречи.

3 Выполнение лабораторной работы

Положим $t_0 = 0$. За начало координат примем положение лодки в момент обнаружения: $X_0 = 0$. Катер в этот момент находится на расстоянии k, то есть $X_0 = k$ относительно лодки.

Перейдём к полярной системе координат. Полюс совместим с точкой обнаружения лодки ($r = 0$, $\theta = 0$), а полярную ось направим через положение катера.

Определим расстояние x, при котором катер и лодка окажутся на одинаковом удалении от полюса. Пусть через время t лодка удалится на расстояние x, а катер — на $x + k$ либо $x - k$ в зависимости от конфигурации.

Так как время движения одинаково, получаем равенства:

в первом случае

$\frac{x}{v} = \frac{x+k}{nv}$

во втором случае

$\frac{x}{v} = \frac{x-k}{nv}$

Решая эти уравнения, получаем два возможных начальных радиуса:

$x_1 = \frac{k}{n+1}$ при $\theta_0 = 0$

$x_2 = \frac{k}{n-1}$ при $\theta_0 = -\pi$

Далее катер должен изменить характер движения: вместо прямолинейного начать обход вокруг полюса так, чтобы его радиальная скорость совпадала со скоростью лодки.

Разложим скорость катера на компоненты:

- радиальная скорость $v_r = \frac{dr}{dt}$

- тангенциальная скорость $v_\tau = r\frac{d\theta}{dt}$

Требуем $v_r = v$.

Полная скорость катера равна nv, следовательно из теоремы Пифагора:
$v_\tau = \sqrt{(nv)^2 - v^2} = v\sqrt{n^2 - 1}$

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

Итоговая система дифференциальных уравнений принимает вид:

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

Начальные условия:

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

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

Исключая время t, получаем уравнение траектории:
$\frac{dr}{d\theta} = \frac{r}{\sqrt{n^2-1}}$

Решение этого уравнения определяет траекторию катера в полярных координатах.

3.1 Условие задачи

Лодка обнаружена на расстоянии 20 км от катера.
Скорость катера превышает скорость лодки в 5 раз, то есть $n = 5$.

Моделирование и построение графиков выполнялись с использованием программных модулей:

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/dθ = r / sqrt(n^2 - 1) Здесь независимая переменная — θ (мы используем t как θ, 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, θ) В Julia правильнее использовать 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))
    □ = VectorFloat64(undef, length(tgrid))
    for (i, tt) in pairs(tgrid)
        ri, □i = boat_polar(tt, p)
        r[i] = ri
        □[i] = □i
    end
    return r, □
end
```

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

```
function run_case(case_name; r0, □span=(0.0, 2pi), n□=10_000, tmin=1e-9, tmax=8.0, nt
```

— Катер (ODE) —

```
    □grid = collect(LinRange(□span[1], □span[2], n□))
    prob = ODEProblem(cutter_ode!, [r0], □span, p)
    sol  = solve(prob, Tsit5(), saveat=□grid)

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

— Лодка (аналитически) —

```
    tgrid = collect(LinRange(tmin, tmax, nt))
    r_boat, □_boat = make_boat_curve(tgrid, p)
    df_boat = DataFrame(t = tgrid, □ = □_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, _boat, 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 span r

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

4.4 Запуск 1: r0 = s/(n+1), t (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: r0 = s/(n-1), t (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/dθ = r / sqrt(n^2 - 1)

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

Лодка: x=t, y=tan(fi+pi)*t -> (r, θ) Важно: используем 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)
    □ = atan(y, x)
    return r, □
end
```

5.3 Определение параметров в Dict

Все параметры — в одном Dict, как в шаблоне. Здесь «case» управляет выбором r0: :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
    :□span => (0.0, 2*pi),           # интервал по □
    :n□ => 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 твоем шаблоне).

```
function run_single_experiment(params::Dict)
    @unpack n, s, fi, case, □span, n□, tspan_boat, nt_boat, solver = params
```

Параметры для ODE в виде именованного кортежа

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

Начальное условие r0 зависит от кейса

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

— Катер (ODE) —

```
□grid = collect(LinRange(□span[1], □span[2], n□))
prob = ODEProblem(cutter_ode!, [r0], □span, p)
sol = solve(prob, solver; saveat=□grid)

r_cutter = first.(sol.u)
```

— Лодка (аналитика) —

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

— Мини-анализ (несколько метрик, чтобы было что сравнивать в сканировании) — 1) финальный радиус катера на θ=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 r0). 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("Прогресс: $i/$(length(all_params)) | n=$(params[:n]) | case=$(params[:ca
```

```
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 θ)

```
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[:□span], p)
        return solve(prob, bench_params[:solver]; saveat=LinRange(bench_params[:□span
    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

Численное исследование проводилось для уравнения

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

и сопоставлялось с движением лодки, описанным аналитически.

5.14 Базовые эксперименты

5.14.1 1. Случай (case = plus)

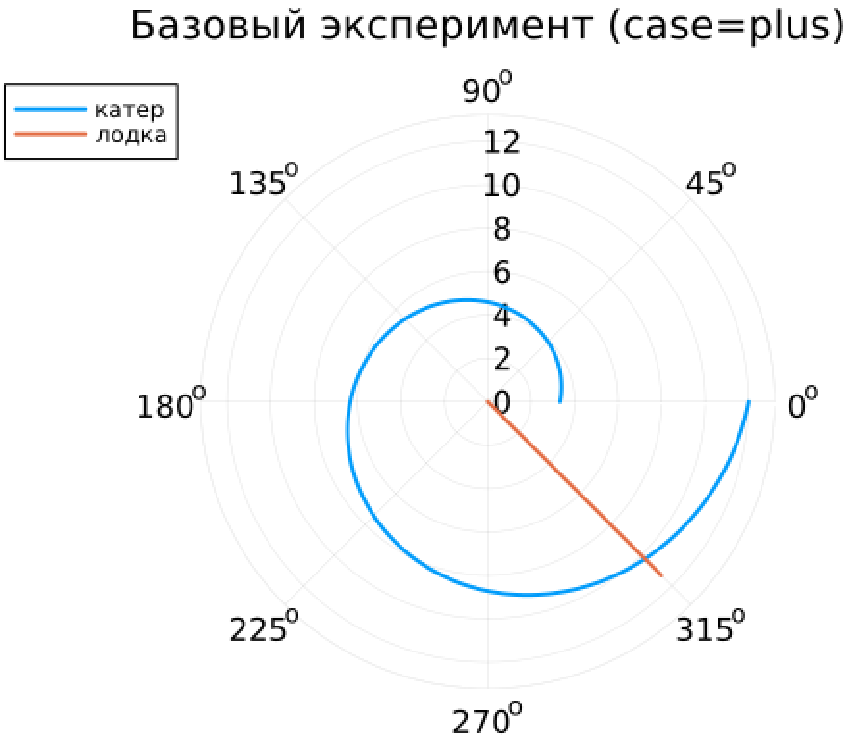

Рисунок 5.1: Базовый эксперимент (case=plus)

Траектория катера представляет собой логарифмическую спираль. Радиус возрастает экспоненциально по углу θ, поскольку производная пропорциональна самому r.

Движение лодки соответствует лучу, исходящему из полюса.

Катер постепенно выходит на траекторию перехвата за счёт более высокой скорости.

5.14.2 2. Случай (case = minus)

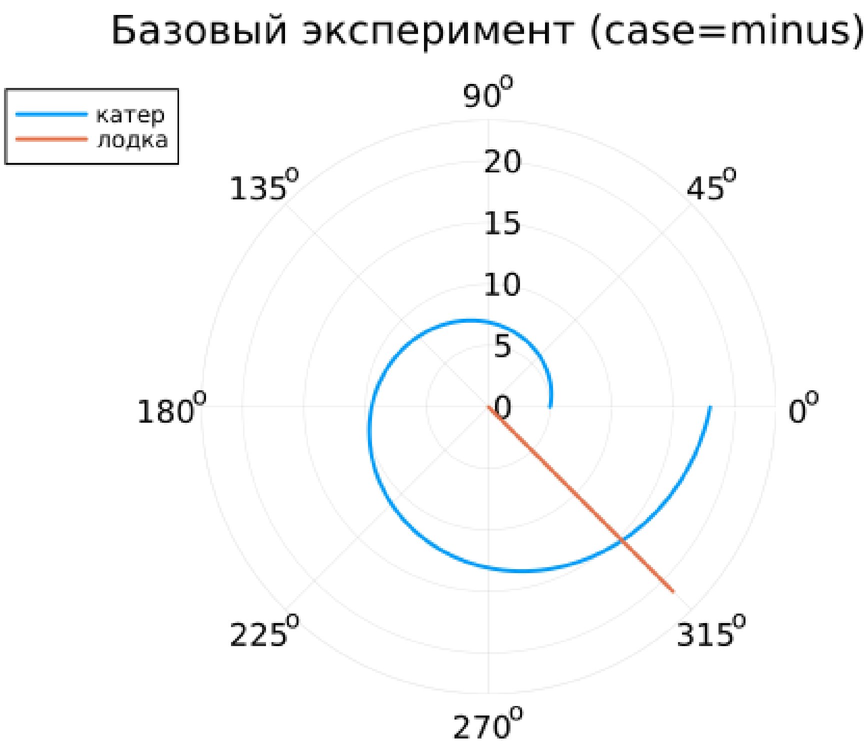

Рисунок 5.2: Базовый эксперимент (case=minus)

Во втором варианте катер стартует с большего радиуса. Форма спирали сохраняется, однако вся траектория сдвинута наружу.

Различие режимов определяется начальными условиями, а не структурой решения.

5.15 Параметрическое исследование по n

Рисунок 5.3: Сканирование траекторий катера

Коэффициент при r равен:

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

Следовательно:

- при меньших n спираль расширяется быстрее;
- при увеличении n рост радиуса замедляется;
- траектория становится более пологой.

5.16 Анализ метрики scale_ratio

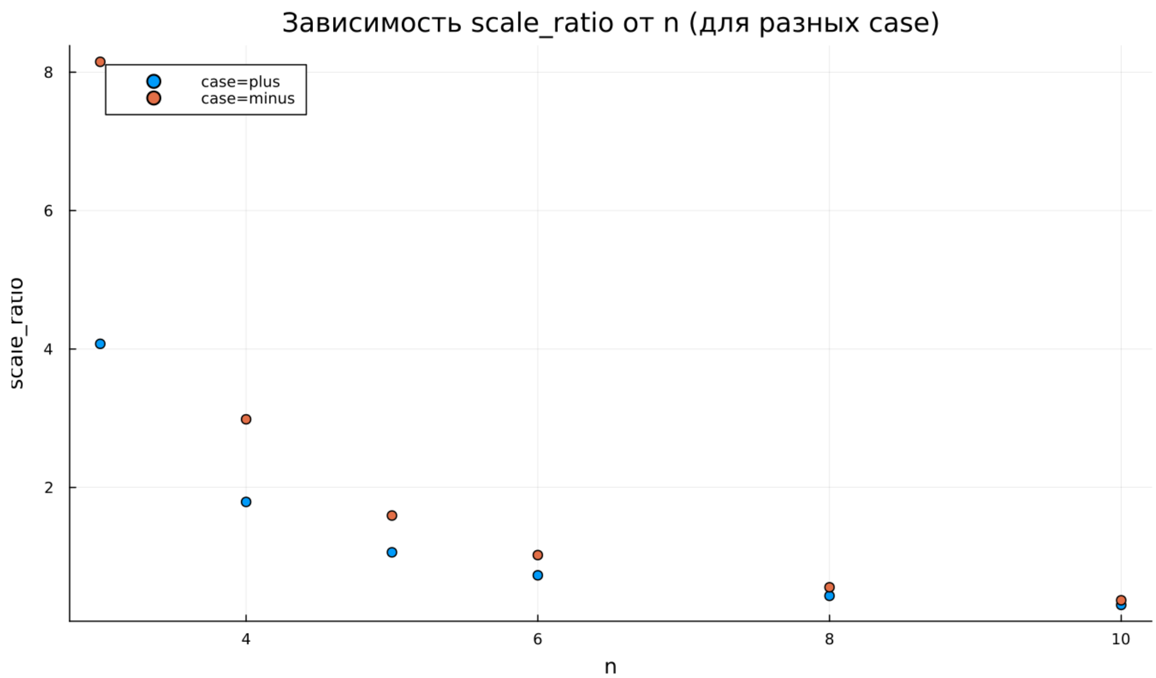

Рисунок 5.4: Зависимость scale_ratio от n

Введена величина

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

Она отражает относительный масштаб траектории катера.

Наблюдается:

- значительное превышение единицы при малых n;
- уменьшение значения при росте n;
- близость масштабов при больших n.

5.17 Время вычислений

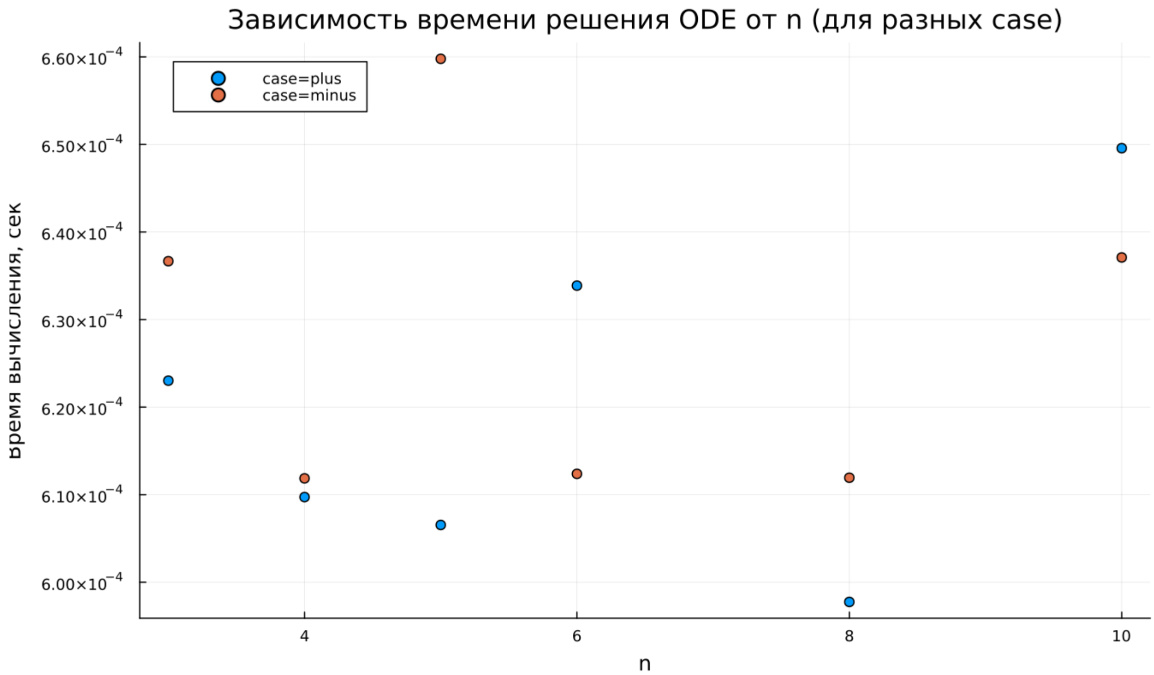

Рисунок 5.5: Зависимость времени вычисления от n

Численное решение демонстрирует:

- среднее время порядка 6×10^{-4} секунды;
- отсутствие выраженной зависимости от n;
- малые флуктуации, обусловленные особенностями алгоритма интегрирования.

6 Выводы

1. Траектория катера описывается логарифмической спиралью.
2. Параметр n регулирует интенсивность радиального роста.
3. Начальные условия влияют на масштаб, но не на форму решения.
4. Численный метод демонстрирует устойчивость и малые вычислительные затраты.

Полученные результаты согласуются с аналитическим решением модели.

Список литературы

1. Задача о погоне