

Математическое моделирование

Лабораторная работа № 2

Абдуллахи Шугофа

Содержание

1 Цель работы	6
2 Задание	7
3 Выполнение лабораторной работы	8
3.1 Условие задачи	10
4 Полярные траектории: катер (ODE) и лодка (аналитически)	11
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
5 Параметрическое исследование: катер (ODE) и лодка (аналитически) в полярных координатах	16
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 исследование по n	34
5.16 Анализ относительного масштаба	35
5.17 Время вычислений	35
6 Выводы	37

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 времени вычисления от n	35

Список таблиц

1 Цель работы

Рассматривается пример построения математической модели для выбора рациональной стратегии в задачах преследования. В качестве иллюстрации анализируется ситуация, при которой катер береговой охраны пытается догнать лодку браконьеров в условиях ограниченной видимости.

После рассеивания тумана лодка фиксируется на расстоянии k км от катера. Далее она продолжает движение равномерно и прямолинейно в неизвестном направлении. Известно, что скорость катера превышает скорость лодки в n раз. Требуется определить такую траекторию движения катера, которая гарантирует перехват цели.

2 Задание

1. Выполнить вывод дифференциальных уравнений, описывающих движение катера при условии, что его скорость в n раз больше скорости лодки.
2. Построить траектории движения объектов для двух вариантов начального расположения.
3. По графическим результатам определить точку встречи катера и лодки.

3 Выполнение лабораторной работы

Положим $t_0 = 0$. Начало координат совместим с точкой обнаружения лодки. В этот момент расстояние между объектами равно k .

Перейдём к полярной системе координат (r, θ) , где полюс совпадает с точкой обнаружения лодки, а полярная ось направлена в сторону катера.

Пусть через некоторое время оба объекта окажутся на одинаковом расстоянии x от полюса. За это время лодка проходит путь x , а катер — либо $x + k$, либо $x - k$ в зависимости от начального положения.

Так как время движения одинаково, получаем равенство:

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

или

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

Отсюда определяются два возможных значения начального радиуса:

$$x_1 = \frac{k}{n+1}, \quad \theta = 0,$$

$$x_2 = \frac{k}{n-1}, \quad \theta = -\pi.$$

После достижения равенства радиусов катер должен двигаться так, чтобы его радиальная скорость совпадала со скоростью лодки.

Разложим скорость катера на составляющие:

- радиальная скорость:

$$v_r = \frac{dr}{dt},$$

- тангенциальная скорость:

$$v_\tau = r \frac{d\theta}{dt}.$$

Поскольку модуль полной скорости катера равен nV , имеем:

$$(nV)^2 = v_r^2 + v_\tau^2.$$

При условии $v_r = V$ получаем:

$$v_\tau = V\sqrt{n^2 - 1}.$$

Следовательно,

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

Таким образом, система уравнений принимает вид:

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

Начальные условия:

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

Исключая параметр t , получаем дифференциальное уравнение траектории:

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

Решение этого уравнения описывает движение катера в полярных координатах.

3.1 Условие задачи

Лодка обнаружена на расстоянии 20 км от катера.

Скорость катера превышает скорость лодки в 5 раз ($n = 5$).

Для численного моделирования и построения графиков использовались внешние программные модули:

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 — θ (мы используем t как θ , как принято в ODEProblem).

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

Лодка: линия $x(t)=t$, $y(t)=\tan(\text{fi}+\pi)*t$, затем перевод в (r, θ) 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)
    θ = atan(y, x)
    return r, θ
end
```

Утилита: генерация траектории лодки для набора t

```

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

```

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

```

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

```

— Катер (ODE) —

```

θgrid = collect(LinRange(θspan[1], θspan[2], nθ))
prob = ODEProblem(cutter_ode!, [r₀], θspan, p)
sol = solve(prob, Tsit5(), saveat=θgrid)

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

```

— Лодка (аналитически) —

```

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

```

— Визуализация —

```

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

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: $x=t, y=\tan(\phi+\pi)*t \rightarrow (r, \theta)$ 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 $r0: :plus (s/(n+1))$ 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 θ
: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

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

— Катер (ODE) —

```
grid = collect(LinRange(span[1], span[2], n))
prob = ODEProblem(cutter_ode!, [rθ], 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))
θ_boat = VectorFloat64(undef, length(tgrid))
for (i, tt) in pairs(tgrid)
    ri, θi = boat_polar(tt, p)
    r_boat[i] = ri
    θ_boat[i] = θi
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,
    "t_points" => sol.t,
    "r_cutter" => r_cutter,
    "t_points_boat" => tgrid,
    "t_boat" => t_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["l_points"], data_base["r_cutter"],
    proj=:polar,
    label="катер",
    xlabel="l",
    ylabel="r",
    title="Базовый эксперимент (case=$(base_params[:case]))",
    lw=2,
    legend=:topleft,
    grid=true
)
plot!(
    p1,
    data_base["l_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["l_points"], data_base2["r_cutter"],
    proj=:polar,

```

```

label="катер",
xlabel="t",
ylabel="r",
title="Базовый эксперимент (case=$(base_params2[:case]))",
lw=2,
legend=:topleft,
grid=true
)
plot!(
p1b,
data_base2["t_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["l_points"], data["r_cutter"],  
        label="n=$(params[:n]), case=$(params[:case])",
```

```

        lw=2,
        alpha=0.8
    )
end
plot!(
    p2,
    xlabel="n",
    ylabel="r(n)",
    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) ($\text{scale_ratio} = \text{r_cutter_final} / \text{r_boat_max}$)

```

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,
        :lspan => base_params[:lspan],
        :n_l => base_params[:n_l],
        :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 all

@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 решение уравнения

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

позволило сопоставить траекторию катера с движением лодки, выраженным в полярной системе координат.

5.14 Базовые эксперименты

5.14.1 1. Режим 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 .

Лодка движется прямолинейно, что в полярных координатах соответствует лучу.

Катер постепенно увеличивает радиус быстрее, обеспечивая возможность перехвата.

5.14.2 2. Режим case = minus

Базовый эксперимент (case=minus)

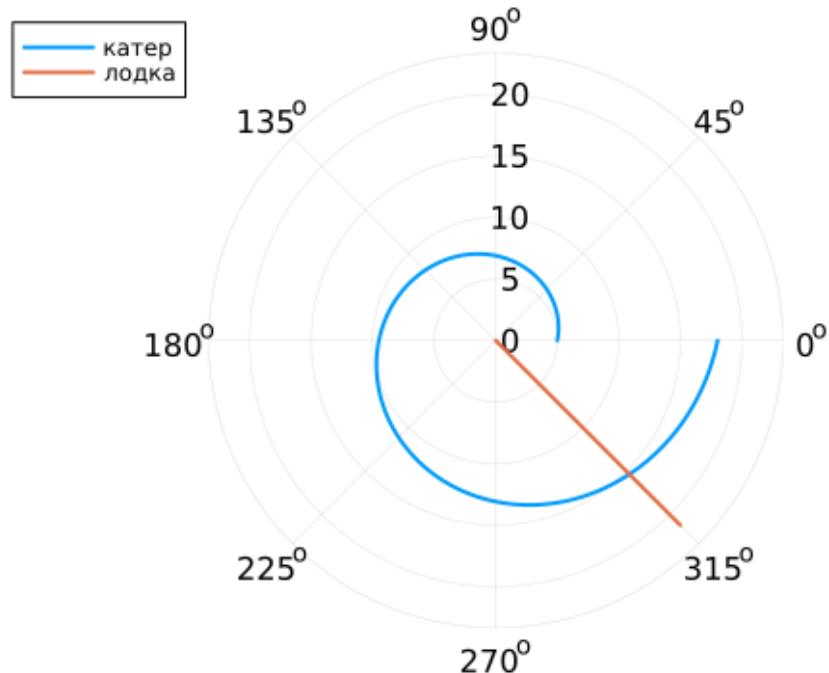


Рисунок 5.2: Базовый эксперимент (case=minus)

При втором наборе начальных условий стартовый радиус больше. Форма спирали сохраняется, но вся траектория смещается наружу.

Отличие режимов проявляется в масштабе, а не в характере роста функции.

5.15 Параметрическое исследование по n

Сканирование: траектории катера (ODE) при разных n и case

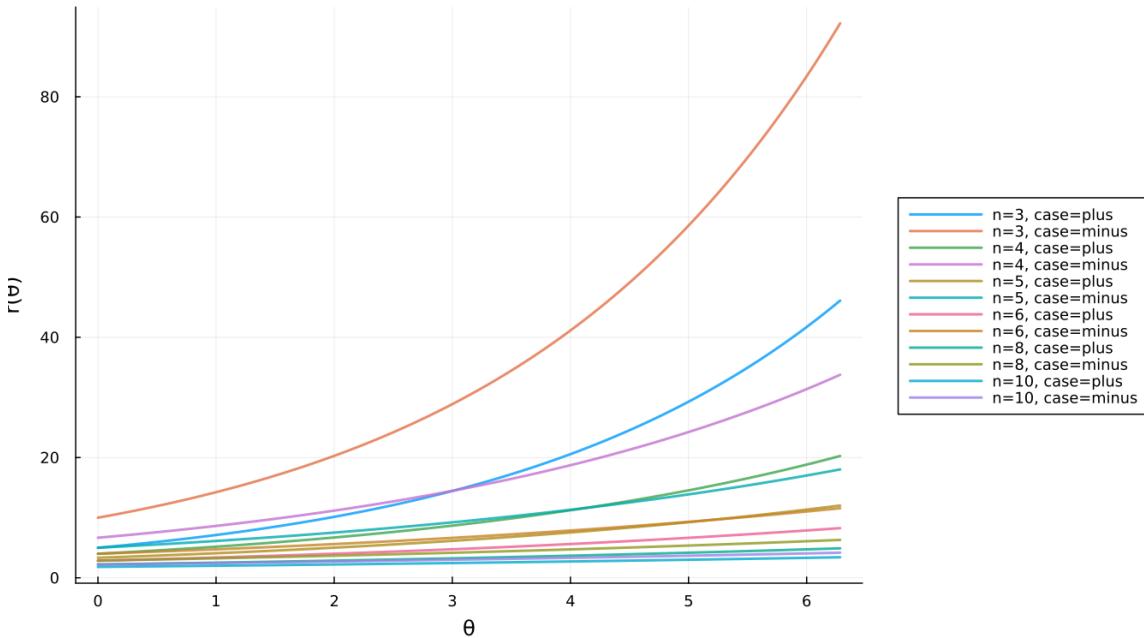


Рисунок 5.3: Сканирование траекторий катера

Из выражения

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

следует, что коэффициент роста равен

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

При увеличении n этот коэффициент уменьшается, что приводит к более плавному росту спирали.

- при малых n радиус увеличивается быстрее;
- при больших n траектория становится более пологой;
- характер движения качественно сохраняется.

5.16 Анализ относительного масштаба

Введена характеристика

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

Она показывает соотношение радиусов катера и лодки к моменту окончания моделирования.

При небольших значениях n катер значительно превосходит лодку по радиальному удалению.

С ростом n различие уменьшается.

Во втором режиме (case=minus) показатель выше из-за большего начального радиуса.

5.17 Время вычислений

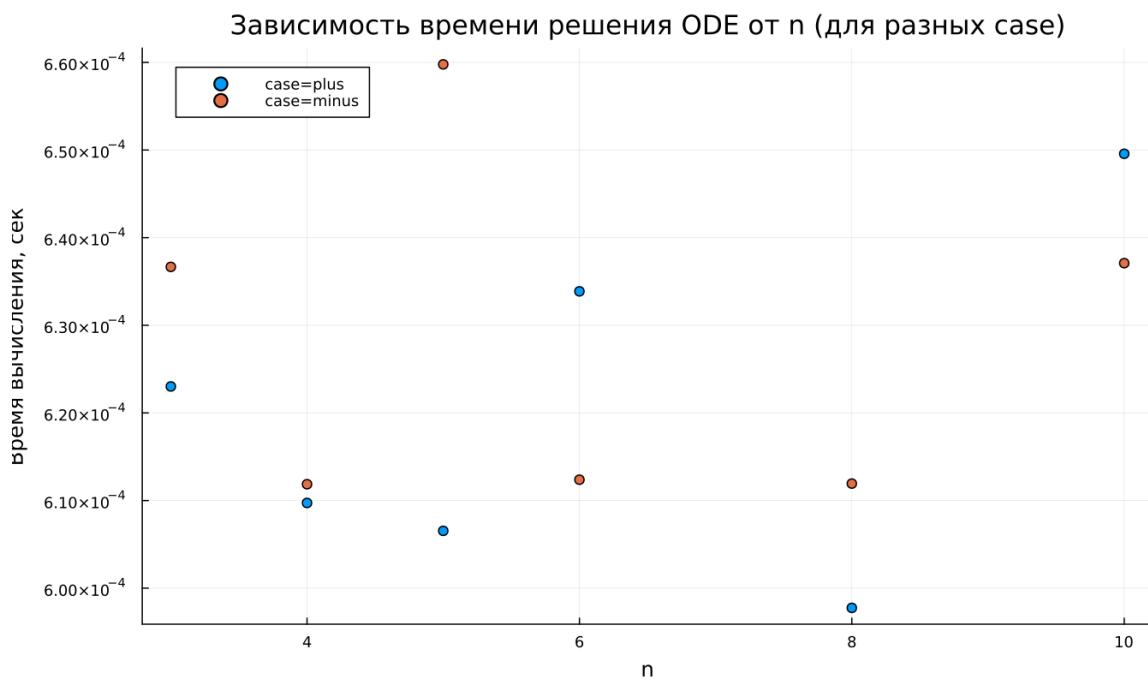


Рисунок 5.4: Зависимость времени вычисления от n

Анализ вычислительных затрат показал:

- время решения порядка 6×10^{-4} секунды;
- существенной зависимости от n не выявлено;
- небольшие вариации связаны с особенностями численного интегрирования.

6 Выводы

1. Движение катера описывается логарифмической спиралью.
2. Параметр n влияет на интенсивность радиального роста.
3. Начальные условия определяют масштаб траектории, не изменяя её форму.
4. Численное решение демонстрирует устойчивость и малые вычислительные затраты.

Полученные результаты согласуются с аналитическим решением дифференциального уравнения.

Список литературы

1. Задача о погоне