Отчет по лабораторной работа № 2

Математическое моделирование

Королев Иван Андреевич

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Задача о погоне. Рассмотрим задачу преследования браконьеров береговой охраной. Необходимо определить по какой траектории необходимо двигаться катеру, чтоб нагнать лодку.

2 Задание

Написать решение задачи о погоне на языке программирования Julia.

3 Теоретическое введение

Julia — высокоуровневый свободный язык программирования с динамической типизацией, созданный для математических вычислений. Эффективен также и для написания программ общего назначения. Синтаксис языка схож с синтаксисом других математических языков, однако имеет некоторые существенные отличия.

4 Выполнение лабораторной работы

4.1 Решение задача о погоне из примера лабораторной работы

Решение задача о погоне из примера лабораторной работы. Построение траектории движения катера береговой охраны и лодки при n=2. (рис. 4.1).

```
[5]: using DifferentialEquations, Plots

[3]: (1032225751%70)+1

[3]: 12

[27]: # Параметры
      k = 6.0  # начальное расстояние
      n = 2.0  # отношение скорости катера к скорости лодки
      r0_case1 = k / 3.0
      r0_case2 = k  # второй случай
      theta0_case1 = (0.0, pi)
      theta0_case2 = (-pi, pi)
      fi = 3 * pi / 4

[27]: 2.356194490192345

[29]: # Функция движения лодки
      x(t) = tan(fi) * t

[29]: x (generic function with 1 method)

[31]: # Функция движения катера
      du_dt(r, p, t) = r / sqrt(n^2 - 1)

[31]: du_dt (generic function with 1 method)
```

Рис. 4.1: Задаем начальные параметры задачи, функция движения лодки, функция движения катера

Решение ДУ для 1-го случая (рис. 4.2).

```
[33]: # Решение ДУ для 1 случая
      prob_case1 = ODEProblem(du_dt, r0_case1, theta0_case1)
      sol_case1 = solve(prob_case1, saveat=0.01)
```

Рис. 4.2: Решение ДУ для 1-го случая

Решение ДУ для 2-го случая (рис. 4.3).

```
[34]: # Решение ДУ для 2 случая
prob_case2 = ODEProblem(du_dt, r0_case2, theta0_case2)
sol_case2 = solve(prob_case2, saveat=0.01)
```

Рис. 4.3: Решение ДУ для 2-го случая

Подготовка данных для движения лодки (рис. 4.4).

```
[35]: # Подготовка данных для движения лодки
ugol = [fi for i in range(0, 15)]
x_lims = [x(i) for i in range(0, 15)]
```

```
[35]: 16-element Vector{Float64}:
   -0.0
   -1.0000000000000002
   -2.0000000000000004
   -3.000000000000001
   -4.000000000000001
   -5.000000000000001
   -6.000000000000002
   -7.000000000000002
   -8.000000000000002
   -9.000000000000002
  -10.000000000000002
  -11.000000000000002
  -12.000000000000004
  -13.000000000000004
  -14.000000000000004
  -15.000000000000004
```

Рис. 4.4: Подготовка данных для движения лодки

Траектория движения катера для 1-го случая (рис. 4.5).

```
[39]: # Построение графика
plot(title="Траектория движения катера береговой охраны и лодки", legend=:best)
plot!(sol_case1.t, sol_case1.u, proj=:polar, lims=(0, 15), label="Катер (случай 1)")
```

Рис. 4.5: Траектория движения катера для 1-го случая

Траектория движения катера для 2-го случая (рис. 4.6).

```
[41]: plot(title="Траектория движения катера береговой охраны и лодки", legend=:best)
plot!(sol_case2.t, sol_case2.u, proj=:polar, lims=(0, 15), label="Катер (случай 2)")
```

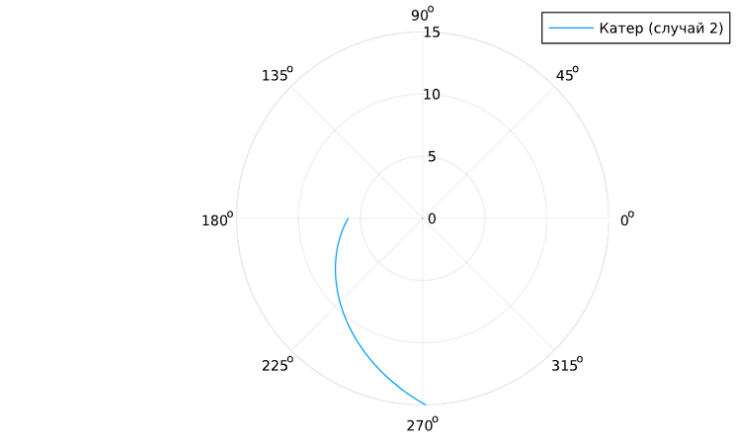

Рис. 4.6: Траектория движения катера для 2-го случая

4.2 Решение 12 Варианта самостоятельного задания

Задаем начальные параметры из условия задания, , функция движения лодки, функция движения катера, решение ДУ для 1-го случая, решение ДУ для 2-го случая, подготовка данных(рис. 4.7).

Вариант 24

[66]:
```
# Параметры
k = 11.4  # расстояние от лодки до катера
n = 4.1  # отношение скорости катера к скорости лодки
r0_case1 = k / 5.1  # начальное расстояние для 1 случая
r0_case2 = k / 3.1  # начальное расстояние для 2 случая
theta0_case1 = (0.0, 2 * pi)  # начальный угол для 1 случая
theta0_case2 = (-pi, pi)  # начальный угол для 2 случая
fi = 3 * pi / 4  # угол движения лодки

# Функция движения лодки
x(t) = tan(fi) * t  # Лодка движется по прямой

# Функция движения катера в полярных координатах
definition_f(r, p, t) = r / sqrt(15.81)

# Решение ДУ для 1 случая
prob_case1 = ODEProblem(definition_f, r0_case1, theta0_case1)
sol_case1 = solve(prob_case1, saveat=0.01)

# Решение ДУ для 2 случая
prob_case2 = ODEProblem(definition_f, r0_case2, theta0_case2)
sol_case2 = solve(prob_case2, saveat=0.01)

# Подготовка данных для движения лодки
ugol = [fi for i in range(0, 15)]  # Углы для построения прямой лодки
x_lims = [x(i) for i in range(0, 15)]  # Координаты лодки
```

Рис. 4.7: Решение задачи

Траектории движения катера и лодки, точки пересечения. (рис. 4.8), (рис. 4.9).

[67]:
```
# Построение графика траектории движения катера и лодки
plot(title="Траектория движения катера и лодки", legend=:best)
plot!(sol_case1.t, sol_case1.u, proj=:polar, lims=(0, 15), label="Катер (случай 1)")
plot!(sol_case2.t, sol_case2.u, proj=:polar, lims=(0, 15), label="Катер (случай 2)")
plot!(ugol, x_lims, proj=:polar, lims=(0, 15), label="Лодка")
```

[67]:

Рис. 4.8: Решение задачи

[68]:

```
# Определение точек пересечения
function find_intersection1(x)
    return (1140 * exp(10 * x) / sqrt(1581)) / 509  # Вычисление точки пересечения для 1 случая
end

function find_intersection2(x)
    return (114 * exp((10 * x / sqrt(1581)) + (10 * pi / sqrt(1581)))) / 31  # Вычисление точки пересечения для 2 случая
end

intersection1 = find_intersection1(fi)
intersection2 = find_intersection2(fi - pi)

# Вывод точек пересечения
println("Точка пересечения (случай 1): ", intersection1)
println("Точка пересечения (случай 2): ", intersection2)
```

```
Точка пересечения (случай 1): 9.628170843477646e8
Точка пересечения (случай 2): 6.651143558300665
```

Рис. 4.9: Решение задачи

5 Выводы

Рассмотрел задачу преследования браконьеров береговой охраной. Определил по какой траектории необходимо двигаться катеру, чтоб нагнать лодку.

Список литературы