Laboratory work report №2  
mathematical modeling

Задача о погоне

Выполнил: Леснухин Даниил Дмитриевич,  
НПИбд-02-22, 1132221553

Table of Contents

Список иллюстраций

# Цель работы

Построить математическую модель для выбора правильной стратегии при решении примера задачи о погоне.

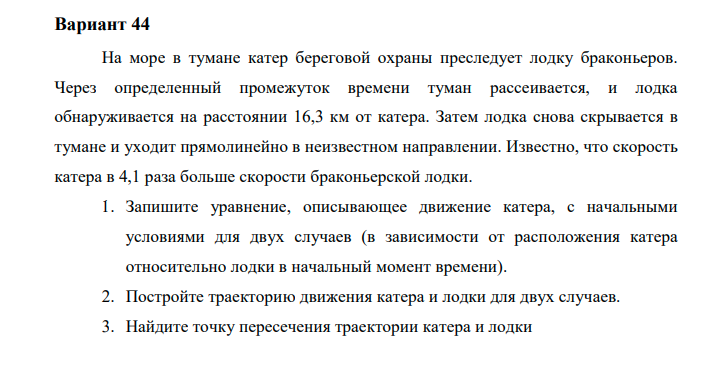
# Задание

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

Формула для выбора варианта лабораторной работы (1132221553%70) + 1 = 44

Постановка задачи конкретному варианту. Рис. 1



Лабораторная работа №2. Вариант 44

Уравнение движения катера

Обозначения:

* n=4.1 — отношение скорости катера к скорости лодки.
* k=16.3км — начальное расстояние между катером и лодкой.
* v — скорость лодки.
* nv — скорость катера.

Начальное положение лодки в момент обнаружения примем за полюс в полярных координатах.

Для двух случаев:

1. Катер позади лодки (x0​=−k).
2. Катер впереди лодки (x0​=+k).

Составим уравнения времени для прямолинейного движения:

t=x/v​, t=(k+x) / n−1 ​(в первом случае) t=x/v​, t=(x−k) / n+1 ​(во втором случае)

Из равенства времён найдём x для обоих случаев:

1. Для первого случая:

x1​=nk / (n−1)​

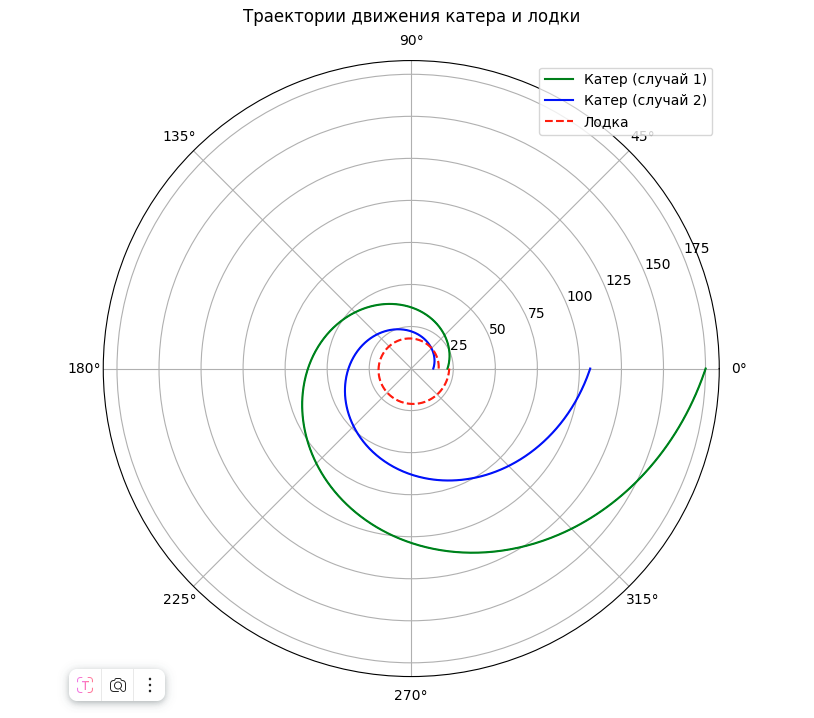
1. Для второго случая:

x2​=nk / (n+1)​

# Построение модели

# Импорт необходимых библиотек  
import numpy as np  
import matplotlib.pyplot as plt  
from scipy.integrate import solve\_ivp  
  
# Заданные параметры  
k = 16.3 # Начальное расстояние между катером и лодкой (км)  
n = 4.1 # Отношение скорости катера к скорости лодки  
v = 1 # Скорость лодки (условная единица)  
  
# Расчёт начального расстояния x для двух случаев  
x1 = (n \* k) / (n - 1) # Первый случай: катер позади лодки  
x2 = (n \* k) / (n + 1) # Второй случай: катер впереди лодки  
  
# Уравнение для траектории катера: 3r dtheta = dr  
def trajectory(theta, r):  
 """  
 Дифференциальное уравнение для радиуса траектории катера.  
 Параметры:  
 theta: угловая координата (в радианах)  
 r: текущий радиус (расстояние от полюса)  
  
 Возвращает производную радиуса по углу.  
 """  
 return r / 3  
  
# Построение траекторий  
# Параметры для решения дифференциального уравнения  
theta = np.linspace(0, 2 \* np.pi, 500) # Углы для полярной траектории  
  
# Решение для первого случая  
sol1 = solve\_ivp(trajectory, [0, 2 \* np.pi], [x1], t\_eval=theta)  
  
# Решение для второго случая  
sol2 = solve\_ivp(trajectory, [0, 2 \* np.pi], [x2], t\_eval=theta)  
  
# Прямолинейное движение лодки  
boat\_t = np.linspace(0, 2 \* np.pi, 500) # Временные точки  
boat\_r = k + v \* boat\_t # Радиус в зависимости от времени  
  
# Построение графиков  
plt.figure(figsize=(10, 8)) # Задаём размер графика  
  
# Траектория катера для первого случая  
plt.polar(sol1.t, sol1.y[0], label="Катер (случай 1)", color="green")  
  
# Траектория катера для второго случая  
plt.polar(sol2.t, sol2.y[0], label="Катер (случай 2)", color="blue")  
  
# Траектория лодки  
plt.polar(boat\_t, boat\_r, label="Лодка", color="red", linestyle="--")  
  
# Оформление графика  
plt.title("Траектории движения катера и лодки", va='bottom') # Заголовок графика  
plt.legend(loc="upper right") # Легенда  
  
# Отображение графика  
plt.show()

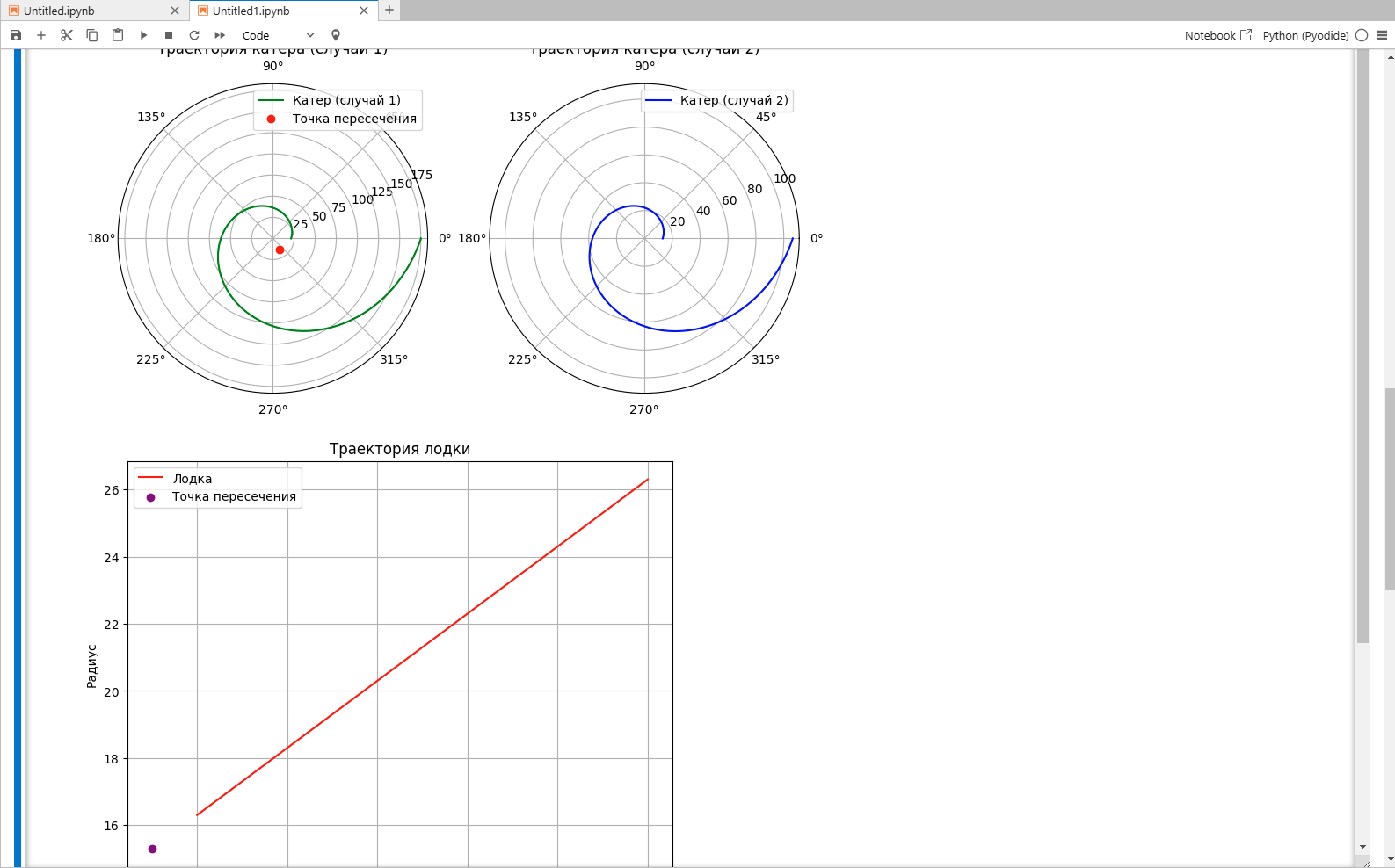
В результате получаем следующий график. Рис. 2

{#fig:002 width = 100% height = 100%}

# Численный поиск точки

Блок кода, отвчающий ща поиск точки пересечения Рис. 3

# Импорт необходимых библиотек  
import numpy as np  
import matplotlib.pyplot as plt  
from scipy.integrate import solve\_ivp  
from scipy.optimize import fsolve  
  
# Задаём параметры задачи  
k = 16.3 # Начальное расстояние между катером и лодкой (в км)  
n = 4.1 # Отношение скорости катера к скорости лодки  
v = 1 # Скорость лодки (условная единица)  
  
# Рассчитываем начальные радиусы для двух случаев  
x1 = n \* k / (n - 1) # Случай 1: катер отстает  
x2 = n \* k / (n + 1) # Случай 2: катер впереди  
  
# Уравнение движения катера  
def trajectory(theta, r):  
 return r / 3 # dr/dθ = r / 3  
  
# Углы для расчётов  
theta = np.linspace(0, 2 \* np.pi, 500)  
  
# Решение для случая 1  
sol1 = solve\_ivp(trajectory, [0, 2 \* np.pi], [x1], t\_eval=theta)  
  
# Решение для случая 2  
sol2 = solve\_ivp(trajectory, [0, 2 \* np.pi], [x2], t\_eval=theta)  
  
# Траектория лодки  
def boat\_trajectory(t):  
 return k + v \* t # Радиус лодки как функция времени  
  
# Создаём временной интервал для лодки  
time = np.linspace(0, 10, 500)  
r\_boat = boat\_trajectory(time)  
  
# Поиск точки пересечения  
def intersection(theta):  
 r\_boat\_at\_theta = k + v \* (theta \* 3 / (n - 1)) # Радиус лодки как функция угла  
 r\_patrol = x1 \* np.exp(theta / 3) # Радиус катера  
 return r\_patrol - r\_boat\_at\_theta  
  
# Находим пересечение  
theta\_intersection = fsolve(intersection, 1)[0] # Угол пересечения  
r\_intersection = x1 \* np.exp(theta\_intersection / 3) # Радиус пересечения  
  
print(f"Точка пересечения: угол θ = {theta\_intersection:.2f} рад, радиус r = {r\_intersection:.2f} км")  
  
# Построение графиков  
plt.figure(figsize=(10, 8))  
  
# Полярный график для случая 1  
plt.subplot(121, polar=True)  
plt.polar(sol1.t, sol1.y[0], label="Катер (случай 1)", color="green")  
plt.polar(theta\_intersection, r\_intersection, 'ro', label="Точка пересечения") # Отметка пересечения  
plt.title("Траектория катера (случай 1)")  
plt.legend(loc="upper right")  
  
# Полярный график для случая 2  
plt.subplot(122, polar=True)  
plt.polar(sol2.t, sol2.y[0], label="Катер (случай 2)", color="blue")  
plt.title("Траектория катера (случай 2)")  
plt.legend(loc="upper right")  
  
# Прямая траектория лодки  
plt.figure(figsize=(8, 6))  
plt.plot(time, r\_boat, label="Лодка", color="red")  
plt.scatter(theta\_intersection \* 3 / (n - 1), r\_intersection, color="purple", label="Точка пересечения")  
plt.title("Траектория лодки")  
plt.xlabel("Время")  
plt.ylabel("Радиус")  
plt.legend()  
plt.grid()  
  
# Показать графики  
plt.show()



Точка пересечения

# Вывод

В ходе выполнения лабораторной работы мы построили математическую модель для выбора правильной стратегии при решении примера задачи о погоне.