Laboratory work report №2
mathematical modeling

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

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

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

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

Постановка задачи конкретному варианту. Рис. 1

Вариант 44

На море в тумане катер береговой охраны преследует лодку браконьеров. Через определенный промежуток времени туман рассеивается, и лодка обнаруживается на расстоянии 16,3 км от катера. Затем лодка снова скрывается в тумане и уходит прямолинейно в неизвестном направлении. Известно, что скорость катера в 4,1 раза больше скорости браконьерской лодки.

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

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:

$x1=nk / (n-1)$

2. Для второго случая:

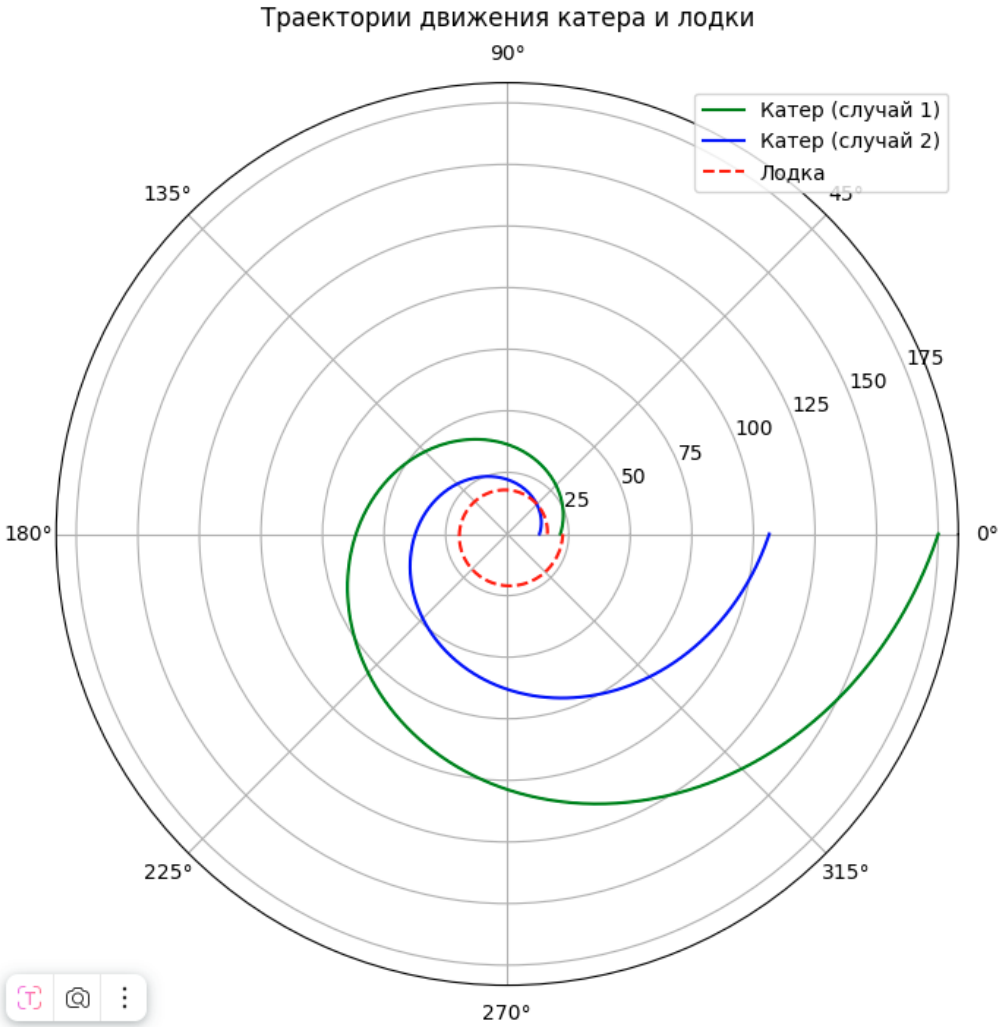
$x2=nk / (n+1)$

```
#
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

#
k = 16.3  #                              ( )
n = 4.1   #
v = 1     #          (          )

#                  x
x1 = (n * k) / (n - 1)  #          :
x2 = (n * k) / (n + 1)  #          :

#                    : 3r dtheta = dr
def trajectory(theta, r):
    """
                                        .
        :
    theta:              (       )
    r:          (             )
```

```
    """
    return r / 3

#
#
theta = np.linspace(0, 2 * np.pi, 500)  #

#
sol1 = solve_ivp(trajectory, [0, 2 * np.pi], [x1], t_eval=theta)

#
sol2 = solve_ivp(trajectory, [0, 2 * np.pi], [x2], t_eval=theta)

#
boat_t = np.linspace(0, 2 * np.pi, 500)  #
boat_r = k + v * boat_t  #

#
plt.figure(figsize=(10, 8))  #

#
plt.polar(sol1.t, sol1.y[0], label="    (    1)", color="green")

#
plt.polar(sol2.t, sol2.y[0], label="    (    2)", color="blue")

#
plt.polar(boat_t, boat_r, label="   ", color="red", linestyle="--")
```

```
#
plt.title("                    ", va='bottom')  #
plt.legend(loc="upper right")  #

#
plt.show()
```

В результате получаем следующий график. Рис. 2



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

Блок кода, отвчающий ща поиск точки пересечения Рис. 3

```
#
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp
from scipy.optimize import fsolve

#
k = 16.3  #                          (   )
n = 4.1   #
v = 1     #          (          )

#
x1 = n * k / (n - 1)  #      1:
x2 = n * k / (n + 1)  #      2:

#
def trajectory(theta, r):
    return r / 3  # dr/d  = r / 3

#
theta = np.linspace(0, 2 * np.pi, 500)
```

```
#             1
sol1 = solve_ivp(trajectory, [0, 2 * np.pi], [x1], t_eval=theta)

#             2
sol2 = solve_ivp(trajectory, [0, 2 * np.pi], [x2], t_eval=theta)

#
def boat_trajectory(t):
    return k + v * t  #

#
time = np.linspace(0, 10, 500)
r_boat = boat_trajectory(time)

#
def intersection(theta):
    r_boat_at_theta = k + v * (theta * 3 / (n - 1))  #
    r_patrol = x1 * np.exp(theta / 3)                #
    return r_patrol - r_boat_at_theta

#
theta_intersection = fsolve(intersection, 1)[0]  #
r_intersection = x1 * np.exp(theta_intersection / 3)  #

print(f"          :      = {theta_intersection:.2f}   ,      r = {r_intersection:.2f}

#
plt.figure(figsize=(10, 8))
```

```
#                  1
plt.subplot(121, polar=True)
plt.polar(sol1.t, sol1.y[0], label="    (    1)", color="green")
plt.polar(theta_intersection, r_intersection, 'ro', label="          ")  #
plt.title("          (    1)")
plt.legend(loc="upper right")

#                  2
plt.subplot(122, polar=True)
plt.polar(sol2.t, sol2.y[0], label="    (    2)", color="blue")
plt.title("          (    2)")
plt.legend(loc="upper right")

#
plt.figure(figsize=(8, 6))
plt.plot(time, r_boat, label="   ", color="red")
plt.scatter(theta_intersection * 3 / (n - 1), r_intersection, color="purple", label="
plt.title("         ")
plt.xlabel("   ")
plt.ylabel("    ")
plt.legend()
plt.grid()

#
plt.show()
```

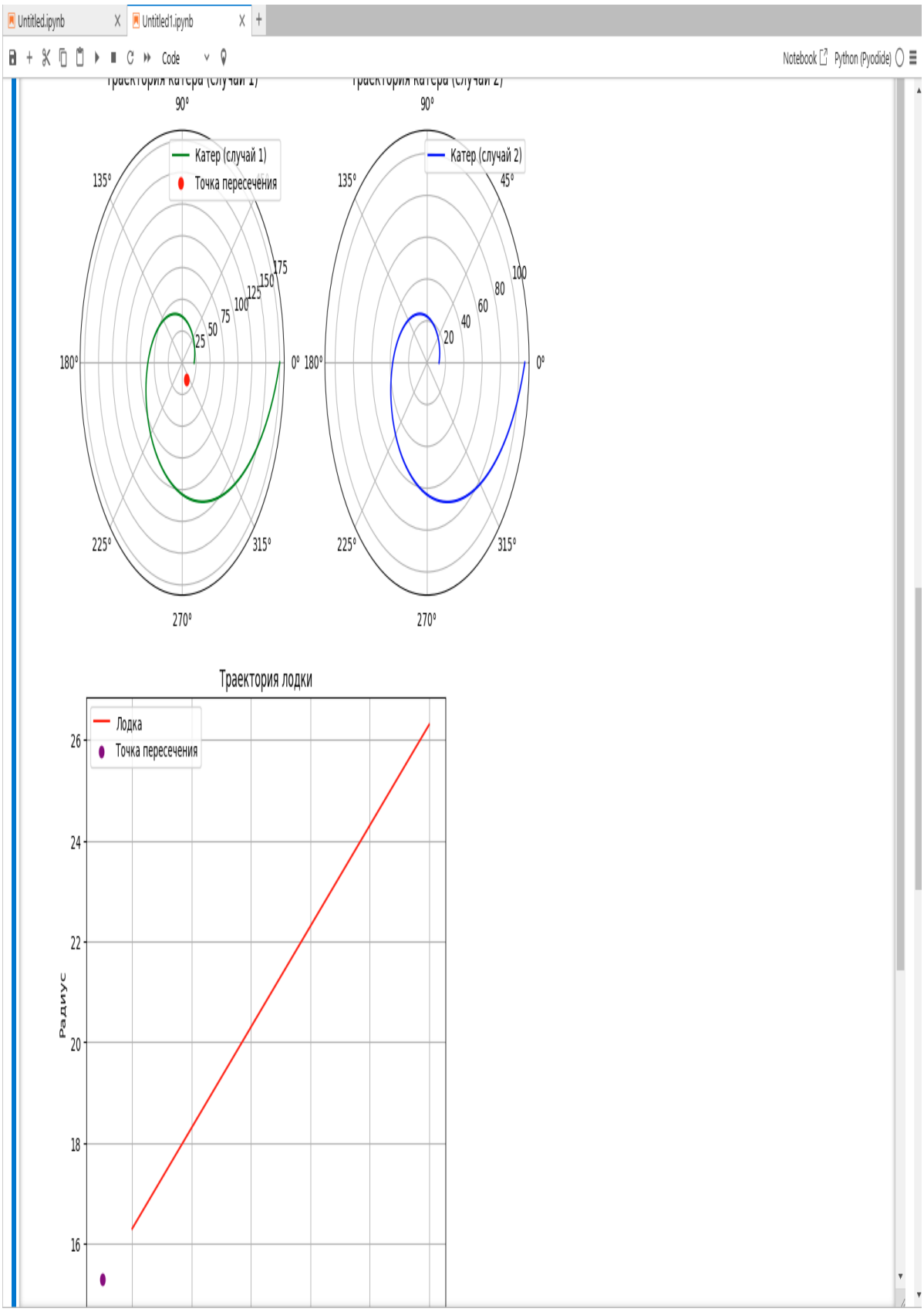


Рис. 1: Точка пересечения

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