МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра прикладної математики

Звіт

із лабораторної роботи №1

з дисципліни «Теорія керування» на тему:

«Навігаційна задача швидкодії»

|  |  |
| --- | --- |
| Виконав: | Керівник: |
| студент групи КМ-63 | *Професор Норкін В. І.* |
| *Артеменко Я. К.* |  |

Київ — 2020

# **ЗМІСТ**

[1 ВСТУП 2](#_Toc35962965)

[2 ОСНОВНА ЧАСТИНА 3](#_Toc35962966)

[2.1 Постановка задачі 3](#_Toc35962967)

[2.2 Описання методу 4](#_Toc35962968)

[2.3 Порядок виконання роботи 5](#_Toc35962969)

[2.4 Контрольний приклад 5](#_Toc35962970)

[2.5 Дослідження змін 7](#_Toc35962971)

[3 ВИСНОВКИ 10](#_Toc35962972)

[4 СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 11](#_Toc35962973)

[ДОДАТКИ 12](#_Toc35962974)

[Додаток А (код програми) 12](#_Toc35962975)

# **1 ВСТУП**

Метою даної лабораторної роботи являється розробка програмного забезпечення для розв’язання поставленої задачі, а саме навігаційної задачі швидкодії. Програмне забезпечення дозволяється організовувати за допомогою будь-якої мови програмування.

# **2 ОСНОВНА ЧАСТИНА**

# **2.1 Постановка задачі**

За допомогою будь-якої мови програмування потрібно розв’язати навігаційну задачу швидкодії, а сам результат потрібно зобразити графічно.

Сама задача стосується корабля, що рухається зі швидкістю, постійної за величиною відносно течії, швидкість якої може змінюватися в міру віддалення від берегової лінії (рис. 1.1).

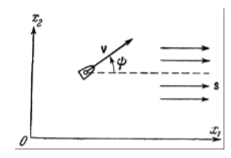


Рисунок 1.1 – Схема руху

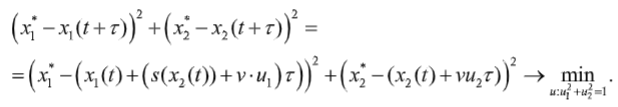
Потрібно визначити програму управління рулями, при якій човен досягає заданої кінцевої точки з заданого початкового пункту за мінімальний час.

Початкові умови:

* v – швидкість човна відносно течії;
* l – відстань від початкової точки човна до кінцевої;
* γ - кут нахилу вектора руху відносно x2;
* s0 – швидкість течії при х2 = 0;
* f(x2) – функція для описання залежності між швидкістю течі та х2.

# **2.2 Описання методу**

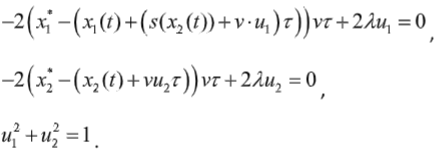
Будемо шукати (міопичне = короткозоре) управління як рішення наступного завдання:



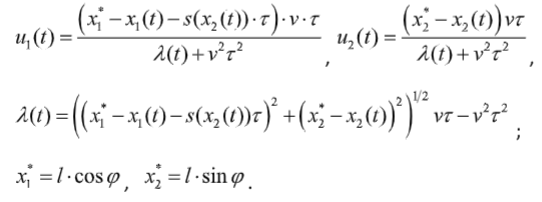
Рішення даної задачі оптимізації відбувається методом множників Лагранжа, тому розглянемо задачу:



Необхідні умови екстремуму мають наступний вигляд:



Звідки отримуємо:



# **Порядок виконання роботи**

1. Задати вхідні дані , де n – номер студента в списку групи.
2. Запрограмувати рекурентні співвідношення - в системі Matlab (або Octava і т.і.).
3. Побудувати графік траєкторії судна . На графіках вказувати:

А) назву графіка і назви осей координат,

Б) точку призначення і значення фіксованих параметрів,

В) легенду (який колір відповідає якомусь значенню параметра),

Г) час досягнення цілі

1. Дослідити залежність траєкторії от варіації параметрів .
2. При яких значеннях параметрів, в тому числі за скільки ітерацій і за який час, корабель досягає точки призначення (для даної (міопичної) стратегії управління)?
3. Підготувати звіт про роботу в електронному вигляді (з графіками, висновками і лістингом програми).
4. Надіслати звіт викладачеві на електронну адресу.

# **2.4 Контрольний приклад**

Нехай задані початкові умови (Варіант №1):

* v - 1 м/с
* l - 1 м
* γ - π/25
* s0 - 1 м/с
* f(x2) = x2

При таких даних траєкторія руху човна буде мати наступний вигляд:

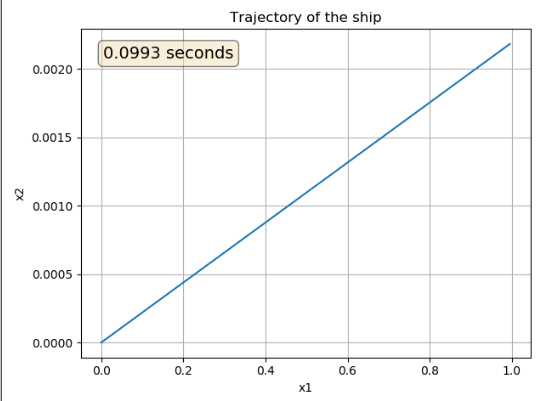


Рисунок 2.1 – Траєкторія руху човна при заданих умовах

Також на графіку можна побачити час, за який човен пройшов шлях від початкової точки до кінцевої. В даному випадку човен досягнув кінцевої точки за 0.0993 секунди.

# **2.5 Дослідження змін**

Далі будуть розглядатись різні випадки і як при певних даних буде реагувати програма:

1. Коли швидкість течії дорівнює 0 м/с:

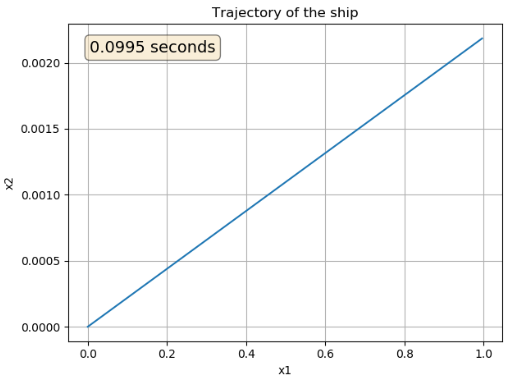


Рисунок 2.2 - Траєкторія руху човна при швидкості течії 0 м/с

1. Коли швидкість течії дорівнює 50 м/с

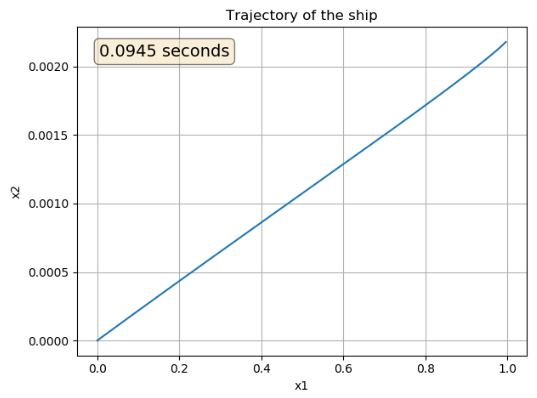


Рисунок 2.3 – Траєкторія руху човна при швидкості течії 50 м/с

1. При збільшенні швидкості корабля до 20 м/с

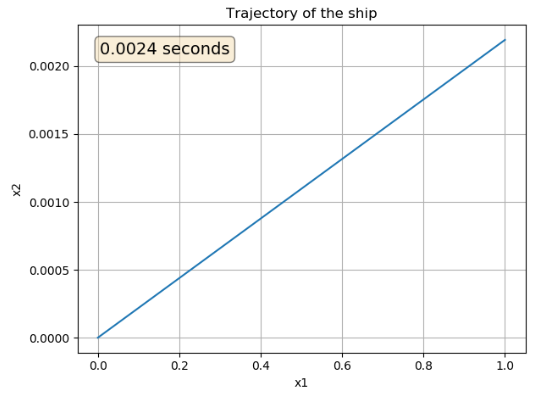


Рисунок 2.4 – Траєкторія руху човна при швидкості човна 20 м/с

1. Збільшення відстані до 100 метрів:

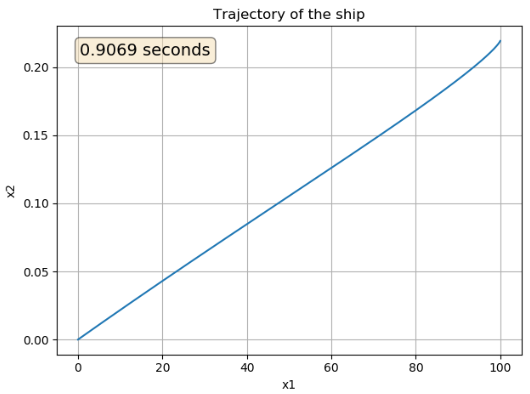


Рисунок 2.5 – Траєкторія руху човна при відстані 100 метрів

1. Збільшення кута в 150 разів, при цьому значення кута буде дорівнювати 18.84:

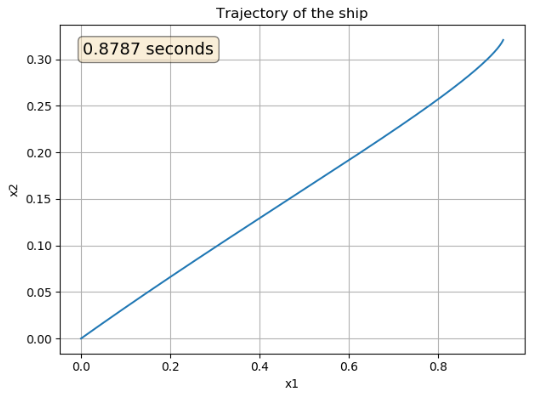


Рисунок 2.6 – Траєкторія руху човна при збільшенні кута в 150 разів

# **3 ВИСНОВКИ**

Під час виконання даної лабораторної роботи було розроблено програмне забезпечення, яке реалізовує розв’язання поставленої задачі, а саме навігаційної задачі швидкодії. Також було розглянуто виконання програми при зміні вхідних даних на інші та розглянуто результати виконання при різних вхідних даних.

# **4 СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

1. Методичні вказівки до лабораторної роботи
2. Лейтман Дж. Введение в теорию оптимального управления. М.: Наука, 1968. 192 стр.
3. Алексеев Е.Р., Чеснокова О.В. Введение в Octave для инженеров и математиков. М.: ALT Linux, 2012. 368 с.

# **ДОДАТКИ**

# **Додаток А (код програми)**

from math import pi, cos, sin, radians, sqrt  
import numpy as np  
import matplotlib.pyplot as plt  
  
  
  
variant = 1  
s0 = variant  
V = variant  
l = variant  
fi = (variant \* pi) / 25  
  
def s(x2):  
 return s0 \* x2  
  
x1\_ = l \* cos(radians(fi))  
x2\_ = l \* sin(radians(fi))  
print(x1\_, x2\_)  
  
N = 10000  
teta = (l \* V)/N  
  
def lambd(t, x1\_k, x2\_k):  
 result = sqrt(((x1\_-x1\_k)-(s(x2\_k))\*teta)\*\*2+(x2\_-x2\_k)\*\*2)\*V\*teta-(V\*\*2)\*(teta\*\*2)  
 return result  
  
def u1(t, x1\_k, x2\_k):  
 result = ((x1\_-x1\_k-s(x2\_k)\*teta)\*V\*teta)/(lambd(t, x1\_k, x2\_k)+(V\*\*2)\*(teta\*\*2))  
 return result  
  
def u2(t, x1\_k, x2\_k):  
 result = ((x2\_-x2\_k)\*V\*teta)/(lambd(t, x1\_k, x2\_k)+(V\*\*2)\*(teta\*\*2))  
 return result  
  
def x1(t, x1\_k, x2\_k):  
 result = x1\_k + (s(x2\_k)+V\*u1(t, x1\_k, x2\_k))\*teta  
 return result  
  
def x2(t, x1\_k, x2\_k):  
 result = x2\_k + V\*u2(t, x1\_k, x2\_k)\*teta  
 return result  
  
  
  
line = np.linspace(0, 1, 10001)  
  
all\_x1 = [0]  
all\_x2 = [0]  
for t in line:  
 x1\_k = all\_x1[-1]  
 x2\_k = all\_x2[-1]  
 all\_x1.append(x1(t, x1\_k, x2\_k))  
 all\_x2.append(x2(t, x1\_k, x2\_k))  
 if round(all\_x1[-1], 2) == round(x1\_, 2) \  
 and round(all\_x2[-1], 2) == round(x2\_, 2):  
 print(t)  
 time = t  
 break  
  
  
fig, ax = plt.subplots()  
ax.plot(all\_x1, all\_x2)  
  
ax.set(xlabel='x1', ylabel='x2',  
 title='Trajectory of the ship')  
ax.grid()  
props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)  
textstr = str(round(time, 4)) + ' seconds'  
ax.text(0.05, 0.95, textstr, transform=ax.transAxes, fontsize=14,  
 verticalalignment='top', bbox=props)  
  
plt.show()