Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра комп’ютерного моделювання процесів і систем

ЗВІТ

з лабораторної роботи №2

“РОЗРОБКА ТА МОДЕЛЮВАННЯ НЕЧІТКОЇ СИСТЕМИ УПРАВЛІННЯ”

з курсу

«Нейронні мережі та методи проектування»

Виконав: студент групи ІКМ-М222к Черкас Ю.В.

Перевірив: професор, д.т.н. Успенський В.Б.

Харків 2023р

Ціль: реалізація алгоритму управління обертання штучним супутником Землі (ШСЗ) на основі нечіткого виведення.

Постановка задачі

Вхідними даними програми є початкові умови: кут відхилення ШСЗ та кутова швидкість на нульовий момент часу (Таблиця 1).

Таблиця 1 – Початкові умови

Номер варіанту	Початкове значення кута ϕ, град.	Початкове значення кутової швидкості ω, град./с
15	-40	0

Вихідні дані – результат моделювання, містить поточні значення куту, кутової швидкості та керованого кутового прискорення. Дані виводяться у файл у 4 стовпця із заданим кроком інтегрування dt. Перший стовпець – час (у секундах), другий – кут ϕ (у градусах), третій – кутова швидкість ω (у град./с), четвертий – кутове прискорення μ (у град./с^2). Результати надати у вигляді графіків від часу.

Програма та вкладені алгоритми використовують наступні **константи**:

- $dt = 0.01$ с – крок інтегрування рівнянь;
- T = 50 с – тривалість моделювання;
- $d\tilde{\mu} = 0.1$ – безрозмірний крок інтегрування;
- $\mu_{min} = -\pi/36$ рад./с^2;
- $\mu_{max} = \pi/36$ рад./с^2;

Будемо вважати, що робочими діапазонами є такі:

- $\phi \in [-\pi\,;+\pi]$ рад. (±180°);
- $\omega \in [-\pi/9\,;+\pi/9]$ рад./с (±20°/с);
- $\mu \in [-\pi/36\,;+\pi/36]$ рад./с^2 (±5°/с^2).

Лістинг програми моделювання управління ШСЗ на мові Python наведений в Додатку А та Додатку Б. На рисунку 1 наведений графік зміни у часі кута орієнтації ШСЗ ϕ, кутової швидкості ω та керуючого моменту μ для першого варіанту бази правил. На рисунку 2 зображений аналогічний графік для другого варіанту бази правил.

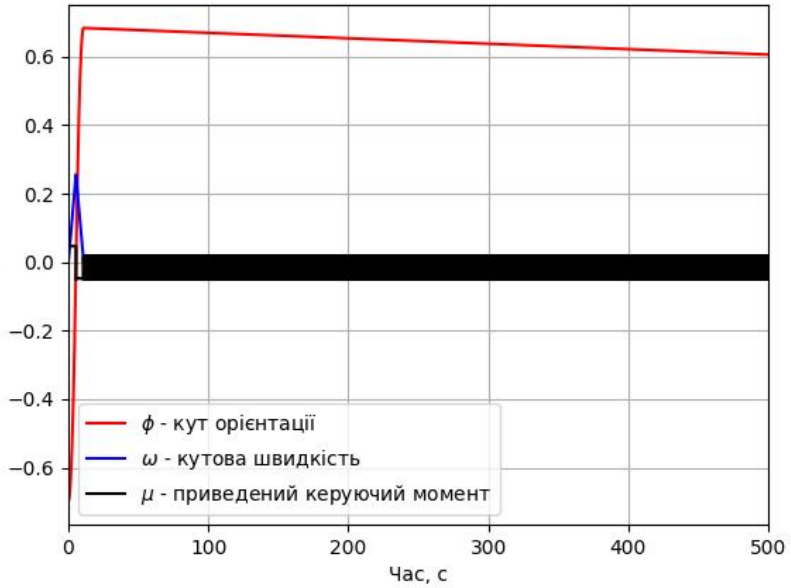

Рисунок 1- Графік залежності кута ϕ, кутової швидкості ω та керуючого моменту μ від часу для першого варіанту бази правил

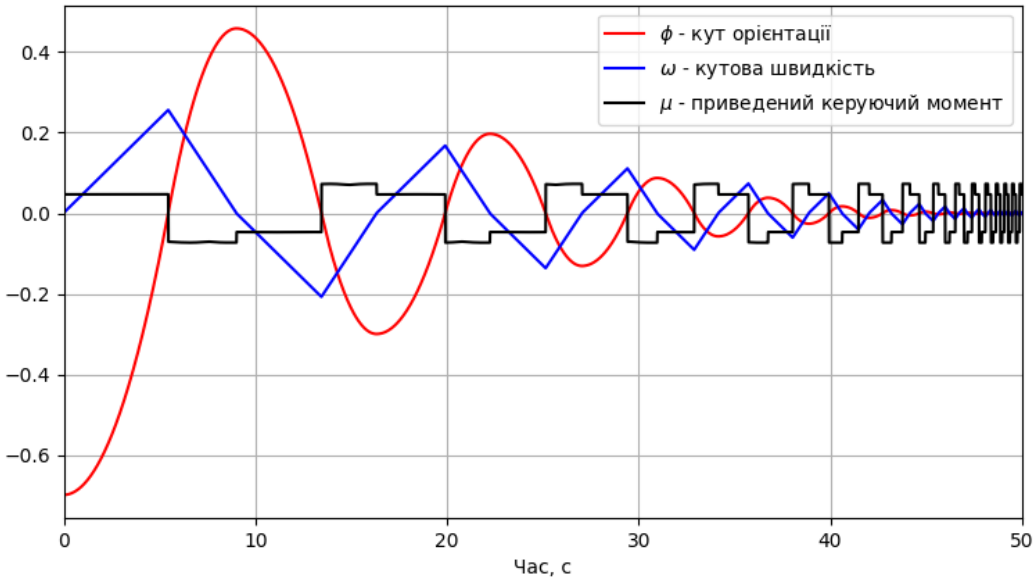

Рисунок 2- Графік залежності кута ϕ, кутової швидкості ω та керуючого моменту μ від часу для другого варіанту бази правил

Висновки

На даній лабораторній роботі ми розглянули проблему стабілізації положення ШСЗ ($\phi(t) \to 0, \quad \omega(t) \to 0$) навколо повздовжньої осі. Як бачимо дана задача може бути успішно вирішеною за допомогою процедури нечіткого виведення, потребуючи при цьому мінімального знання законів фізики та без застосування вищої математики. Проте ключову роль в даному випадку грає якість бази правил створеної експертом. При невдалому підборі бази правил, час на стабілізацію може бути завеликим або вона взагалі не наступить (рисунок 1).

По результатам моделювання бачимо, що положення ШСЗ практично повністю стабілізується наприкінці 50 секундного інтервалу (рисунок 2). Варто відмітити, що значення приведеного керуючого моменту змінюються в часі ступінчасто. Вочевидь це являється результатом розбиття неперервних інтервалів зміни кута ϕ, кутової швидкості ω та керуючого моменту μ на дискретну кількість термів. Також заслуговує на увагу той факт, що навіть при наближені значень кута ϕ та кутової швидкості ω до 0, значення керуючого моменту μ продовжує коливатись відносно нуля з певною амплітудою, що являється не доцільним. Це потребую подальшого розгляду та ймовірно класичний алгоритм здатний вирішити цю задачу більш оптимально.

Додаток А

Лістинг основної програми на мові Python

```
import numpy as np
import matplotlib.pyplot as plt
from helpers import get_general_membership_function


t = 0
t_max = 50
dt = 0.01
dmu = 0.1

phi=-40 * np.pi /180
omega = 0
phi_min = -np.pi
phi_max = np.pi
omega_min = -np.pi/9
omega_max = np.pi/9
mu_min = -np.pi/36
mu_max = np.pi/36

phi_t, omega_t, mu_t, t_n = [], [], [], []
afile = open("results.csv", "w")
afile.write("t,phi,omega,mu\n")

while t < t_max - 0.5 * dt:
    phi_ = 200 * (phi - phi_min) / (phi_max - phi_min) - 100
    omega_ = 200 * (omega - omega_min) / (omega_max - omega_min) - 100

    mu_ = -100
    s1 = 0
    s2 = 0

    while mu_ < 100 - 0.5 * dmu:
        xi_dash = get_general_membership_function(phi_, omega_, mu_)
        s1 = s1 + mu_ * xi_dash * dmu
        s2 = s2 + xi_dash * dmu
        mu_ = mu_ + dmu

    mu_dash = s1 / s2
    mu = (mu_dash + 100) * (mu_max - mu_min) / 200 + mu_min

    phi = phi + omega * dt
    omega = omega + mu * dt
```

```
    phi_t.append(phi)
    omega_t.append(omega)
    mu_t.append(mu)
    t_n.append(t)

    afile.write(f'{t},{phi},{omega},{mu}\n')

    t = t + dt



plt.plot(t_n, phi_t, 'r')
plt.plot(t_n, omega_t, 'b')
plt.plot(t_n, mu_t, 'k')
plt.legend([r'$\phi$ - кут орієнтації', r'$\omega$ - кутова швидкість',
r'$\mu$ - приведений керуючий момент'])
plt.grid(True)
plt.xlabel('Час, с')
plt.show()
```

Додаток Б

Лістинг допоміжних методів на мові Python

```python
import numpy as np


rules = np.array([
    [1, 1, 6],
    [1, 2, 6],
    [1, 3, 6],
    [1, 4, 6],
    [1, 5, 6],
    [1, 6, 6],
    [2, 1, 6],
    [2, 2, 6],
    [2, 3, 6],
    [2, 4, 5],
    [2, 5, 5],
    [2, 6, 5],
    [3, 1, 6],
    [3, 2, 6],
    [3, 3, 6],
    [3, 4, 5],
    [3, 5, 5],
    [3, 6, 5],
    [4, 1, 2],
    [4, 2, 2],
    [4, 3, 2],
    [4, 4, 1],
    [4, 5, 1],
    [4, 6, 1],
    [5, 1, 2],
    [5, 2, 2],
    [5, 3, 2],
    [5, 4, 1],
    [5, 5, 1],
    [5, 6, 1],
    [6, 1, 1],
    [6, 2, 1],
    [6, 3, 1],
    [6, 4, 1],
    [6, 5, 1],
    [6, 6, 1]
])
```

```python
def get_membership_function(x, a, b, c, d):
    if x <= a:
        return 0
    elif x > a and x <= b:
        return (x - a) / (b - a)
    elif x > b and x <= c:
        return 1
    elif x > c and x <= d:
        return (d - x) / (d - c)
    else:
        return 0



def get_term_membership_function(term, x):
    match term:
        case 1:
            return get_membership_function(x, -1000, -200, -100, -50)
        case 2:
            return get_membership_function(x, -100, -50, -50, -10)
        case 3:
            return get_membership_function(x, -50, -10, 0, 0)
        case 4:
            return get_membership_function(x, 0, 0, 10, 50)
        case 5:
            return get_membership_function(x, 10, 50, 50, 100)
        case 6:
            return get_membership_function(x, 50, 100, 200, 1000)



def get_general_membership_function(phi, omega, mu):
    xi_dash = 0

    for rule in rules:
        alfa = get_term_membership_function(rule[0], phi)
        beta = get_term_membership_function(rule[1], omega)
        gama = min(alfa, beta)
        delta = get_term_membership_function(rule[2], mu)
        xi = min(gama, delta)
        if xi_dash < xi:
            xi_dash = xi

    return xi_dash
```