**Пономаренко, ІКМ-220Б**

**Лабораторна робота 2**

**Розробка та моделювання нечіткої системи управління**

**Мета роботи** − реалізація алгоритму управління на основі нечіткого виведення та порівняння його роботи із класичним алгоритмом.

**Хід роботи**

**Вхідними даними** програми є початкові умови: кут відхилення ШСЗ та кутова швидкість на нульовий момент часу

|  |  |  |
| --- | --- | --- |
| Номер варіанту | Початкове значення кута , град. | Початкове значення кутової швидкості , град./с |
| 10 | 60 | 5 |

**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  (у градусах), третій – кутова швидкість  (у град./с), четвертий – кутове прискорення (у град./с2). Результати надати у вигляді графіків від часу.

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

с – крок інтегрування рівнянь за методом Ейлера. Також це крок визначення поточного значення керованого кутового прискорення ;

T = 800 c – тривалість моделювання;



 рад./с2;

 рад./с2;

Chislo\_pravil=36 – число правил у базі.

Функції приналежноств задані наступним чином:

1-НВ: ,

2-НС: ,

3-НМ: ,

4-ПМ: ,

5-ПС: ,

6-ПВ: .

База правил:

|  |  |  |  |
| --- | --- | --- | --- |
| Номер  висловлювання | Номер терму для  вхідної ЛЗ "кут", n1 | Номер терму для вхідної ЛЗ "кутова швидкість", n2 | Номер терму для вихідної ЛЗ "кутове прискорення", m |
| 1 | 1 | 1 | 6 |
| 2 | 1 | 2 | 6 |
| 3 | 1 | 3 | 6 |
| 4 | 1 | 4 | 5 |
| 5 | 1 | 5 | 5 |
| 6 | 1 | 6 | 5 |
| 7 | 2 | 1 | 6 |
| 8 | 2 | 2 | 6 |
| 9 | 2 | 3 | 6 |
| 10 | 2 | 4 | 5 |
| 11 | 2 | 5 | 5 |
| 12 | 2 | 6 | 5 |
| 13 | 3 | 1 | 6 |
| 14 | 3 | 2 | 6 |
| 15 | 3 | 3 | 6 |
| 16 | 3 | 4 | 5 |
| 17 | 3 | 5 | 5 |
| 18 | 3 | 6 | 5 |
| 19 | 4 | 1 | 6 |
| 20 | 4 | 2 | 5 |
| 21 | 4 | 3 | 4 |
| 22 | 4 | 4 | 2 |
| 23 | 4 | 5 | 2 |
| 24 | 4 | 6 | 1 |
| 25 | 5 | 1 | 5 |
| 26 | 5 | 2 | 4 |
| 27 | 5 | 3 | 4 |
| 28 | 5 | 4 | 1 |
| 29 | 5 | 5 | 1 |
| 30 | 5 | 6 | 1 |
| 31 | 6 | 1 | 3 |
| 32 | 6 | 2 | 3 |
| 33 | 6 | 3 | 3 |
| 34 | 6 | 4 | 1 |
| 35 | 6 | 5 | 1 |
| 36 | 6 | 6 | 1 |

**Постановка задачі:** Розглядається одновісне кероване обертання ШСЗ навколо повздовжньої осі.

Нехай OX, OY – нерухомі у просторі осі; Ox, Oy – осі, пов'язані із ШСЗ (рис.1.2). На рисунку зображено центральну частину ШСЗ (вигляд спереду) та для наочності панелі сонячний батарей. Положення пов'язаної системи координат відрізняється від положення нерухомих осей на кут . Повздовжня вісь спрямована "на глядача". Датчик кутової швидкості, розташований на ШСЗ, вимірює кутову швидкість  , яка є похідною кута  за часом. Керуючий вплив реалізується пристроєм, що створює момент навколо повздовжньої осі.

x

X

Y

y

O

φ

Рис.1.2 – ШСЗ та системи координат

Призначенням алгоритму управління є визначення такої величини керуючого впливу з врахуванням поточних значень кута  та кутової швидкості , щоб забезпечити стійке довготривале співпадіння осей x, y із однойменними осями X, Y.

Математична модель задачі досить проста та має вигляд

, (1.1)

, (1.2)

де  – поточний час;  – змінні у часі кутова швидкість та кут;  – приведений керуючий момент, змінний у часі. Він співпадає із кутовим прискоренням, яке система згідно технічних умов може формувати "на свій розсуд". Початкові умови  та  є відомими. Задача побудови алгоритму в цих умовах полягає у визначенні залежності  від  та , тобто , яка забезпечить для розв'язку рівнянь (1.1), (1.2) умови

. (1.3)

Для подальшого важливо визначити діапазони для змінних, в яких розглядатимемо задачу. Це особливо важливо при розробці саме нечіткого алгоритму.

Будемо вважати, що робочими діапазонами є такі:

 рад. (±180°);

 рад./с (±10°/с);

 рад./с2 (±5°/с2).

На рис. 1, 2 показані блок-схеми роботи алгоритму.

Головна програма

ввести початкові дані ; t=0

, 

, , 

Обчислити загальну ФП за алг. ЗФП

















вивести до файлу дані 

Головна програма

так

так

алгоритм нечіткого виведення

Рисунок 1 - Блок-схема головної програми

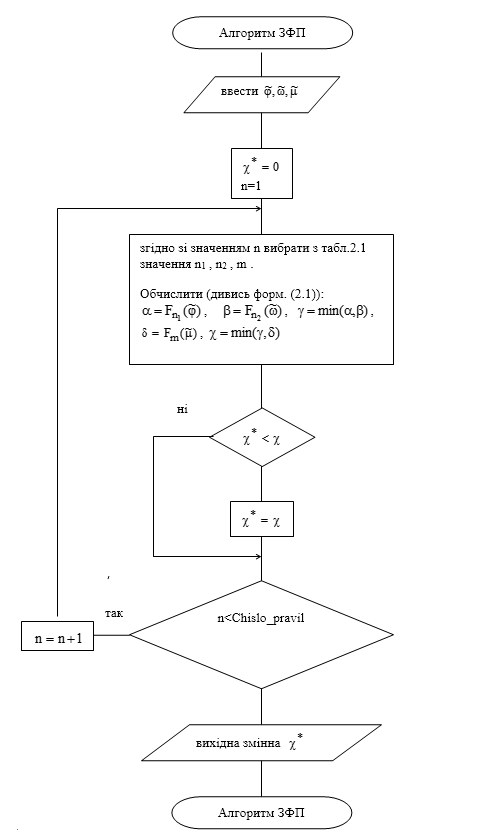


Рисунок 2 - Блок-схема алгоритму визначення загальної ФП вихідної змінної

Дані для роботи алгоритму (data.json):

{

"data": [

[**1**, **1**, **6**],

[**1**, **2**, **6**],

[**1**, **3**, **6**],

[**1**, **4**, **5**],

[**1**, **5**, **5**],

[**1**, **6**, **5**],

[**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**, **6**],

[**4**, **2**, **5**],

[**4**, **3**, **4**],

[**4**, **4**, **2**],

[**4**, **5**, **2**],

[**4**, **6**, **1**],

[**5**, **1**, **5**],

[**5**, **2**, **4**],

[**5**, **3**, **4**],

[**5**, **4**, **1**],

[**5**, **5**, **1**],

[**5**, **6**, **1**],

[**6**, **1**, **3**],

[**6**, **2**, **3**],

[**6**, **3**, **3**],

[**6**, **4**, **1**],

[**6**, **5**, **1**],

[**6**, **6**, **1**]

],

"F": [

[-**100**, -**100**, -**100**, -**50**],

[-**100**, -**50**, -**50**, -**10**],

[-**50**, -**10**, **0**, **0**],

[**0**, **0**, **10**, **50**],

[**10**, **50**, **50**, **100**],

[**50**, **100**, **100**, **100**]

]

}

Нижче наведено код моєї програми на python.

**import** **math**

**import** **json**

**import** **pandas** **as** **pd**

file = "data.json"

out\_file = "out.xlsx"

dt = **0.1** # крок інтегрування рівнянь

T = **800** # тривалість моделювання, cек

dmu\_ = **1** # безрозмірний крок інтегрування чисельника та знаменника

mumin = -math.pi / **36** # рад./с2

mumax = math.pi / **36** # рад./с2

# Я задаю

fimin = -math.pi # мінімальниц кут відхилення ШСЗ

fimax = math.pi # максимальний кут відхилення ШСЗ

omegamin = -**10** / **180** \* math.pi # мінімальна кутова швидкість

omegamax = **10** / **180** \* math.pi # максимальна кутова швидкість

df = pd.DataFrame(columns=["t", "fi", "omega", "mu"]) # save results here

**with** open(file, 'r', encoding="utf-8") **as** f:

data = json.load(f)

# clear out file

**with** open(out\_file, "w", encoding="utf-8") **as** f:

f.write("")

**def** **Ft**(x, data):

a, b, c, d = data

**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)

**elif** d <= x:

**return** **0**

**else**:

**print**("Шось пішло не по плану.. x =", x)

# формула 2.1

**def** **F**(n, x):

**return** Ft(x, data["F"][n-**1**])

**def** **zfp**(fi\_, omega\_, mu\_):

chistar = **0**

**for** n1, n2, m **in** data["data"]:

alpha = F(n1, fi\_)

beta = F(n2, omega\_)

gamma = min(alpha, beta)

delta = F(m, mu\_)

chi = min(gamma, delta)

chistar = chi **if** (chistar < chi) **else** chistar

**return** chistar

**def** **main**(fi, omega, t=**0**):

iteration = **0**

**while** (t < T - **0.5** \* dt):

fi\_ = **200** / (fimax - fimin) \* (fi - fimin) - **100**

omega\_ = **200** / (omegamax - omegamin) \* (omega - omegamin) - **100**

mu\_, s1, s2 = -**100**, **0**, **0**

**while** mu\_ < **100** - **0.5** \* dmu\_:

chistar = zfp(fi\_, omega\_, mu\_)

s1 = s1 + mu\_ \* chistar \* dmu\_

s2 = s2 + chistar \* dmu\_

mu\_ = mu\_ + dmu\_

mu\_star = s1 / s2

mu = (mu\_star + **100**) / **200** \* (mumax - mumin) + mumin

fi += omega \* dt

omega += mu \* dt

t += dt

# save to results array

df.loc[iteration] = [t, fi, omega, mu]

iteration += **1**

**print**(t, T)

df.to\_excel(out\_file, index=False)

**if** \_\_name\_\_ == "\_\_main\_\_":

fi = **60** / **180** \* math.pi # Початкове значення кут, рад

omega = **5** / **180** \* math.pi # Початкове значення кутової швидкості, рад/c

main(fi, omega)

Після завершення роботи программи треба побудувати графіки:  
**import** **pandas** **as** **pd**

**import** **matplotlib.pyplot** **as** **plt**

**import** **numpy** **as** **np**

file = 'out.xlsx'

# Read the CSV file into a DataFrame

df = pd.read\_excel(file)

# Convert fi, omega, and mu to radians

df['fi'] = np.rad2deg(df['fi'])

df['omega'] = np.rad2deg(df['omega'])

df['mu'] = np.rad2deg(df['mu'])

# Create three plots for fi, omega, and mu

fig, axs = plt.subplots(**3**, **1**, figsize=(**8**, **10**))

axs[**0**].plot(df['t'], df['fi'])

axs[**0**].set\_ylabel('fi')

axs[**1**].plot(df['t'], df['omega'])

axs[**1**].set\_ylabel('omega')

axs[**2**].plot(df['t'], df['mu'])

axs[**2**].set\_ylabel('mu')

axs[**2**].set\_xlabel('Time')

plt.show()

Я отримав такі результати (рис. 3):

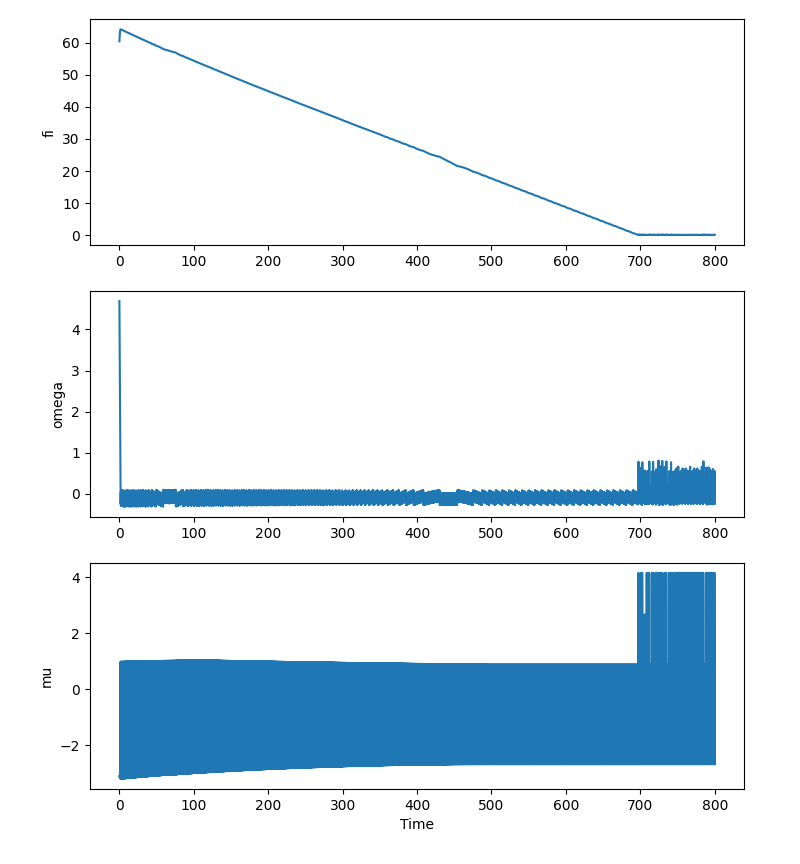


Рисунок 3 - графік залежності кута, кутової швидкості, кутового прискорення від часу

Як ми бачимо, кут прямує до нуля і на 0 врівноважується. Для цього йому потрібно приблизно 700 секунд.

Давайте подивимось на результати отримані класичним методом (рис. 4):

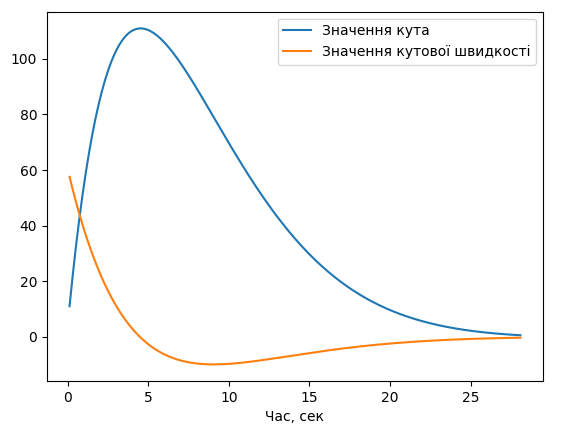


Рисунок 4 - залежність кута та кутової швидкості від часу (класичний метод)

**Висновки**

У результаті порівняння роботи нечіткого алгоритму управління з класичним алгоритмом можна зробити наступні висновки:

* Нечіткий алгоритм управління дає більш точний та ефективний результат у порівнянні з класичним алгоритмом. Це можливо завдяки використанню нечітких змінних та правил, що дозволяють більш точно моделювати реальний світ.
* Нечіткий алгоритм управління дозволяє знизити кількість помилок та забезпечує більш стабільну роботу системи. Це забезпечується тим, що нечіткі змінні дозволяють враховувати велику кількість факторів, які можуть впливати на роботу системи, а також застосуванням правил, що дозволяють коректно обробляти різноманітні ситуації.
* Нечіткий алгоритм управління вимагає більш складних налаштувань та підтримки, що може бути проблемою в деяких випадках. Однак, на практиці використання нечіткого алгоритму управління може дозволити отримати значно кращі результати у порівнянні з класичним алгоритмом.

Отже, можна стверджувати, що реалізація алгоритму управління на основі нечіткого виведення є дуже ефективним та перспективним напрямком в області управління системами.