Міністерство освіти і науки України
Національний університет водного господарства
та природокористування
Навчально-науковий інститут енергетики, автоматики
та водного господарства
Кафедра автоматизації, електротехнічних та
комп'ютерно-інтегрованих технологій

**04-03-421М**

# МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з навчальної дисципліни
«**Чисельні методи та моделювання**»
(Частина 3. Моделювання) для здобувачів вищої освіти
першого (бакалаврського) рівня за освітньо-професійною
програмою «Автоматизація, комп'ютерно-інтегровані
технології та робототехніка» спеціальності
174 «Автоматизація, комп'ютерно-інтегровані технології та
робототехніка» денної та заочної форм навчання

Рекомендовано науково-
методичною радою
з якості ННІ ЕАВГ
Протокол № 4 від 17.12.2024 р.

Рівне – 2024

Методичні вказівки до виконання лабораторних робіт з навчальної дисципліни «Чисельні методи та моделювання» для здобувачів вищої освіти першого (бакалаврського) рівня за освітньо-професійною програмою «Автоматизація, комп'ютерно-інтегровані технології та робототехніка» спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка» денної та заочної форм навчання. Частина 3. [Електронне видання] / Мащенко В. А., Сафоник А. П., Подвишенний В. С. – Рівне : НУВГП, 2024. – 77 с.

**Укладачі:** Мащенко В. А., кандидат фізико-математичних наук, доцент, доцент кафедри автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій; Сафоник А. П., доктор технічних наук, професор, професор кафедри автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій; ПодвишеннийВ. С., викладач кафедри автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій.

**Відповідальний за випуск**: Древецький В. В., доктор технічних наук, професор, завідувач кафедри автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій.

**Керівник групи забезпечення**: Христюк А. О., кандидат технічних наук, доцент, доцент кафедри автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій.

**Зміст**

# Вступ

Навчальна дисципліна «Чисельні методи та моделювання» носить важливий характер для ознайомлення студентів із класичними і сучасними методами наближених обчислень та основними обчислювальними алгоритмами для розв'язування прикладних задач у вибраному програмному забезпеченні та основоположних принципів комп'ютерного моделювання фізико-технічних процесів.

У методичних вказівках викладено основи теорії моделювання і базові принципи моделювання динамічних та технічних систем. Особливу увагу приділено технології імітаційного моделювання та програмній реалізації імітаційних моделей, плануванню та проведенню імітаційних експериментів, обробці та інтерпретації результатів моделювання.

Окремо приділено значну увагу теоретичним основам прикладних методів сучасної технології нечіткого моделювання в контексті розв'язку практичних задач із використанням спеціальних програмних засобів системи MATLAB.

## Лабораторна робота № 15
## „Моделювання поведінки динамічної системи”

### 15.1. Теоретичні відомості

Розглянемо систему диференціальних рівнянь 1-го порядку:

$$\begin{cases} \dfrac{dx}{dt} = P(x, y), \\ \dfrac{dy}{dt} = Q(x, y). \end{cases} \tag{15.1.1}$$

Обмежимося розглядом тільки стаціонарних (автономних) систем, тобто систем, в яких $P$ і $Q$ не залежать явно від часу. У найпростішому випадку $P$ і $Q$ – лінійні функції:

$$\begin{cases} \dfrac{dx}{dt} = ax + by, \\ \dfrac{dy}{dt} = cx + dy. \end{cases} \tag{15.1.2}$$

Особливими точками системи (15.1.1) називаються точки, в яких $P$ і $Q$ одночасно дорівнюють нулю.

Очевидно, що система (15.1.2) має єдину особливу точку – (0, 0). Будемо шукати розв’язок системи (15.1.2) у вигляді:

$$x = Ae^{\lambda t}, \qquad y = Be^{\lambda t}.$$

Підставимо розв’язки у систему (15.1.2) і скоротивши на загальний множник $e^{\lambda t}$, отримаємо:

$$\begin{cases} (a - \lambda)A + bB = 0, \\ cA + (d - \lambda)B = 0. \end{cases} \tag{15.1.3}$$

Система (15.1.3) має нетривіальний розв’язок, якщо її визначник рівний нулю:

$$\lambda^2 - (a + d)\lambda + ad - cb = 0. \tag{15.1.4}$$

Рівняння (15.1.4) називається характеристичним. Маємо його розв’язки:

$$\lambda_{1,2} = \frac{a+d}{2} \pm \sqrt{\frac{(a+d)^2}{4} + bc - ad} = \frac{\mathrm{tr}\mathbf{M}}{2} \pm \sqrt{\left(\frac{\mathrm{tr}\mathbf{M}}{2}\right)^2 - \det\mathbf{M}},$$

де $\mathbf{M}$ – матриця коефіцієнтів системи (15.1.2). Якщо корені характеристичного рівняння різні і не рівні нулю ($\lambda_1 \neq \lambda_2 \neq 0$), то розв'язок має такий вигляд:

$$\begin{cases} x = C_1 e^{\lambda_1 t} + C_2 e^{\lambda_2 t}, \\ y = C_1 \chi_1 e^{\lambda_1 t} + C_2 \chi_2 e^{\lambda_2 t}. \end{cases} \tag{15.1.5}$$

Заміною змінних можна звести систему (15.1.2) до системи:

$$\begin{cases} \frac{d\tilde{x}}{dt} = S_1 \tilde{x}, \\ \frac{d\tilde{y}}{dt} = S_2 \tilde{y}. \end{cases} \tag{15.1.6}$$

або до диференціального рівняння:

$$\frac{d\tilde{y}}{d\tilde{x}} = \frac{S_2 \tilde{y}}{S_1 \tilde{x}}, \tag{15.1.7}$$

де

$$\begin{cases} \tilde{x} = \alpha x + \beta y, \\ \tilde{y} = \gamma x + \delta y. \end{cases} \tag{15.1.8}$$

Постійні $\alpha, \beta, \gamma, \delta$ знаходяться наступним чином: про диференціюємо систему рівнянь (15.1.8)

$$\begin{cases} \frac{d\tilde{x}}{dt} = \alpha \frac{dx}{dt} + \beta \frac{dy}{dt}, \\ \frac{d\tilde{y}}{dt} = \gamma \frac{dx}{dt} + \delta \frac{dy}{dt}. \end{cases} \tag{15.1.9}$$

та підставимо замість $\frac{dx}{dt}$ та $\frac{dy}{dt}$ їх вирази із системи (15.1.2)

$$\begin{cases} \frac{d\tilde{x}}{dt} = \alpha(ax + by) + \beta(cx + dy), \\ \frac{d\tilde{y}}{dt} = \gamma(ax + by) + \delta(cx + dy). \end{cases} \tag{15.1.10}$$

Враховуючи систему (.6), отримаємо:

$$\begin{cases} \alpha(ax+by)+\beta(cx+dy)=S_1(\alpha x+\beta y), \\ \gamma(ax+by)+\delta(cx+dy)=S_2(\gamma x+\delta y). \end{cases} \qquad (15.1.11)$$

Рівність повинна виконуватися при всіх $x$ та $y$, відповідно коефіцієнти при змінних у лівій та правій частинах рівняння повинні співпадати:

$$\begin{cases} (a-S_1)\alpha+c\beta=0, \\ b\alpha+(d-S_1)\beta=0, \end{cases} \quad \begin{cases} (a-S_2)\gamma+c\delta=0, \\ b\gamma+(d-S_2)\delta=0. \end{cases} \qquad (15.1.12)$$

Ми отримали системи лінійних однорідних рівнянь відносно $\alpha, \beta, \gamma, \delta$. Система має нетривіальний розв'язок, якщо її визначник рівний нулю, тобто $S_i$ повинні задовольняти рівнянню:

$$(a-S_i)(d-S_i)-bc=0, \qquad i=1,2. \qquad (15.1.13)$$

яке співпадає із рівнянням (15.1.4). Таким чином, $S_i$ є коренями характеристичного рівняння (.4), тобто $S_1=\lambda_1$, $S_2=\lambda_2$. При цьому $\alpha^2+\beta^2>0$, $\gamma^2+\delta^2>0$. Перша умова із системи (15.1.12) дає відношення:

$$\frac{\alpha}{\beta}=\frac{c}{\lambda_1-a}=\frac{\lambda_1-d}{b}, \qquad (15.1.14)$$

друга –

$$\frac{\gamma}{\delta}=\frac{c}{\lambda-a}=\frac{\lambda-d}{b}. \qquad (15.1.15)$$

Розв'язок рівняння (15.1.7) має такий вигляд:

$$\tilde{y}=\tilde{C}\left|\tilde{x}\right|^{\frac{\lambda_2}{\lambda_1}}. \qquad (15.1.16)$$

У випадку дійсних коренів рівняння (15.1.16) визначає в залежності від знаку показника степеня або сімейство парабол, або сімейство гіпербол.

Афінні перетворення (15.1.8) не змінюють характеру особливої точки. Заміна змінних призводить тільки до її масштабування і повороту координатних осей. Визначимо нове положення осей після афінного перетворення у випадку дійсних

коренів характеристичного рівняння. Якщо $\tilde{x} = 0$, то із першого рівняння системи (15.1.8) слідує, що $y = -\frac{\alpha x}{\beta}$. При $\tilde{y} = 0$ із другого рівняння системи знаходимо, що $y = -\frac{\gamma x}{\delta}$. Кутові коефіцієнти виражаються через параметри вихідної системи у випадку із формулами (15.1.14) та (15.1.15).

У випадку комплексних коренів нові змінні $\tilde{x}$ та $\tilde{y}$ будуть комплексними.

Нехай:

$$\lambda_1 = R + iI\ ,\ \ \lambda_2 = R - iI\ ,\ \ \tilde{x} = u + iv\ ,\quad \tilde{y} = u - iv\ ,$$

де $R, I, u, v$ – дійсні величини. Тоді система рівнянь (15.1.6) запишеться у вигляді:

$$\begin{cases} \dfrac{du}{dt} + i\dfrac{dv}{dt} = (R + iI)(u + iv), \\ \dfrac{du}{dt} + i\dfrac{dv}{dt} = (R - iI)(u - iv). \end{cases} \tag{15.1.17}$$

Звідки

$$\begin{cases} \dfrac{du}{dt} = uR - vI, \\ \dfrac{dv}{dt} = vR + uI. \end{cases} \tag{15.1.18}$$

Тоді рівняння інтегральних кривих має такий вид:

$$\frac{dv}{du} = \frac{vR + uI}{uR - vI}. \tag{15.1.19}$$

Після переходу до полярних координат $u = r\cos(\varphi)$, $v = r\sin(\varphi)$ отримаємо рівняння логарифмічної спіралі:

$$r = C\exp\left(\frac{R}{I}\varphi\right). \tag{15.1.20}$$

У випадку чисто уявних коренів рівняння (15.1.20) переходить у рівняння сімейства кіл $r = C$.

Якщо корені характеристичного рівняння кратні $\lambda_1 = \lambda_2 = \lambda$, то заміною змінних:

$$\tilde{x} = ax + \frac{b-c}{2} y, \qquad \tilde{y} = y \tag{15.1.21}$$

система може бути приведена до виду:

$$\begin{cases} \dfrac{d\tilde{y}}{dt} = \tilde{x} + \lambda\tilde{y}, \\ \dfrac{d\tilde{x}}{dt} = \lambda\tilde{x}. \end{cases} \tag{15.1.22}$$

або рівняння:

$$\frac{d\tilde{y}}{d\tilde{x}} = \frac{\tilde{x} + \lambda\tilde{y}}{\lambda\tilde{x}}. \tag{15.1.23}$$

Розв'язок рівняння (.23) має вигляд:

$$\tilde{y} = \frac{1}{\lambda}\tilde{x}\, ln|\tilde{x}| + \tilde{C}\tilde{x}. \tag{15.1.24}$$

Рівняння (15.1.6) та (15.1.23) називаються рівняннями, записаними в канонічному вигляді.

Якщо хоч один із коренів характеристичного рівняння рівний нулю, то рівняння (15.1.2) лінійно залежні, їх можна скоротити, і рівняння (15.1.2) будуть мати такий вигляд:

$$\frac{dy}{dx} = k. \tag{15.1.25}$$

## 15.2. Алгоритм аналізу результатів

Тип особливої точки автономної лінійної системи (15.1.2) визначається характеристичним рівнянням (15.1.4).

1. Корені дійсні, різні і одного знаку ($0 < \mathbf{M} < \left(\left(\frac{\mathrm{tr}\mathbf{M}}{2}\right)^2\right)$):

   рівняння (15.1.16) визначає сімейство парабол. Якщо корені характеристичного рівняння додатні, то розв'язки будуть прямувати до нескінченності. У випадку від'ємних коренів розв'язки із зростанням часу будуть

необмежено зменшуватися, фазові траєкторії будуть прямувати до нуля.

а) Корені додатні ($\mathrm{tr}\mathbf{M} < 0$): особлива точка – стійкий вузол.

б) Корені від'ємні ($\mathrm{tr}\mathbf{M} > 0$): особлива точка – нестійкий вузол.

2. Корені дійсні, однакові і відмінні від нуля: особлива точка – або вироджений вузол або дикретичний (зірковий) вузол. Дикретичний вузол можливий тільки у випадку:

$$\frac{dy}{dx} = \frac{ay}{ax}.$$

3. Корені дійсні, різні і різних знаків ($\det\mathbf{M} < 0$): рівняння (15.1.16) визначає сімейство гіпербол, особлива точка – сідло.

4. Корені комплексно-спряжені (але не чисто уявні) ($\det\mathbf{M} > \left(\frac{\mathrm{tr}\mathbf{M}}{2}\right)^2$).

а) дійсна частина від'ємна ($\mathrm{tr}\mathbf{M} < 0$): особлива точка – стійкий вузол;

б) дійсна частина додатня ($\mathrm{tr}\mathbf{M} > 0$): особлива точка – нестійкий вузол;

5. Корені чисто явні ($\mathrm{tr}\mathbf{M} = 0, \det\mathbf{M} > 0$): особлива точка – центр.

6. Один із коренів рівний нулю: особливі точки повністю заповнюють із координатних осей.

Ці правила зручно представити у вигляді рис. 15.2.1.

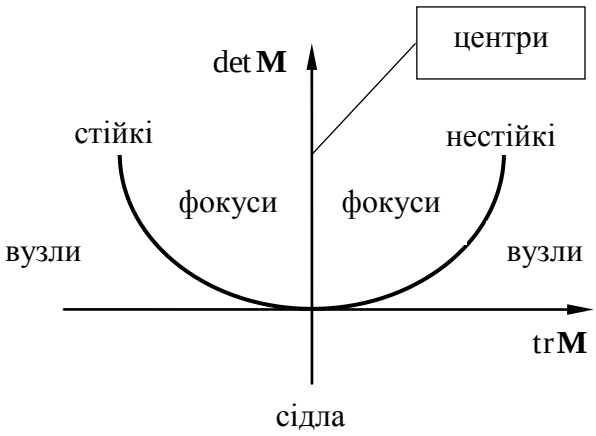


Рис. 15.2.1. Вид особливої точки при різних значеннях визначника і сліду матриці системи рівнянь.

Добрий Для аналізу нелінійної системи проводять її лінеаризацію. Для цього:

1. Визначають особливі точки системи.
2. В околі особливих точок заміною змінних приводять вихідну систему до виду:

$$\begin{cases} \dfrac{dx}{dt} = ax + by + p(x, y), \\ \dfrac{dy}{dt} = cx + dy + q(x, y). \end{cases}$$

Причому:

$$\lim_{\substack{x \to x_0 \\ y \to y}} \frac{p(x, y)}{ax + by} = 0, \qquad \lim_{\substack{x \to x_0 \\ y \to y}} \frac{q(x, y)}{cx + dy} = 0,$$

якщо $(x_0, y_0)$ – особлива точка.

3. Обмежуємося тільки врахуванням лінійних доданків.

4. Визначають тип особливих точок лінеарезованої системи. Характери особливих точок лінеаризованої і вихідної системи співпадають, крім наступних випадків:
   а) якщо особлива точка лінеаризованої системи – центр, то особлива точка вихідної системи або центр, або фокус.
   б) якщо хоча б один із коренів лінеарезованої ситсеми рівний нулю, то для аналізу особливої точки вихідної системи треба додаткові дослідження.

## 15.3. Приклад виконання

### 15.3.1. Модель Вольтерри

Віто Вольтерра запропонував модель „хижак – жертва”. Нехай на деякій замкнутій території живуть два види: вегетаріанці – жертви, що харчуються підніжним кормом, якого є у надлишку, та хижаки, що полюють на жертв. В якості пари хижак – жертва можуть виступати вовки і зайці, щуки і карасі та інші.

Якщо не має хижаків, то жертви розмножувалися нескінченно і їх чисельність описується рівнянням Мальтуса:

$$\frac{dx}{dt} = \alpha x, \qquad (15.3.1)$$

де $x$ – чисельність виду; $\alpha > 0$ – коефіцієнт приросту.

Якщо не було б жертв, то хижаки від без годівлі поступово вимирали:

$$\frac{dy}{dt} = -\gamma x, \qquad y = y_0 e^{-\gamma t}, \qquad (15.3.2)$$

де $\gamma > 0$ – коефіцієнт зменшення; $y$ – чисельність хижаків у даний момент часу; $y_0$ – чисельність хижаків у початковий момент часу. Зростання чисельності жертв перешкоджає їх зустріч із хижаками, частота яких пропорційна як числу жертв, так і числу хижаків – $xy$. Тоді швидкість зміни чисельності жертв описується рівнянням:

$$\frac{dx}{dt} = x(\alpha - \beta y), \qquad (15.3.3)$$

де $\beta > 0$ – коефіцієнт зменшення жертв при зустрічі із хижаками.

Аналогічно, зустріч хижака із жертвою збільшує ймовірність виживання хижака, тобто приросту популяції хижаків:

$$\frac{dy}{dt} = -y(\gamma - \delta x), \qquad (15.3.4)$$

де $\delta > 0$ – коефіцієнт, що залежить від того, як часто зустріч хижака із жертвою закінчується трапезою.

Проаналізуємо отриману нелінійну систему:

$$\begin{cases} \dfrac{dx}{dt} = x(\alpha - \beta y), \\ \dfrac{dy}{dt} = -y(\gamma - \delta x). \end{cases} \qquad (15.3.5)$$

Знайдемо стаціонарний, тобто незалежне від часу стан системи. Якщо чисельність популяцій постійна, то їх похідні по часу рівні нулю

$$\begin{cases} 0 = x(\alpha - \beta y), \\ 0 = -y(\gamma - \delta x). \end{cases} \qquad (15.3.6)$$

Звідки

$$x_1 = 0, \quad y_1 = 0, \quad x_2 = \frac{\gamma}{\delta}, \quad y_2 = \frac{\alpha}{\beta}. \qquad (15.3.7)$$

Похідні перетворюються у нуль на прямих

$$x = \frac{\gamma}{\delta}, \quad y = \frac{\alpha}{\beta}. \qquad (15.3.8)$$

відповідно, чисельність популяцій має екстремум.

Ситуація, коли в наявності є тільки одна популяція (або хижаків, або жертв), аналізувалася вище при виводі системи рівнянь (15.3.5). На фазовій площині їм відповідає промінь $y = 0$, який приходить в початок координат та промінь $x = 0$, який виходить із початку координат. Очевидно, що початок координат є особливою типу сідло. Фазові траєкторії в околі цієї

точки ведуть себе як гіперболи і напрямлені проти руху часової стрілки.

Більш цікава стаціонарна точка $(x_2, y_2)$. Розкладемо праві частини системи (15.3.5) в околі стаціонарної точки та обмежимося випадком малих відхилень від положення рівноваги $\eta$ та $\xi$:

$$\begin{cases} \eta = x - x_2 = x - \dfrac{\gamma}{\delta}, \\ \xi = y - y_2 = y - \dfrac{\alpha}{\beta}. \end{cases} \tag{15.3.9}$$

Тоді система перетвориться до такого виду:

$$\begin{cases} \dfrac{d\eta}{dt} = -\beta\eta\xi - \dfrac{\gamma\beta}{\delta}\xi, \\ \dfrac{d\xi}{dt} = \delta\eta\xi + \dfrac{\alpha\delta}{\beta}\eta. \end{cases} \tag{15.3.10}$$

Оскільки $\eta$ та $\xi$ малі величини, знехтуємо доданками $\eta\xi$. Тоді

$$\begin{cases} \dfrac{d\eta}{dt} = -\dfrac{\gamma\beta}{\delta}\xi, \\ \dfrac{d\xi}{dt} = \dfrac{\alpha\delta}{\beta}\eta. \end{cases} \tag{15.3.11}$$

Характеристичне рівняння системи:

$$\begin{vmatrix} -\lambda & -\dfrac{\gamma\beta}{\delta} \\ \dfrac{\alpha\delta}{\beta} & -\lambda \end{vmatrix} = 0 \tag{15.3.11}$$

має такі корені

$$\lambda_1 = i\sqrt{\alpha\gamma}, \qquad \lambda_2 = -i\sqrt{\alpha\gamma} \tag{15.3.12}$$

### 15.3.2. Моделювання у додатку Simulink системи MATLAB

Для побудови фазових траєкторій побудуємо побудуємо модель Вольтерри (15.3.5) у додатку Simulink (рис. 15.3.1).

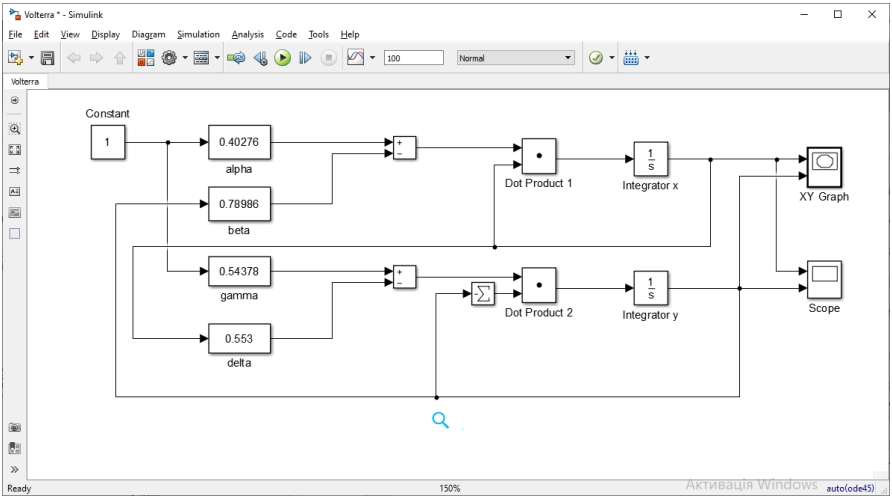


Рис. 15.3.1. Модель (15.3.5) у додатку Simulink

Корені характеристичного рівняння чисто уявні, відповідно, особлива точка – центр. В околі особливої точки фазова траєкторія представляє собою еліпс (рис. 15.3.2).

Чисельність популяцій відчуває коливання, які не співпадають за фазою (рис. 15.3.3).

Отримана залежність узгоджується із експериментальними даними. Однак модель є нестійкою: при стрибкоподібній зміні числа особин в одній із популяцій (наприклад із-за міграції тварин, діяльності людини або інших факторів, які не враховані у моделі) коливання назавжди змінять свій характер, системи перейде із однієї фазової траєкторії на іншу. Крім того, особлива точка типу центр не є грубою, тобто при введенні у модель навіть малих поправок, вона може змінити свій характер.

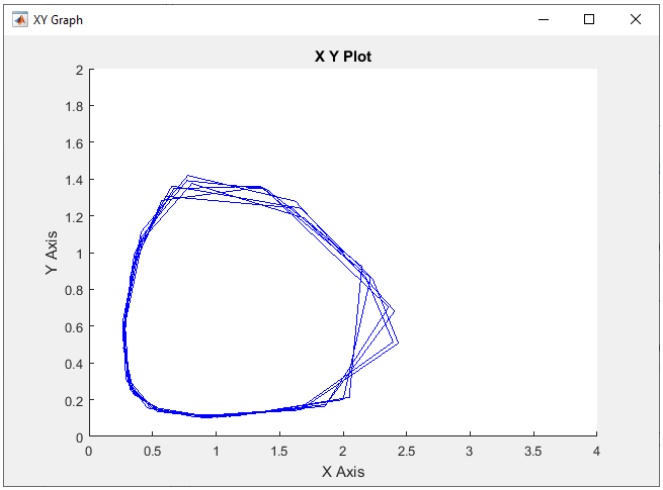


Рис. 15.3.2. Фазовий портрет моделі Вольтерри.

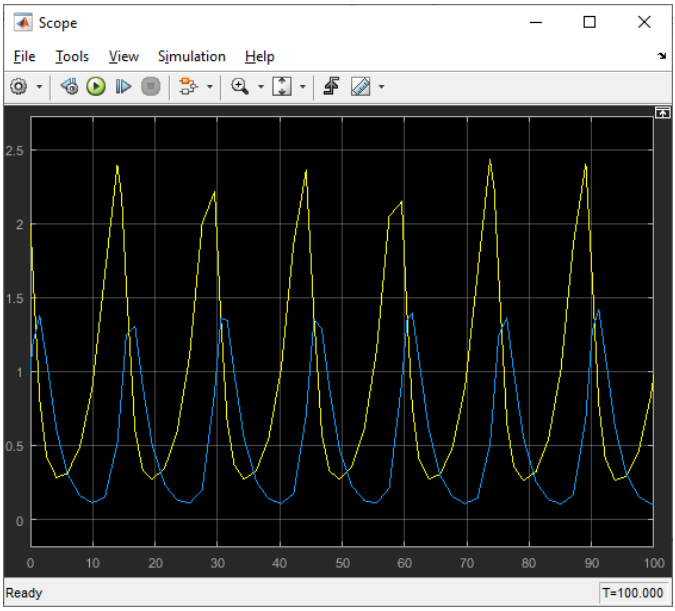


Рис. 15.3.3. Залежність числа хижаків (синя лінія) і жертв (жовта лінія) від часу.

## 15.4. Завдання для виконання

1. Ознайомитися із теоретичними відомостями.
2. Побудувати динамічну модель згідно варіанту.
3. Провести моделювання та визначити особливі точки.
4. Оформити звіт про виконання лабораторної роботи.

Варіанти моделей

1. Затухаючих коливань моделі Альфреда Д. Лотки:

$$\begin{cases} \dfrac{dx}{dt} = k_0 - k_1 xy, \\ \dfrac{dy}{dt} = k_1 xy - k_2 y, \\ \dfrac{dz}{dt} = k_2 y. \end{cases}$$

2. Незатухаючих коливань моделі Альфреда Д. Лотки:

$$\begin{cases} \dfrac{dx}{dt} = k_1 ax - k_2 xy, \\ \dfrac{dy}{dt} = k_1 xy - k_3 y. \end{cases}$$

3. Рівняння коливальних процесів хімічних реакціях моделі Лефевра:

$$\begin{cases} \dfrac{dx}{dt} = a - (b+1)x + x^2 y, \\ \dfrac{dy}{dt} = bx - x^2 y. \end{cases}$$

4. Рівняння модифікованої моделі Лотки-Вольтерри:

$$\begin{cases} \dfrac{dx}{dt} = x(\alpha - \beta y - \beta' x), \\ \dfrac{dy}{dt} = -y(\gamma - \delta x). \end{cases}$$

5. Рівняння модифікованої моделі міжвидової конкуренції:

$$\begin{cases} \dfrac{dN_1}{dt} = N_1\left(r_1 - \beta_1 N_1 - \alpha_2 N_2\right), \\ \dfrac{dN_2}{dt} = N_2\left(r_2 - \beta_2 N_2 - \alpha_1 N_1\right). \end{cases}$$

6. Рівняння Лоренца моделі конвекції:

$$\begin{cases} \dfrac{dx}{dt} = -\sigma x + \sigma y, \\ \dfrac{dy}{dt} = rx - y - xz, \\ \dfrac{dz}{dt} = -bz + xy. \end{cases}$$

## Лабораторна робота № 16
## „Дослідження порогу перколяції на квадратні гратці”

### 16.1. Теоретичні відомості

Розглянемо двовимірну квадратну гратку. Кожний вузол такої гратки може бути зайнятий з ймовірністю $p \in [0,1]$, або вільний з ймовірністю $1 - p$. Для $p$, меншої деякої визначеної ймовірності $p_c$, на гратці існує тільки скінченні області, які ми будемо називати кластерами. Кластер – множина зайняти вузлів, які пов’язані із найближчими сусідніми зайнятими вузлами найкоротшою відстанню, а величину $p_c$ називають порогом перколяції. Для $p \geq {}_c$ на необмеженій гратці існує нескінченний кластер (перколяційний кластер), що зв’язує кожну сторону гратки з протилежною стороною. Іншими словами, частина вузлів необмеженої гратки належить кластеру, в більшості випадків найбільшому, маючи при цьому нулеві значення при $p < p_c$ і значення рівні одиниці при $p \geq p_c$.

Основна задача теорії перколяції полягає у визначені величини $p_c$ на двовимірній або тривимірній гратці, як для необмеженої так і обмеженої. Виникає логічне питання, чи можна за допомогою комп’ютерного моделювання отримати оцінку порога перколяції? Нехай ми вміємо генерувати обмежену квадратну двовимірну гратку, заповнену із заданою ймовірністю $p$, і контролювати появу на ній перколяційного кластера. Щоб визначити чи лежить $p$ вище або нижче порогу перколяції, необхідно провести усереднення по багатьом таким граткам. Обчислення по всьому діапазону змін $p$ призведе до зменшення похибки у визначенні порогу перколяції при досягненні достатньої точності.

Алгоритм розв’язку задачі може бути наступним. Елементам двовимірного масиву першочергово присвоюється значення рівне нулю. У подальшому програма переглядає кожен вузол гратки (елемент масиву) генеруючи при цьому рівномірно розподілене випадкове число $r \in [0,1]$. Якщо $r$ менше початково

вибраного значення ймовірності $p$, елементу масиву присвоюється одиниця. Після перегляду всіх елементів масиву генерується одна конфігурація.

### 16.2. Приклад виконання

Описаний вище алгоритм реалізований у вигляді m-файлу `percolation` в системі MATLAB.

```
function A=percolation(p,n)
% Стохастичне моделювання конфігурації
двовимірної
% гратки розмірністю n×n із заданою
ймовірністю p
A=zeros(n,n);
for i=1:n
  for j=1:n
    r=rand(1);
    if r>p A(i,j)=1;
    end
  end
end
imagesc(A);
colormap(pink);
axis square;
for i=1:n
  for j=1:n
    A(i,j)=1-A(i,j);
  end
end
```

Приклади конфігурацій отриманих за допомогою програми `percolation` представлені на рис 16.2.1 для $p$ = 0,7 і рис 16.2.2 для $p$ = 0,2. Чорний колір вузла відповідає одиниці, а білий – нулеві.

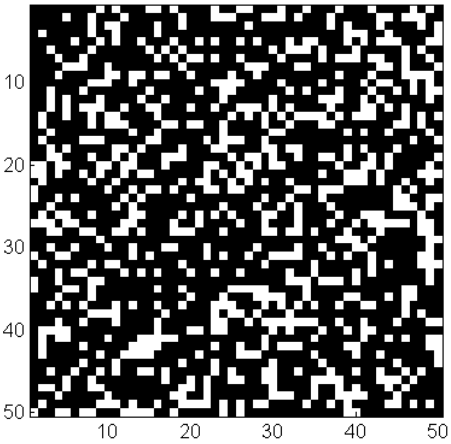


Рис. 16.2.1. Конфігурація для імовірності заняття вузла гратки $p = 0{,}7$.

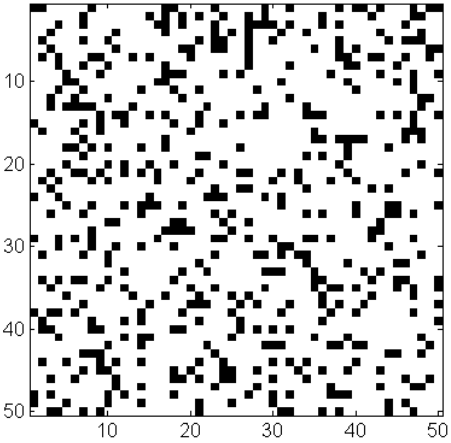


Рис. 16.2.2. Конфігурація для імовірності заняття вузла гратки $p = 0{,}2$.

Крім порогу перколяції визначають наступні характеристики, якщо $N$ – повне число вузлів:

$< N_s >$ – середнє число кластерів розміру $s$;

$n_s(p) = \frac{< N_s >}{N}$ – розподіл кластерів по розмірам;

$\varpi_s = \frac{s n_s(p)}{\sum_s s n_s(p)}$ – імовірність того, що випадково вибраний зайнятий вузол належить кластеру розміру $s$, де $s n_s(p)$ – кількість зайняти вузлів, що належать кластеру розміру $s$, $\sum_s s n_s(p)$ – повне число зайнятих вузлів;

$S = \sum_s s \varpi_s = \frac{\sum_s s^2 n_s(p)}{\sum_s s n_s(p)}$ – середній розмір кластера;

$P_\infty(p) = \frac{N_\infty}{N_{\text{зайнятих}}}$ – ймовірність того, що випадковим чином вибраний вузол буде належати перколяційному кластеру;

$< \vec{r} >= \frac{1}{s} \sum_{i=1}^{s} \vec{r}_i$ – радіус-вектор центра мас кластера;

$R_s^2 = \frac{1}{s} \sum_{i=1}^{s} (\vec{r}_i - < \vec{r} >)^2$ – радіус циркуляції;

$\xi(p)^2 = \frac{2 \sum_s < R_s^2 > s^2 n_s}{\sum_s s^2 n_s}$ – довжина кореляції.

### 16.3. Алгоритм Хошена-Копельмана

Одним із основних алгоритмів для визначання чи існує провідний кластер є алгоритм багатократного маркування Хошена-Копельмана. Цей алгоритм особливо корисний, коли досліджуються розподіл кластерів за розмірами.

Важливою особливістю алгоритму Хошена-Копельмана є його одно прохідність. За один прохід алгоритм дозволяє вияснити до якого кластеру відноситься той чи інший вузол гратки. Ідея алгоритму полягає у тому, що всім зайнятим вузлам присвоюються різні кластерні мітки.

Розглянемо роботу алгоритму на прикладі задачі вузлів на квадратній гратці. Для того щоб робота із всіма елементами масиву була однаковою, додамо до цього одну строчку з номером 0і один стовпець із номером 0, заповнений нулями. Крім того створимо масив кластерних міток `C`. Першочергово значення елементів цього масиву співпадають із їх номерами. У процесі перевірки ці номери можуть змінюватися. Якщо значення елементу масиву стало менше його номера, то таку кластерну мітку називають неправильною. Правильне значення кластерної мітки – значення елементу масиву.

Починає переглядати послідовно всі елементи масиві, починаючи із елемента `A(1,1)`. Нульовий стовпець будемо пропускати. Можливі наступні ситуації.

- Якщо значення поточного елементу масиву рівне 0, то переходимо до наступного елементу.
- Якщо поточне значення рівне 1, то здійснюємо таку перевірку:
  - Якщо сусід зліва `A(i,j-1)` та сусід зверху `A(i-1,j)` мають значення 0, то в якості якісної роботи гіпотези приймаємо, що даний елемент входить у новий кластер, присвоюємо поточному елементу номер чергової кластерної мітки. Слід відзначити, що може виявитися, що новий кластер як на здається є просто віткою якого-небудь другого кластера. Вияснити ми зможемо тільки переглянувши весь масив цілком.
  - Якщо сусід зверху має значення 0, а сусід зліва не дорівнює нулю, то поточна комірка і її сусід зліва належать одному кластеру. Поточеному елементу присвоюємо номер кластерної мітки сусіда зліва.

– Якщо сусід зверху має значення, не рівне нулю, а сусід зліва – нульове значення, то поточна комірка належить одному і тому ж кластеру. Однак, у сусіда зверху кластерна мітка може бути неправильною, так як у результаті наступних перевірок могло виявитися, що кластер, до якого належить ця комірка, злився із другим кластером. Тому поточному елементу слід присвоїти номер правильної кластерної мітки сусіда зверху.

– Якщо і сусід зверху і сусід зліва мають ненульові значення, то всі три комірки належать одному кластеру. Поточному елементу присвоюємо найменший із номерів правильних кластер них міток сусіда зверху і сусіда зліва. Корегуємо масив кластер них міток. Для цього із масиву `C`, що відповідає більшій із кластерних міток заносимо номер правильної кластерної мітки.

Описаний вище алгоритм реалізований у вигляді m-файлу `claster` в системі MATLAB.

```
function [s,C]=claster(A)
% A - Вхідна матриця
% s - Кількість кластерів
% N - Мітки провідних вертикальних кластерів
%d=0;
n=length(A)+1;
B=zeros(n,n);
C=zeros(n-1,n-1);
for i=2:n
  for j=2:n
      B(i,j)=A(i-1,j-1);
  end
end
k=1;
for i=2:n
    for j=2:n
```

```
        if B(i,j)==1 if B(i-1,j)==0 if B(i,j-
1)==0 B(i,j)=k; k=k+1;
                                        else
B(i,j)=B(i,j-1);
                                        end
                        else if B(i,j-1)==0
B(i,j)=B(i-1,j);
                                else
B(i,j)=min(B(i,j-1),B(i-1,j));
                                    s=max(B(i,j-
1),B(i-1,j));
                                    B(i,j-1)=B(i,j);
                                    B(i-1,j)=B(i,j);
                                    for ii=2:i
                                        for jj=2:n
                                            if
B(ii,jj)==s B(ii,jj)=B(i,j);
                                            end
                                        end
                                    end
                                end
                        end
        end
    end
end
k=1;
s=1;
for i=1:n-1
    for j=1:n-1
        C(i,j)=B(i+1,j+1);
        if B(i+1,j+1)>k k=B(i+1,j+1); s=s+1;
        end
        if C(i,j)~=0 D(C(i,j))=C(i,j);
        end
    end
end
for i=1:k-2
```

```
    for j=2:k-1
        if D(j)==0 D(j)=D(j+1); D(j+1)=0;
        end
    end
end
A;
%'Кількість кластерів'
s;
C;
for i=2:s
    for ii=1:n-1
        for jj=1:n-1
            if C(ii,jj)==D(i) C(ii,jj)=i;
            end
        end
    end
end
C;
k=0;
jj=1;
N(1)=0;
for i=1:n-1
    if C(1,i)~=0 & C(1,i)~=k
        k=C(1,i);
        ii=0;
        for j=1:n-1
            if k==C(n-1,j) & ii==0
                %d=1;
                'Вертикальна перколяція'
                'кластер'
                k
                ii=1;
                N(jj)=k;
                jj=jj+1;
            end
        end
    end
```

```
end
imagesc(C);
colormap(hot);
```

Приклад реалізації алгоритму представлено для квадратної гратки 25×25 для ймовірностями заняття вузла гратки $p$ = 0,25 та $p$ = 0,75 на рис. 16.3.1–16.3.5.

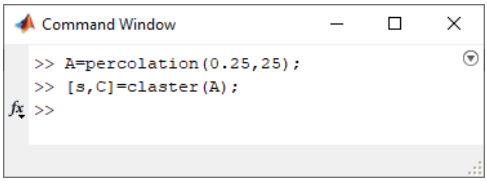


Рис. 16.3.1. Вікно Command Window системи MATLAB.

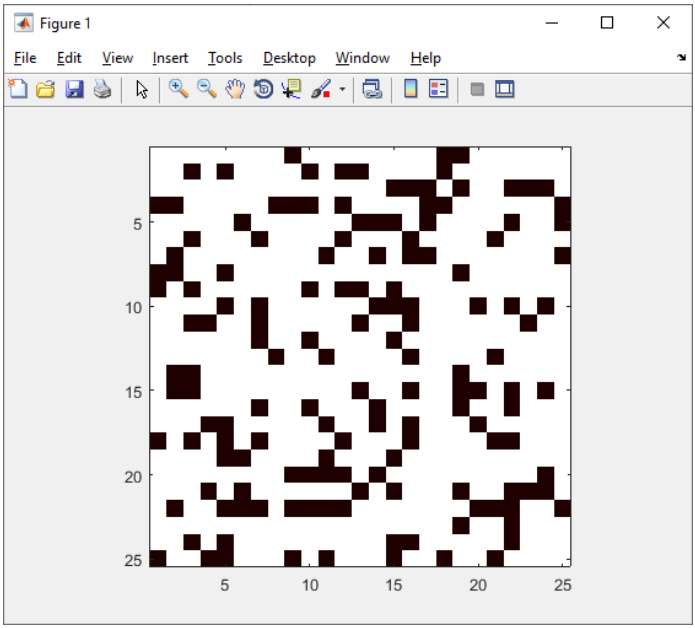


Рис. 16.3.2. Конфігурація для імовірності заняття вузла гратки $p$ = 0,25.

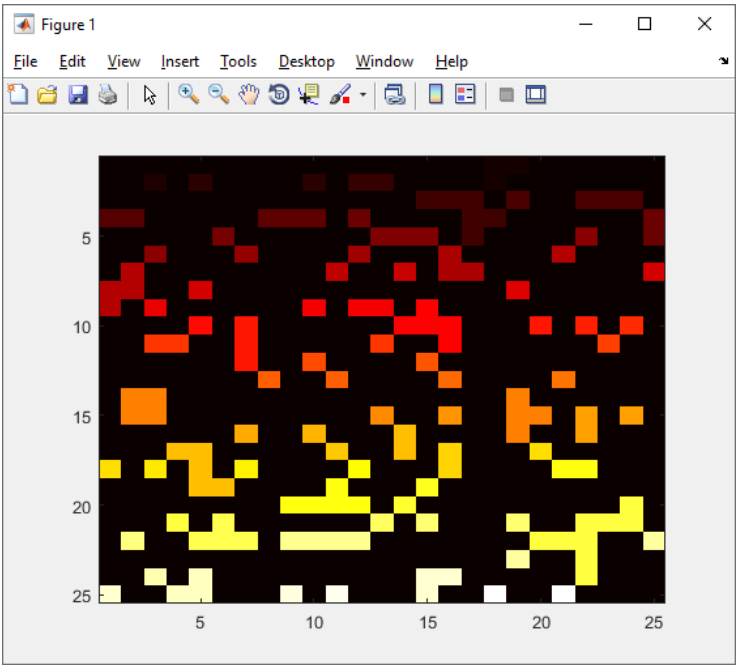


Рис. 16.3.3. Маркування кластерів за алгоритмом Хошена-Копельмана.


Рис. 16.3.4. Вікно Command Window системи MATLAB.

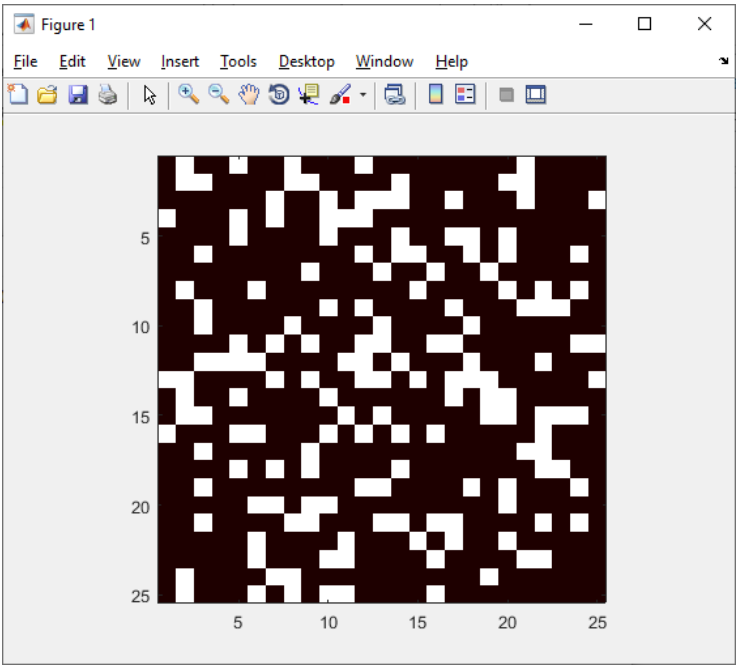


Рис. 16.3.5. Конфігурація для імовірності заняття вузла гратки $p = 0{,}75$.

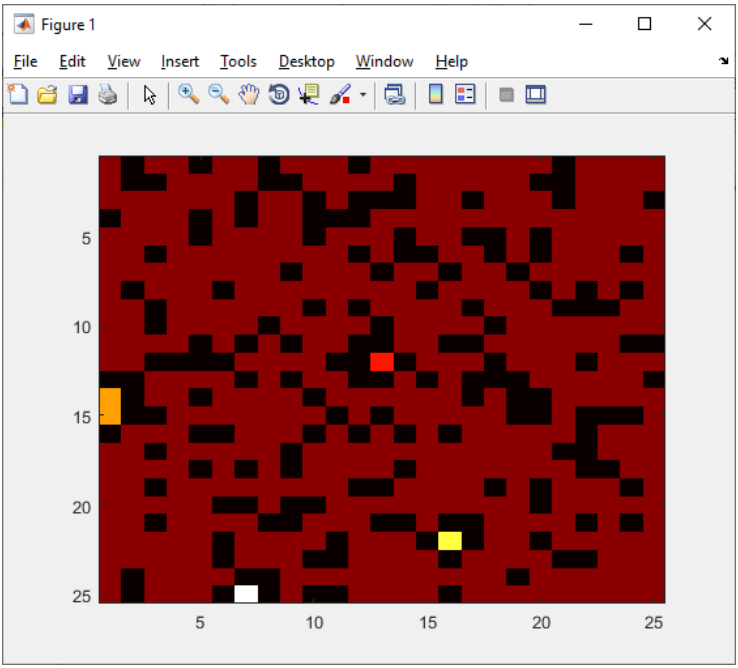


Рис. 16.3.6. Маркування кластерів за алгоритмом Хошена-Копельмана.

## 16.4. Завдання для виконання

1. Ознайомитися із теоретичними відомостями.
2. Вибрати розмірність $n$ квадратної гратки за номером у журналі $k$:

$$n = k + 10 .$$

3. Проводячи моделювання визначити при якій ймовірності $p$ на квадратній гратці розмірністю $n \times n$ буде реалізована вертикальна або горизонтальна провідність або вертикальна і горизонтальна провідності одночасно.
4. Побудувати графік залежності кількості кластерів $s$ на квадратній гратці від ймовірності $p$.
5. Оформити звіт про виконання лабораторної роботи.

## Лабораторна робота № 17
## „Побудова та дослідження часової динаміки початкових станів клітинного автомату „Життя””

### 17.1. Теоретичні відомості

Клітинні автомати є дискретними динамічними системами, поведінка яких повністю визначається в термінах локальних залежностей. Це характерно і для більшого класу неперервних динамічних систем, зокрема таких, що описуються рівняннями в частинних похідних. В цьому аспекті клітинкові автомати у інформатиці є аналогом фізичного поняття „поля” – клітинний автомат може визначатися як стилізований світ.

За своєю поведінкою клітинні автомати діляться на чотири класи. До першого класу відносяться автомати, які через деякий час приходять до стійкого однорідного стану. Автомати другого класу через деякий час після запуску генерують стаціонарні або періодичні у часі структури. У автоматах третього класу після проходження деякого часу перестає спостерігатися кореляція процесу з початковими умовами. І на кінець, поведінка автоматів четвертого класу сильно визначається початковими умовами. З їх допомогою можна генерувати досить різні шаблони поведінки. Такі автомати є кандидатами на прототип клітинної обчислювальної машини. Зокрема, за допомогою специфічних клітинних конфігурацій гри „Життя”, яка є автоматом четвертого типу, можна побудувати всі дискретні елементи цифрового комп’ютера.

Клітинний ігровий автомат „Життя” був створений Джоном Гортоном Конвеєм, математиком Кембриджського університету. Ситуації, що виникають в процесі гри дуже подібні до реальних процесів, що проходять при народженні, розвитку і загибелі колонії живих організмів. По цій причині „Життя” можна віднести до категорії ігор, які швидко розвиваються в наш час, що та імітують процеси у живій природі.

Розглянемо нескінченну плоску гратку квадратних комірок-клітинок. Час у грі будемо вважати дискретним ($t = 0, 1, 2, ...$). Клітинка може бути „живою” (зайнятою) або „мертвою” (порожньою). Зміна її стану у наступний момент часу $t + 1$ визначається станом її сусідів у момент часу $t$ (сусідів у клітинки вісім, четверо мають з нею спільні ребра, а четверо тільки вершини).

Основна ідея полягає в тому, щоб почавши з деякої конфігурації клітинок, прослідкувати за еволюцією початкової позиції під дією „генетичних законів” Конвея, які керують народженням, загибеллю і виживанням клітинок. Конвей ретельно підбирав свої правила і довго перевіряв їх на „практиці”, щоб, по можливості, вони задовольняли трьом умовам:

– не повинно бути ні однієї вихідної конфігурації, для якої існувало б просте доведення можливості необмеженого зростання популяції;
– в то же час повинні існувати такі початкові конфігурації, які свідомо володіють здатністю безмежно розвиватися;
– повинні існувати прості початкові конфігурації, які на протязі значного проміжку часу ростуть, зазнаючи змін і закінчують свою еволюцію одним із наступних трьох способів: повністю зникають (або через перенаселення, тобто досить великої густини живих клітинок, або навпаки, із-за розрізненості клітинок, що утворюють конфігурацію); переходять у стійку конфігурацію і перестають змінюватися взагалі або, виходять на коливальний режим, тобто існує нескінченний цикл перетворень із визначеним періодом.

Наслідком даних вимог виявилися наступні правила гри „Життя”:

1. **Виживання**. Кожна клітинка, що має дві або три сусідніх „живих” клітинки, виживає в переходить в наступне покоління.

2. **Загибель.** Кожна клітинка, у якої більше трьох сусідів, гине внаслідок перенаселеності. Кожна клітинка, навколо якої

вільні всі сусідні клітинки або ж зайнята всього одна клітинка, гине від самотності.

3. **Народження**. Якщо число зайнятих клітинок, з якими межує будь-яка порожня клітинка, в рівне трьом, то на цій клітинці проходить народження нового організму.

### 17.2. Приклад виконання

Для аналізу ситуацій, що виникають у грі „Життя”, використаємо розроблену програму „Життя” в системі MATLAB.

В програмі „Життя”, моделюється клітинковий автомат із квадратною матрицею розмірністю 32×32, яка і є полем для гри.

Програмно алгоритм реалізований у файлі game_life.m системи MATLAB.

Ігрове поле буде зберігатися у двовимірному масиві. Початкову конфігурацію гри можна вибрати із запропонованих стандартних або самостійно.

Після формування початкової конфігурації починається ігровий цикл. Ігровий цикл завершується, коли всі клітинки загинуть, або буде знайдена оптимальна конфігурація (однаковий стан клітинок на наступному і попередньому кроці).

### 17.3. Інтерфейс програми „Життя”

Клітинковий автомат гра „Життя” реалізований в системі MATLAB у вигляді сторінки. Сторінка представляє собою окреме вікно з відповідним інтерфейсом користувача (рис. 17.3.1).

В системі MATLAB сторінка створюється за допомогою двох типів файлів з розширенням *.fig та *.m.

У верхній частині вікна сторінки міститься строчка меню, яка складається із пунктів „Теорія” і „Допомога”.

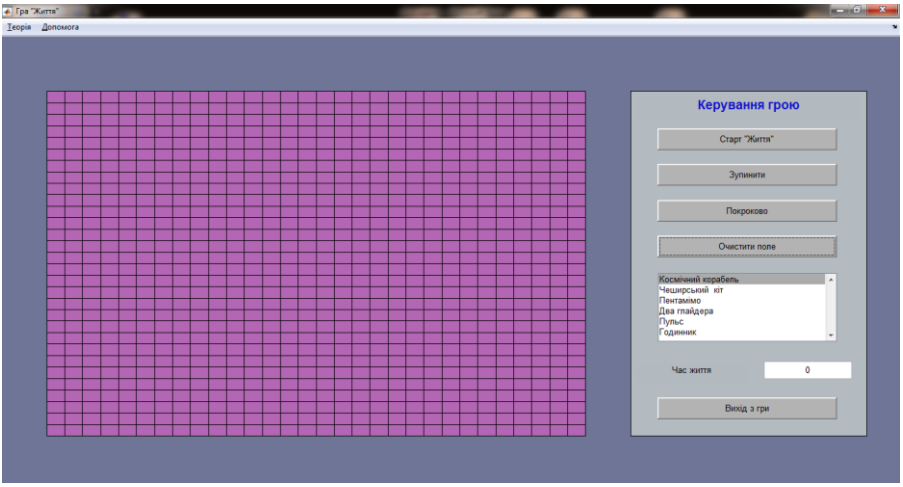


Рис. 17.3.1. Вікно програмного додатку „Життя” в системі MATLAB.

### 17.3.1. Пункт меню „Теорія”

Пункт меню „Теорія” дозволяє переглянуть теоретичний матеріал за темою: Клітинковий автомат гра „Життя”. Для перегляду потрібно клацнути лівою кнопкою мишки.

### 17.3.2. Пункт меню „Допомога”

Пункт меню „Допомога” містить дав підпункти дає можливість переглянути інформацію по роботі з програмою та інформацію про саму програму. Клацніть лівою кнопкою мишки для отриманні відповіді на Ваші запитання.

### 17.3.3. Вікно програми „Життя”

Полотно сторінки програмного додатку нами умовно розбито на дві частини. У лівій частині вікна розміщено поле, розграфлене на клітинки. На даному полі буде проходити еволюція вибраних Вами конфігурацій. Для того щоб вибрати

конфігурацію, необхідно одинарним натисканням будь-якої кнопки миші вибрати клітинки. Деякі стандартні конфігурації пропонуються нами у правій частині вікна (рис. 17.3.2).

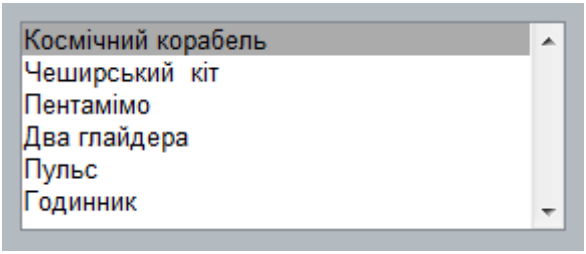


Рис. 17.3.2. Стандартні початкові конфігурації.

У правій частині вікна знаходяться елементи інтерфейсу за допомогою яких проходить керування грою. Це кнопки: „Старт „Життя"", „Зупинити", „Очистити поле", „Покроково", „Вихід з гри" (рис. 17.3.1).

Також у правій частині вікна знаходяться елементи інтерфейсу за допомогою яких проходить керування грою. Це кнопки: „Старт „Життя"", „Зупинити", „Очистити поле", „Покроково", „Вихід з гри."

**Кнопка „Старт „Життя""**. Запускає гру „Життя". Необхідно використовувати при вибраній конфігурації. Проводиться 40 часових кроків. Для необхідності продовження – ще раз натисніть на дану кнопку. При відсутності на ігровому полі початкової конфігурації програма видає наступне повідомлення (рис. 17.3.3).

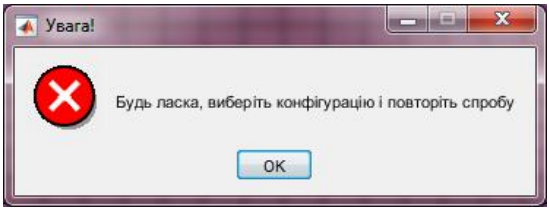


Рис. 17.3.3. Вікно „Увага!".

**Кнопка „Зупинити”**. Зупиняє гру на наступному часовому кроці. При натисканні на кнопку „Старт „Життя”” гра продовжується з того часового кроку, на якому відбулася зупинка.

**Кнопка „Покроково”**. Дозволяє досліджувати еволюцію конфігурації на кожному часовому кроці. Необхідно використовувати при вибраній конфігурації.

**Кнопка „Очистити поле”**. Оновлю ігрове поле.

**Кнопка „Вихід з гри”**. Вихід із сторінки у вікно системи MATLAB.

Крім того для зручності дослідження еволюції системи у текстовому полі виводиться „Час життя”.

### 17.3.4. Реалізація комбінацій на площині у грі „Життя”

Виходячи із вказаних алгоритмічних правил дослідимо закономірності організації структур.

Перша закономірність – властивість локалізації – структури, що розділенні двома „мертвими” клітинками не впливають одна на одну.

Друга закономірність – система клітинок, яку описує гра „Життя”, розвивається необоротно. Конфігурація в момент часу $t$ повністю визначає майбутнє (стан в моменти $t + 1$, $t + 2$ і так далі). Але відновити минуле системи за її теперішнім не вдається. Картина така ж, як у одномірних відображеннях, тільки прообразів у даній конфігурації може бути нескінченно багато.

В нашому випадку можна довести наступне твердження скориставшись властивістю локалізації. Розмістимо навколо даної конфігурації множину локалізованих одиночних клітинок або їх пар так, щоб вони не впливали на неї і одна на одну. Зрозуміло, що всі вони зникнуть на наступному кроці, ніяким чином не вплинувши на майбутнє системи.

Тут ми може відмітити ознаки нелінійних дисипативних структур: ці структури визначають поведінку системи при $t$, що прямує до нескінченності у випадку різних початкових даних.

Третя закономірність – як показують спостереження при роботі із програмою „Життя” за процесом розвитку колоній, конфігурації, які не володіють на початку гри симетрією, виявляють тенденцію до переходу у симетричні форми. Набуті властивості симетрії у процесі подальшої еволюції не втрачаються, симетрія конфігурацій може лише збагачуватися.

Початкові конфігурації клітинок будемо класифікувати за наступними параметрам:

– за кількістю клітинок у комбінації: єдина клітинка, дуплет (2 клітинки у комбінації), триплет (3 клітки) і т. д.
– за перспективою розвитку: розвиваються (необмежене зростання), стабільні (кількість кліток у популяції коливається навколо деякого середнього значення), вимираючі (популяція стабільно зменшується), періодичні (кількість клітинок приймає декілька фіксованих значень через визначений період).

Розглянемо типові структури, що з’явилися у гру „Життя”.

Найпростішими є стаціонарні, тобто таку, що не залежать від часу структури (сам Конвей називав їх „любителями спокійного життя”). Їх приклади представлені на рис. 17.3.4.

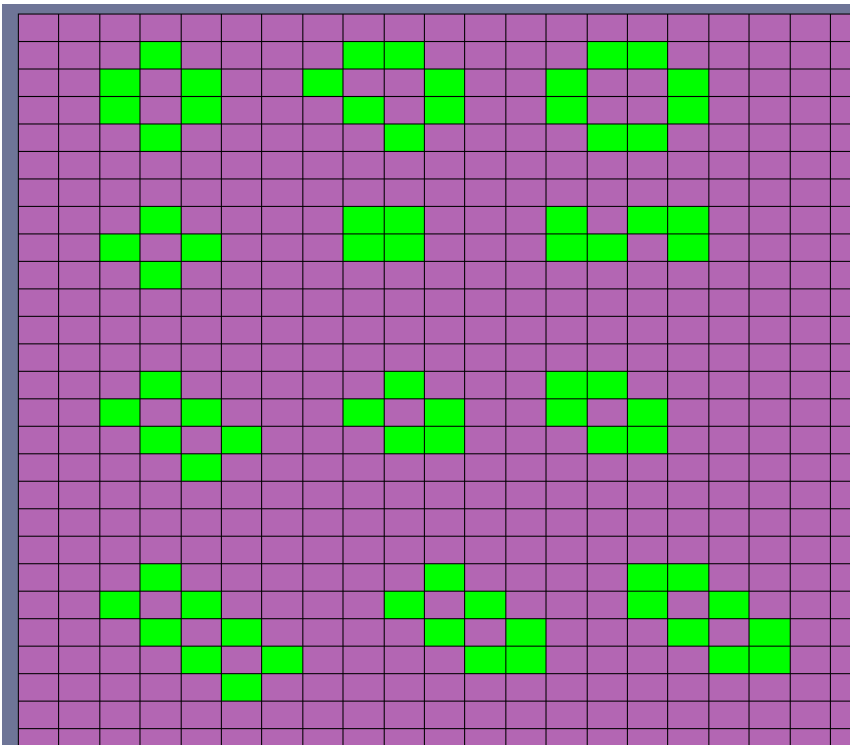

Рис. 17.3.4. Приклади стаціонарних структур.

За допомогою даних стаціонарних структур можна отримати множину інших. Справді, якщо ми маємо стаціонарну структуру, то конфігурація, отримана поворотом на 90°, також буде стаціонарною. Конфігурації у нижніх рядках показують, як можна добудовувати визначені структури до будь-яких розмірів.

Використовуючи властивість локалізації можна побудувати „великі" стаціонарні структури із „малих"-елементарних.

Можна вважати, що стаціонарні структури повторюють себе на кожному кроці у часі. Але є і інші конфігурації, що повторюють себе через $N$ кроків, так називаємі $N$-„цикли" (періодичні структури).

Приклади 2-„циклів" представлені на рис 17.3.5.

При еволюції різних співтовариств часто зустрічається 2-„цикл”, представлений у другій строчці, що називається „семафором”.

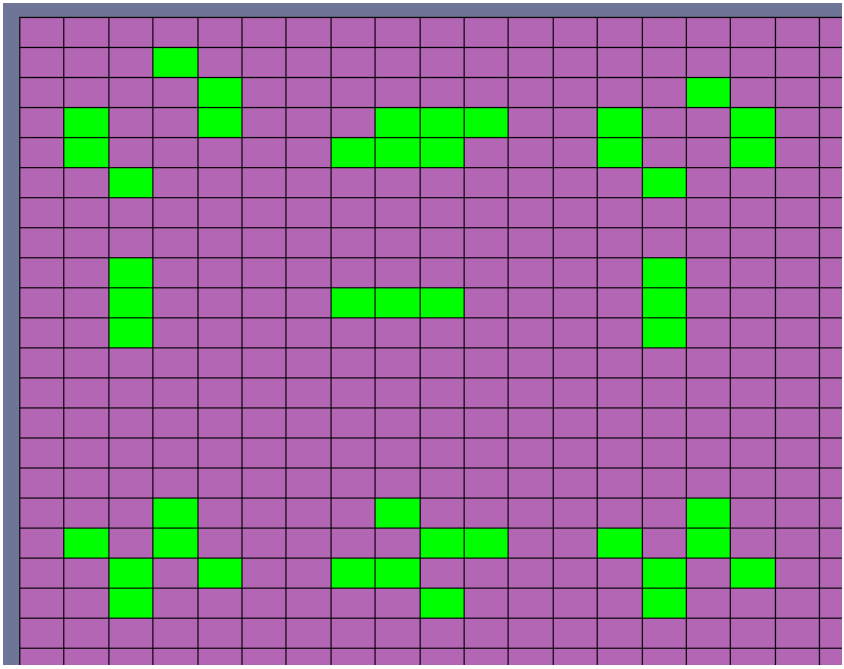

Рис. 17.3.5. Приклади періодичних структур (2-цикли), що реалізуються у грі „Життя” (жаба, семафор, годинник).

У грі „Життя” відомо багато різних періодичних конфігурацій, однак ефективні алгоритми, що дозволяють будувати різні конфігурації зі заданим періодом $N$, на сьогодні не створені.

Еволюція навмання вибраних початкових даних часто призводить до виникнення найпростіших локалізованих структур (рис. 17.3.5) і „семафорів”.

Однак можливі і більш складні типи еволюції, наприклад, коли співтовариство клітинок симетрично „добудовується”, і виникають цикли більшого періоду, що мають складну форму.

У грі „Життя” існують конфігурації, які можуть рухатися по площині. Однією із них є „планер” (або „глайдер”) – конфігурація із 5 клітинок (рис. 17.3.6)

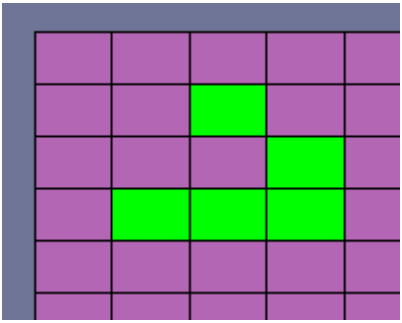

Рис 17.3.6. Конфігурація „планер” („глайдер”)

Після другого ходу „планер” трохи зсовується і відображається відносно діагоналі. В результаті двох наступних ходів „планер” „виходить із піке” і лягає на попередній курс та зсовується на одну клітку вправо і одну клітинку вниз відносно початкової позиції.

Швидкість, з якою „шахматний король” переміщується по дошці у любому напрямку, Конвей називає „швидкістю світла”. Вибір Конвея впав власне на цей термін із-за того, що у придуманій ним грі більші швидкості просто не є досяжними. Ні одна конфігурація не відтворює себе достатньо швидко, щоб рухатися з такою швидкістю.

Доведено, що максимальна швидкість на діагоналі складає одну четверту „швидкості світла”. Оскільки „планер” переходить сам у себе після чотирьох ходів и при цьому опускається на одну клітку по діагоналі, то говорять, що він ковзає по полю із швидкістю, рівною одній четвертій „швидкості світла”.

Конвей також показав, що швидкість будь-якої скінченої фігури, що переміщується по вертикалі або по горизонталі на клітинки, не може перевищувати половину „швидкості світа”. При цьому швидкість руху фігури визначається наступним чином:

$$\upsilon = \frac{k}{m}, \qquad (17.3.1)$$

де $k$ – число ходів, необхідних для відновлення фігури, $m$ – кількість клітинок на яку вона при цьому переміщується.

Зрозуміло, що в силу симетрії є „планери”, які поширюються вдовж будь-якої діагоналі квадрата у обох напрямках.

Окремо відмітимо, що деякі конфігурації можуть переміщуватися не вздовж діагоналей, а по вертикальним і горизонтальним прямим. Такі, наприклад, як три „кораблі” (рис. 17.3.7). Відзначимо, що далеко не будь-яка конфігурація такого типу буде „кораблем”. До речі, „планер” є теж кораблём „найлегшої ваги”. Під час руху кораблів виникають „іскри”, які тут же загасають при подальшому русі „корабля”.

Одиночні „кораблі” без ескорту не можуть бути розміщенні у довжину більше шести клітинок, в іншому випадку на полі починають з’являтися різні мілкі фігури, що перешкоджають руху „корабля”. Конвей виявив, що більш довгим кораблям (які він називав „надважкими”) необхідний ескорт із двох або більшого числа „кораблів” менших розмірів. „Кораблі” ескорту не дають можливості виникати перешкодам на шляху „надважкого” „корабля”.

Конвей обчислив, що „корабель” довжиною у 100 клітинок потребує ескорту, що складається із 33 кораблів.

Цікавими з точки зору конфігурацій є випадки, коли відбуваються зіткнення між рухомими фігурами, або зіткнення рухомих фігур з різними стаціонарними конфігураціями („стаціонари”). Зіткнення можуть бути досить різні в залежності від курсу „планера” і його фази при зіткненні.

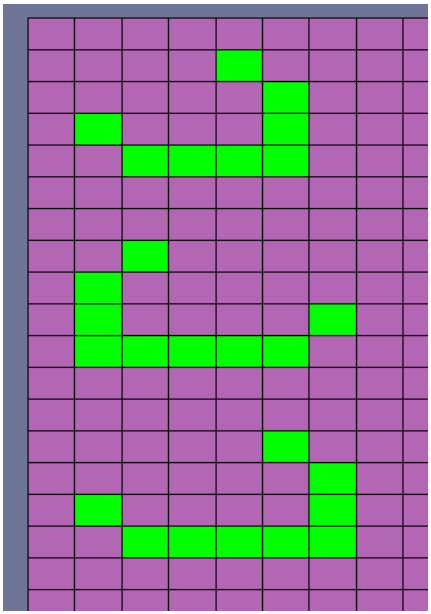

Рис. 17.3.7. „Кораблі” – конфігурації, що реалізуються у грі „Життя”, що здатні переміщуватися.

Зіткнення двох „планерів” або „планера” із „стаціонаром” може приводить до їх „анігіляції”. В зіткненні може народжуватися цілий набір „семафорів” і „стаціонарів”.

Звернемо увагу на наступну закономірність. Якщо конфігурація весь час локалізована у квадрату розміром $N \times N$, то вона є набором „стаціонарів” і „циклів”, період яких не перевищує $2N$. Справді, кожна клітинка може знаходиться в одному із двох станів, а в області всього $N^2$ клітинок, тому при

$$t > 2^N, \qquad (17.3.2)$$

конфігурації почнуть повторюватися.

Розглядаючи неперервні середовища, можна говорити про резонансні збудження – початкових даних (початкових конфігурацій), що призводять до більш складної еволюції розв’язків, ніж у інших випадках.

У грі „Життя” є аналог такої поведінки. Звернемо увагу на конфігурацію представлену на рис. 17.3.8, що називається „*r*-пентаміно”.

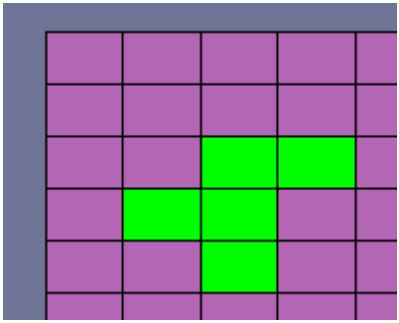

Рис. 17.3.8. Конфігурація „*r*-пентаміно”.

Клітинки, що виникаю займають все більшу частину площини, народжуючи декілька „планерів”, причому ця спільнота буде розвиватися і далі. Ні одна із конфігурацій, що складається із п’яти клітинок, не призводить до такої складної поведінки.

Як правило, еволюція вибраних навмання конфігурацій призводить до появи наборів „стаціонарів”, „семафорів”, „планерів”. При цьому загальна кількість клітинок при $t \to ($, прямуючому до нескінченності, виявляється обмеженим (це було закладено у вимогах Конвея), однак, при деяких початкових даних еволюція системи може якісно змінюватися. Така поведінка характерна для ряду біологічних систем, зокрема, еволюційних процесів.

Малоймовірна подія може якісно змінити поведінку системи, що призводить до появи нових видів.

Власне тому гра „Життя” знаходить використання у екологічних моделях, при моделюванні морфогенеза, та інших біологічних задачах.

Чим більшу площу займає спільнота клітинок, тим складніше вона може себе проявити. Тому великий інтерес

викликають необмежено зростаючі у просторі конфігурації. Одну із них, що називається „катапульта” або „планерна рушниця.

Видно, що „катапульта” через кожні 30 кроків повторює себе і випускає „планер” (рис. 17.3.9). „Планерна рушниця” заповнює простір потоком „планерів”. Існує гіпотеза, згідно якої не існує ні одної початкової конфігурації, здатної безмежно зростати. Іншими словами, будь-яка конфігурація, що складається із скінченого числа „живих” клітинок, не може перейти у конфігурацію, в якій число „живих” клітинок перевершувало б деяку скінчену границю. Але гіпотеза виявилася помилковою. Спростування – „планера рушниця”.

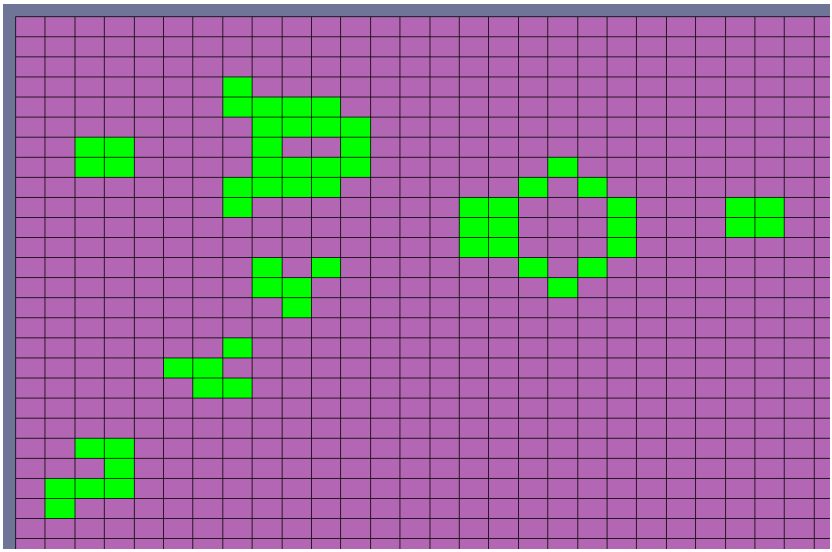

Рис. 17.3.9. „Планерна рушниця” („катапульта”) – конфігурація, що генерує за кожні 30 крокі „планер”.

У грі „Життя” існують ще більш складні спільноти клітинок, які можуть рухатися, створюючи за собою великий набір „семафорів” і „стаціонарів”. Одна із них – „паротяг” (він має

доволі складну структуру). Пошук таких конфігурацій – досить складний процес, що потребує використання спеціальних алгоритмів і під силу кваліфікованим спеціалістам.

Також були знайдені особливі конфігурації, які Джон Тьюкі назвав „садами Едему”. „Сади Едему” не можуть виникати у ході роботи клітинкового автомату, оскільки їх не здатна генерувати ніяка конфігурація.

### 17.4. Завдання для виконання

1. Ознайомитися із теоретичними відомостями.
2. Освоїти інтерфейс програми „Життя”.
3. Провести моделювання обраних конфігурацій у програмі „Життя”.
4. Спробувати запропонувати свою конфігурацію до одного із типів.
5. Оформити звіт про виконання лабораторної роботи.

# Лабораторна робота № 18
# „Використання систем нечіткого виводу в задачах керування”

## 18.1. Теоретичні відомості

### 18.1.1. Базова архітектура систем нечіткого виводу

Математична теорія нечітких множин дозволяє описувати нечіткі поняття і знання, оперувати ними і робити нечіткі висновки.

При описі об’єктів і процесів за допомогою нечітких множин використовується поняття нечіткої і лінгвістичної змінних.

Нечітка змінна характеризується трійкою

$$< a, X, A > \qquad (18.1.1)$$

де $a$ – ім’я змінної, $X$ – універсальна множина (область визначення $a$), $A$ – нечітка множина на $X$, що описує обмеження (тобто $\mu_A(x)$) на значення нечіткої змінної $a$.

Лінгвістичною змінною називається набір

$$< b, T, X, G, M > \qquad (18.1.2)$$

де $b$ – ім’я лінгвістичної змінної; $T$ – множина її значень (терм-множина), що представляють собою імена нечітких змінних, областю визначення, кожної з яких є множина $X$. Множина $T$ називається базовою терм-множиною лінгвістичної змінної; $G$ – синтаксична процедура, що дозволяє оперувати елементами терм-множини $T$, зокрема, генерувати нові терми (значення); $M$ – семантична процедура, що дозволяє перетворити кожне нове значення лінгвістичної змінної, утвореною процедурою $G$, у нечітку змінну, тобто сформувати відповідну нечітку множину.

Щоб уникнути великої кількості символів символ $b$ використовують як для назви самої змінної, так і для всіх її значень; для позначення нечіткої множини і його назви користуються одним символом.

Нечіткими висловленнями будемо називати висловлення наступного виду:

1. Висловлювання $<b \in b'>$, де $b$ – ім'я лінгвістичної змінної, $b'$ – її значення, якому відповідає нечітка множина на універсальній множині $X$.

2. Висловлювання $<b \in \mu_{b'}>$, де $\mu$ – модифікатор, якому відповідають слова „ДУЖЕ", „БІЛЬШ-МЕНШ", „НАБАГАТО БІЛЬШЕ" і ін.

3. Складні висловлювання, утворені з висловлень виду 1 і 2 та сполучників „І", „АБО", „ЯКЩО ..., ТОДІ ...", „ЯКЩО ..., ТОДІ ..., ІНАКШЕ".

Якщо значення фіксованої лінгвістичної змінної відповідають нечітким множинам однієї універсальної множини $X$, то можна ототожнювати модифікатори „ДУЖЕ" чи „НЕ" з операціями „CON" і „доповнення", а сполучники „І", „АБО" з операціями „перетин" і „об'єднання" над нечіткими множинами.

Найбільш важливим застосуванням теорії нечітких множин є контролери нечіткої логіки. Їх функціонування дещо відрізняється від роботи звичайних контролерів; для опису системи замість диференційних рівнянь використовуються знання експертів. Ці знання можуть бути виражені за допомогою лінгвістичних змінних, які описані нечіткими множинами.

Загальна структура мікроконтролера, що використовує нечітку логіку, показана на рис. 18.1.1. Вона містить у своєму складі наступні складові: блок фазифікації, базу знань, блок рішень, блок дефазифікації.

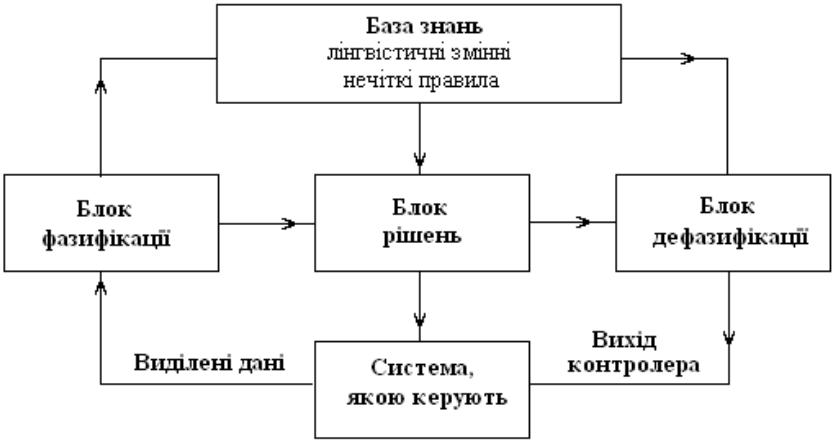


Рис. 18.1.1. Загальна структура нечіткого мікро контролера.

Блок фазіфікації перетворює чіткі величини, виміряні на виході об'єкта керування, у нечіткі величини, що описані лінгвістичними змінними в базі знань.

Блок рішень використовує нечіткі умовні ( if – then ) правила, закладені в базі знань, для перетворення нечітких вхідних даних у необхідні керуючі впливи, що носять також нечіткий характер.

Блок дефазіфікації перетворює нечіткі дані з виходу блоку рішень у чітку величину, що використовується для керування об'єктом.

Всі системи з нечіткою логікою функціонують за одним принципом: показання вимірювальних приладів: фазіфікуються (перетворюються в нечіткий формат), обробляються, дефазіфікуються й у вигляді звичайних сигналів подаються на виконавчі пристрої.

### 18.1.2. Основні етапи нечіткого виводу

Розробка і застосування систем нечіткого виводу включає декілька етапів, реалізація яких виконується за допомогою розглянутих раніше основних положень нечіткої логіки.

Інформація, яка поступає на вхід системи нечіткого виводу, вимірюється деяким чином за допомогою вхідних змінних, що відповідають реальним змінним процесу управління. Інформація, яка формується на виході системи нечіткого висновку, відповідає вихідним змінним, що є управляючими змінними процесу управління.

Системи нечіткого виводу призначені для перетворення значень вхідних змінних процесу управління у вихідні змінні на основі використовування нечітких правил. Для цього такі системи повинні містити базу правил і реалізовувати нечітке виведення висновків на основі посилань або умов, представлених у формі нечітких лінгвістичних висловів.

Розглянемо основні особливості кожного з вказаних етапів.

База правил систем нечіткого виводу призначена для формального представлення емпіричних знань або знань експертів в тій або іншій проблемній області. В системах нечіткого виводу використовуються правила нечітких продукцій, в яких умови і висновки сформульовані в термінах нечітких лінгвістичних висловлювань. Сукупність таких правил далі називатимемо базами правил нечітких продукцій


Рис. 18.1.2. Діаграма процесу нечіткого виводу

База правил нечітких продукцій є кінцевою множиною правил нечітких продукцій, узгоджених відносно використовуваних лінгвістичних змінних. Найбільш часто база правил представляється у формі структурованого тексту:

ПРАВИЛО 1: ЯКЩО „Умова, 1" то „Висновок 1" ($F_1$)

ПРАВИЛО 2: ЯКЩО „Умова 2" то „Висновок 2" ($F_2$)

... ... ...

ПРАВИЛО *n*: ЯКЩО „Умова n" то „Висновок n" ($F_n$)

Через $F_n$ позначені коефіцієнти визначеності або вагові коефіцієнти відповідних правил. Ці коефіцієнти можуть приймати значення з інтервалу $[0,1]$. У випадку, якщо ці вагові коефіцієнти відсутні, зручно прийняти їх значення рівні 1.

Узгодженість правил щодо лінгвістичних змінних означає, що в якості умов і висновків використовуються нечіткі лінгвістичні висловлювання. При цьому в кожному з нечітких висловів повинні бути визначений функції належності значень терм-множини для кожної з лінгвістичних змінних.

В системах нечіткого виводу лінгвістичні змінні, які використовуються в нечітких висловлюваннях підумов правил нечітких продукцій, часто називають вхідними лінгвістичними змінними, а змінні, які використовуються в нечітких висловлюваннях підвисновків правил нечітких продукцій, називають вихідними лінгвістичними змінними.

Таким чином, під час формування бази правил нечітких продукцій необхідно визначити:

– множину правил нечітких продукцій;
– множину вхідних лінгвістичних змінних;
– множину вихідних лінгвістичних змінних.

База правил нечітких продукцій вважається заданою, якщо задані названі множини. Нагадаємо, що вхідна вихідна або вихідна лінгвістична змінна вважається заданою або визначеною, якщо для неї визначена базова терм-множина з відповідними функціями належності кожного терму, а також дві процедури *G* і *M*. Найбільш поширеним випадком є використання функцій належності термів трикутних або у вигляді трапецій. При цьому для зручності записів застосовують спеціальні скорочення для найменування окремих термів вхідних і вихідних лінгвістичних змінних (табл. 18.1.1).

Таблиця 18.1.1.

Загальноприйняті скорочення для значень основних лінгвістичних змінних в системах нечіткого виводу

| Символічне позначення | Англомовна нотація | Українськомовна нотація |
|---|---|---|
| NB | Negative Big | Негативне велике |
| NM | Negative Middle | Негативне середнє |
| NS | Negative Small | Негативне мале |
| ZN | Zero Negative | Негативне близьке до нуля |
| Z | Zero | Нуль, близьке до нуля |
| ZP | Zero Positive | Позитивне близьке до нуля |
| PS | Positive Small | Позитивне мале |
| PM | Positive Middle | Позитивне середнє |
| PB | Positive Big | Позитивне велике |

На формування бази правил систем нечіткого виводу часто впливають деякі додаткові чинники, які визначаються специфікою поставленої задачі або алгоритму нечіткого виводу.

### 18.1.2.1. Фазифікація

В контексті нечіткої логіки під фазифікацією розуміємо не тільки окремий етап виконання нечіткого виводу, але і власне процес або процедура знаходження значень функцій належності нечітких множин (термів) на основі звичайних (не нечітких) початкових даних. Фазифікацію ще називають веденням нечіткості.

Метою етапу фазифікації є встановлення відповідності між конкретним (зазвичай чисельним) значенням окремої вхідної змінної системи чіткого виводу і значенням функції належності відповідною їй терма вхідної лінгвістичної змінної. Після завершення цього етапу для всіх вхідних змінних повинні бути визначені конкретні значення функцій належності по кожному з

лінгвістичних термів, які використовуються в підумовах бази правил системи нечіткого виводу.

Формально процедура фазифікації виконується таким чином. До початку цього етапу стають відомими конкретні значення всіх вхідних змінних системи нечіткого виводу, тобто множина значень $V = \{a_1, a_2, ..., a_n\}$. Ці значення можуть бути отримані або від датчиків, або деяким іншим, зовнішнім по відношенню до системи нечіткого виводу способом.

Далі розглядається кожна з підумов виду правил системи нечіткого виводу. При цьому значення $a_i$, використовується як аргумент $\mu(x)$, тим самим знаходиться кількісне значення $\beta_i = \mu(a_i)$. Це значення і є результатом фазифікації підумови бази правил системи нечіткого виводу.

Етап фазифікації вважається закінченим, коли будуть знайдені всі значення $\beta_i = \mu(a_i)$ для кожної з підумов всіх правил системи нечіткого виводу. Цю множину значень позначають через $B = \{\beta_1, \beta_2, ..., \beta_n\}$. При цьому якщо деякий терм $\alpha''$ лінгвістичної змінної $\beta_i$ не присутній ні в одному з нечітких виводів, то відповідне йому значення функції належності не визначається в процесі газифікації

### 18.1.2.2. Агрегація

Агрегація є процедурою визначення ступеня істинності умов за кожним з правил системи нечіткого виводу.

Формально процедура агрегації виконується таким чином. До початку цього етапу є вже відомими значення істинності всіх підумов системи нечіткого виводу, тобто множина значень $B = \{\beta_1, \beta_2, ... \beta_n\}$. Далі розглядається кожна з умов правил системи нечіткого виводу. Якщо умова правила є нечітким висловлюванням, то ступінь його істинності рівний відповідному значенню $\beta_i$.

Якщо ж умова складається з декількох підумов, причому лінгвістичні змінні в підумовах попарно не рівні один одному, то визначається ступінь істинності складного вислову на основі відомих значень інтенсивності підумов. При цьому для

визначення результату нечіткої кон'юнкції або зв'язки „І" та диз'юнкції або зв'язки „АБО" може бути використана одна з відповідних формул. При цьому значення $\beta_i$ використовуються як аргументи відповідних логічних операцій. Таким чином знаходятся кількісні значення істинності всіх умов правил системи нечіткого виводу.

Етап агрегації вважається закінченим, коли будуть знайдені всі значення $\beta_k''$ для кожного з правил $R_k$, що входить в базу правил системи нечіткого виводу. Цю множину значень позначимо через $B'' = \{\beta_1'', \beta_2'', ..., \beta_k''\}$.

### 18.1.2.3. Активізація

Активізація в системах нечіткого виводу є процедурою або процесом знаходження ступеня істинності кожного з підвисновків правил нечітких продукцій. Активізація в загальному випадку багато в чому аналогічна композиції нечітких відношень, але не тотожна їй.

При формуванні бази правил системи нечіткого виводу задаються вагові коефіцієнти $F_n$ для кожного правила (якщо ваговий коефіцієнт не заданий явно, то його значення рівне 1).

Формально процедура активізації виконується таким чином. До початку цього етапу є відомими значення істинності всіх умов системи нечіткого виводу, тобто множина значень $B'' = \{\beta_1'', \beta_2'', ..., \beta_k''\}$ і значення вагових коефіцієнтів $F_n$ для кожного правила. Далі розглядається кожний з висновків правил системи нечіткого виводу. Якщо висновок правила є нечітким висловлюванням, то ступінь його істинності рівний добутку відповідного значення $\beta_i''$ на ваговий коефіцієнт $F_i$.

Якщо ж висновок складається з декількох підвисновків, причому лінгвістичні змінні в підвисновках попарно не рівні один одному, то ступінь істинності кожного з них рівний добутку відповідного значення $\beta_i''$ на ваговий коефіцієнт $F_i$. Таким чином, знаходяться всі значення $c_k$ ступенів істинності підвисновків для кожного з правил $R_k$, що входить в

розглядувану базу правил системи нечіткого виводу. Цю множину значень позначимо через $C = \{c_1, c_2, ..., c_q\}$, де $q$ – загальна кількість підвисновків в базі правил.

Після знаходження множини $C = \{c_1, c_2, ..., c_q\}$ визначаються функції належності кожного з підвисновків для вихідних лінгвістичних змінних. Для цієї цілі можна використовувати один з методів, що є модифікацією того або іншого методу нечіткої композиції:

1. min-активізація: $\mu'(y) = min\{c_i, \mu(y)\}$;
2. prod-активізація: $\mu'(y) = c_i\mu(y)$;
3. average-активізація: $\mu'(y) = 0{,}5(c_i + \mu(y))$,

де $\mu(y)$ – функція належності терма, який є значенням деякої вихідної змінної $\omega_j$, заданій на універсумі $Y$.

Етап активізації вважається закінченим, коли для кожної з вихідних лінгвістичних змінних, що входить в окремі підвисновки правил нечіткого виводу будуть визначені функції належності нечітких множин їх значень.

### 18.1.2.4. Акумуляція

Акумуляція в системах нечіткого виводу є процедурою або процесом знаходження функції належності для кожної з вихідних лінгвістичних змінних.

Мета акумуляції полягає в тому, щоб об’єднати або акумулювати всі ступені істинності висновків (підвисновків) для отримання функції належності кожної з вихідних змінних. Причина необхідності виконання цього етапу полягає в тому, що підвисновки, що відносяться до однієї і тієї ж вихідної лінгвістичної змінної, належать різним правилам системи нечіткого виводу.

Формально процедура акумуляції виконується таким чином. До початку цього етапу мають бути відомими значення істинності всіх підвисновків для кожного з правил $R_k$, що входить в базу правил системи нечіткого виводу у формі

сукупності нечітких множин: $C_1, C_2, ..., C_q$, де $q$ – загальна кількість підвисновків в базі правил. Далі послідовно розглядається кожна з вихідних лінгвістичних змінних $\omega_j \in W$ і нечіткі множини, що відносяться до них: $C_{j1}, C_{j2}, ..., C_{jq}$. Результат акумуляції для вихідної лінгвістичної змінної $\omega_j$, визначається як об'єднання нечітких множин $C_{j1}, C_{j2}, ..., C_{jq}$ за однією з відповідних формул.

Етап акумуляції вважається закінченим, коли для кожної з вихідних лінгвістичних змінних будуть визначені підсумкові функції належності нечіткої множини їх значень, тобто сукупність нечітких множин: $C'_{j1}, C'_{j2}, ..., C'_{js}$, де $s$ – загальна кількість вихідних лінгвістичних змінних в базі правил системи нечіткого виводу.

### 18.1.2.5. Дефазифікація

Дефазифіація в системах нечіткого виводу є процедурою або процесом знаходження звичайного (не нечіткого) значення для кожної з вихідних лінгвістичних змінних множини $W = \{\omega_1, \omega_2, ..., \omega_n\}$.

Мета дефазифікації полягає в тому, щоб, використовуючи результати акумуляції всіх вихідних лінгвістичних змінних, отримати звичайне кількісне значення (crisp value) кожній з вихідних змінних, яке може бути використане спеціальними пристроями, зовнішніми по відношенню до системи нечіткого виводу.

Використовувані в сучасних системах управління пристрої і механізми здатні сприймати традиційні команди у формі кількісних значень відповідних керуючих змінних. Саме з цієї причини необхідно перетворити нечіткі множини в деякі конкретні значення змінних. Тому дефазифікацію називають також приведенням до чіткості.

Формально процедура дефазифікації виконується таким чином. До початку цього етапу є відомими функції належності всіх вихідних лінгвістичних змінних у формі нечітких множин:

$C'_{j1}, C'_{j2}, ..., C'_{js}$, де $s$ – загальна кількість вихідних лінгвістичних змінних в базі правил системи нечіткого виводу. Далі послідовно розглядається кожна з вихідних лінгвістичних змінних $\omega_j \in W$ і нечітка множина $C'_j$, що відноситься до неї. Результат дефазифікації для вихідної лінгвістичної змінної визначається у вигляді кількісного значення $y_j \in R$, отриманого за однією з нижче наведених формул.

Етап дефазифікації вважається закінченим, коли для кожної з вихідних лінгвістичних змінних будуть визначені підсумкові кількісні значення у формі деякого дійсного числа, тобто у вигляді $y_1, y_2, ..., y_s$, де s – загальна кількість вихідних лінгвістичних змінних в базі правил системи нечіткого висновку.

Для виконання чисельних розрахунків на етапі дефазифікації можуть бути використані наступні формули, які отримали назву методів дефазифікації.

1. Метод центра тяжіння:

$$y = \frac{\int_{Min}^{Max} x\mu(x)dx}{\int_{Min}^{Max} \mu(x)dx}; \qquad (18.1.1)$$

2. Метод центра тяжіння для одноточкових множин:

$$y = \frac{\sum_{i=1}^{n} x_i \mu(x_i)}{\sum_{i=1}^{n} \mu(x_i)}; \qquad (18.1.2)$$

3. Метод центра площі $y = u$:

$$\int_{Min}^{u} \mu(x)dx = \int_{u}^{Max} \mu(x)dx; \qquad (18.1.3)$$

4. Метод лівого модального значення:

$$y = \min\{x_m\}; \qquad (18.1.4)$$

5. Метод правого модального значення:

$$y = \max\{x_m\}. \qquad (18.1.5)$$

### 18.1.3. Алгоритм Мамдані

Алгоритм Мамдані є одним з перших, який знайшов застосування в системах нечіткого виводу. Він був запропонований в 1975 р. англійським математиком Е. Мамдані (Ebrahim Mamdani) як метод для управління паровым двигуном. По своїй суті цей алгоритм породжує розглянуті вище етапи, оскільки найбільшою мірою відповідає їх параметрам.

Формально алгоритм Мамдані може бути визначений таким чином.

Формування бази правил систем нечіткого виводу. Особливості формування бази правил співпадають з розглянутими вище при описі даного етапу.

Фазифікация вхідних змінних. Особливості фазифікації співпадають з розглянутими вище під час опису даного етапу.

Агрегація підумов в нечітких правилах продукцій. Для знаходження ступеня істинності умов кожного з правил нечітких продукцій використовуються парні нечіткі логічні операції. Ті правила, ступінь істинності умов яких відмінний від нуля, вважаються активними і використовуються для подальших розрахунків.

Активізація підвисновків в нечітких правилах продукцій. Здійснюється за формулою:

$$\mu'(y) = min\{c_i, \mu(y)\}, \qquad (18.1.6)$$

при цьому для скорочення часу виведення Враховуються тільки активні правила нечітких продукцій.

Акумуляція висновків нечітких правил продукцій здійснюється за формулою:

$$\mu'_D(\mathrm{x}) = max\{\mu_A(x), \mu_B(x)\}, \qquad (18.1.7)$$

для об’єднання нечітких множин відповідних терм підвисновків, що відносяться до одних і тих самих вихідних лінгвістичних змінних.

Дефазифікация вихідних змінних. Традиційно використовується метод центру тяжіння у формі або метод центру площ.

Розглянемо приклад. Будемо рахувати, що базу знань створюють два нечітких правила виду:

$П_1$: якщо $x$ є $A_1$ і $y$ є $B_1$, тоді $z$ є $C_1$,

$П_2$: якщо $x$ є $A_2$ і y є $B_2$, тоді $z$ є $C_2$,

де $x, y$ – імена вхідних змінних, $z$ – ім'я змінної виводу; $A_1, A_2, B_1, B_2, C_1, C_2$ – деякі задані функції належності. При цьому чітке значення $z_0$ необхідно визначити на основі наведеної інформації і чітких значень $x_0$ і $y_0$.

Алгоритм Мамдані в цій ситуації математично може бути описаний таким чином.

1. Введення нечіткості. Знаходяться ступені істинності для передумов кожного правила

$$A_1(x_0), A_2(x_0), B_1(y_0), B_2(y_0).$$

2. Логічний висновок. Знаходяться рівні «відсікання» для передумов кожного з правил (з використанням операції МІНІМУМ)

$$\alpha_1 = A_1(x_0) \wedge B_1(y_0),$$
$$\alpha_2 = A_2(x_0) \wedge B_2(y_0),$$

де „$\wedge$" позначена операція логічного мінімуму (*min*). Потім знаходяться „усічені" функції належності

$$C_1' = (\alpha_1 \wedge C_1(z)),$$
$$C_2' = (\alpha_2 \wedge C_2(z)).$$

3. Композиція. Проводиться об'єднання знайдених усічених функцій з використанням операції МАКСИМУМ (*max*, позначене далі як „$\vee$"), що приводить до отримання підсумкової нечіткої підмножини для зміни виходу з функцією належності:

$$\mu_\Sigma(z) = C(z) = C_1(z) \vee C_2(z) = (\alpha_1 \wedge C_1(z)) \vee (\alpha_2 \wedge C_2(z)).$$

4. Приведення до чіткості. Проводиться для знаходження $z_0$, наприклад, центроїдним методом.

В моделі типу Мамдані взаємозв'язок між входами $x = (x_1, x_2, ..., x_n)$ і виходом у визначається нечіткою базою знань наступного формату:

| | |
|---|---|
| ЯКЩО | $\left(x_1 = a_{1,j_1}\right) i \left(x_2 = a_{2,j_1}\right) i ..., i \left(x_n = a_{n,j_1}\right)$ з вагою $\omega_{j_1}$ |
| АБО | $\left(x_1 = a_{1,j_2}\right) i \left(x_2 = a_{2,j_2}\right) i ..., i \left(x_n = a_{n,j_2}\right)$ з вагою $\omega_{j_2}$ |
| … | |
| АБО | $\left(x_1 = a_{1,j_k}\right) i \left(x_2 = a_{2,j_k}\right) i ..., i \left(x_n = a_{n,j_k}\right)$ з вагою $\omega_{j_k}$ |
| ТО | $y = d_j$, $j = \overline{1,m}$ |

де $a_{i,j_p}$ – лінгвістичний терм, яким оцінюється змінна $x_i$ в строчці с номером $j_p$ $\left(p = \overline{1,k_j}\right)$; $k_j$ – кількість строчок-кон'юнкцій, у яких вихід у оцінюється лінгвістичним термом $d_j$; $\omega_{j_1}$ – ваговий коефіцієнт правила з порядковим номером $j_p$ число із діапазону $[0,1]$, що задає відносну вагу правила при нечіткому логічному виводі; $m$ – кількість термів, що використовуються для лінгвістичної оцінки вихідної змінної.

## 18.2. Приклад використання систем нечіткого виводу в задачах управління

### 18.2.1. Нечітка модель управління змішувачем води при прийнятті душу

Як приклад використання систем нечіткого виводу у задачах управління розглянемо керування змішувачем води при прийнятті душу.

При прийнятті душу на вхід змішувача подається холодна і гаряча вода по магістральним трубам. Найбільш комфортні умови для душу створюються при наявності на виході змішувача теплої води постійної температури. Оскільки під час прийняття душу може спостерігатися нерівномірні витрати води, температура води на виході змішувача буде коливатися, що викличе необхідність ручної зміни подачі холодної та гарячої води. Задача управління буде полягати в тому, щоб зробити регулювання температури води автоматичною,

забезпечивши постійну температуру води на виході змішувача (рис. 18.2.1).

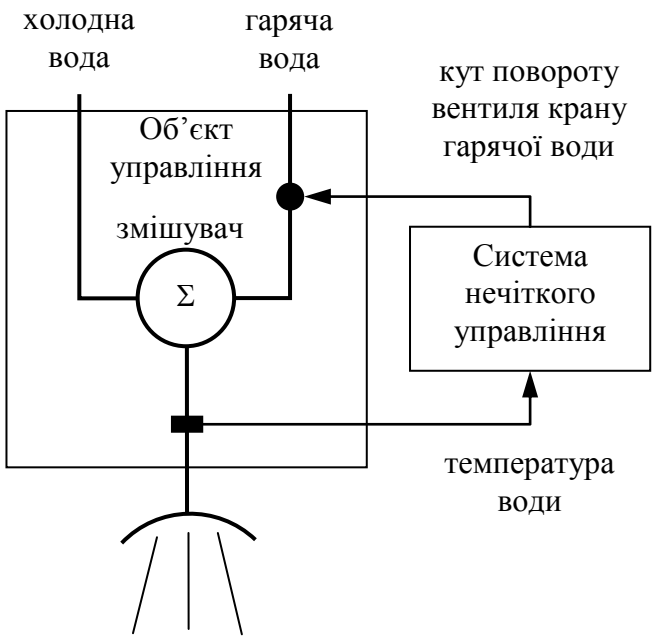


Рис. 18.2.1. Схема моделі нечіткого управління змішувачем при прийнятті душу.

Досвід прийняття душу дозволяє сформулювати декілька евристичних правил, які ми використовуємо у випадку регулювання температури води на виході змішувача:

1. Якщо вода гаряча, то слід повернути вентиль крану гарячої води на великий кут за рухом годинникової стрілкою.

2. Якщо вода не дуже гаряча, то слід повернути вентиль крану гарячої води на невеликий кут за рухом годинникової стрілкою.

3. Якщо вода тепла, то залишити вентиль крану гарячої води у тому положенні.

4. Якщо вода прохолодна, то слід повернути вентиль крану холодної води на невеликий кут проти руху годинникової стрілки.

5. Якщо вода холодна, то слід повернути вентиль крану холодної води на великий кут проти руху годинникової стрілки.

Ця інформація буде використовуватися при побудові бази правил системи нечіткого виводу, яка дозволить реалізувати дану модель управління.

### 18.2.2. Побудова бази нечітких лінгвістичних правил

Для формування бази правил системи нечіткого виводу необхідно попередньо визначити вхідні і вихідні лінгвістичні змінні. Очевидно, в якості вхідної лінгвістичної змінної слід використовувати температуру води на виході змішувача або формально: $\beta_1$ – „температура води”. В якості вихідної лінгвістичної змінної будемо використовувати кут попороту вентиля крану гарячої води або формально: $\beta_2$ – „кут повороту”.

В такому випадку система нечіткого виводу буде мати 5 правил нечітких продукцій такого виду:

ПРАВИЛО_1: ЯКЩО „вода гаряча” ТО „повернути вентиль крану гарячої води на великий кут за рухом годинникової стрілкою”.

ПРАВИЛО_2: ЯКЩО „вода недуже гаряча” ТО „повернути вентиль крану гарячої води на невеликий кут за рухом годинникової стрілкою”.

ПРАВИЛО_3: ЯКЩО „вода тепла” ТО „то залишити вентиль крану гарячої води у тому положенні”.

ПРАВИЛО_4: ЯКЩО „вода прохолодна” ТО „повернути вентиль крану гарячої води на невеликий кут за проти руху годинникової стрілки”.

ПРАВИЛО_5: ЯКЩО „вода холодна” ТО „повернути вентиль крану гарячої води на великий кут за проти руху годинникової стрілки”.

### 18.2.2. Фазифікація вхідної та вихідної змінної

В якості терм-множини першої лінгвістичної змінної будемо використовувати множину $T_1$ = {„гаряча”, „недуже гаряча”, „тепла”, „прохолодна”, „холодна”) або в символічному вигляді $T_1$ = {PB, PS, Z, NS, NB} з функціями належності, які представлені на рис. 18.2.2.

В якості тер-множини другої лінгвістичної змінної будемо використовувати множину $T_2$ = {„великий кут за годинниковою стрілкою”, „невеликий кут за годинниковою стрілкою”, „залишити без змін”, „невеликий кут проти годинникової стрілки”, „великий кут проти годинникової стрілки”) або в символічному вигляді $T_2$ = {PB, PS, Z, NS, NB} з функціями належності, які представлені на рис. 18.2.3.

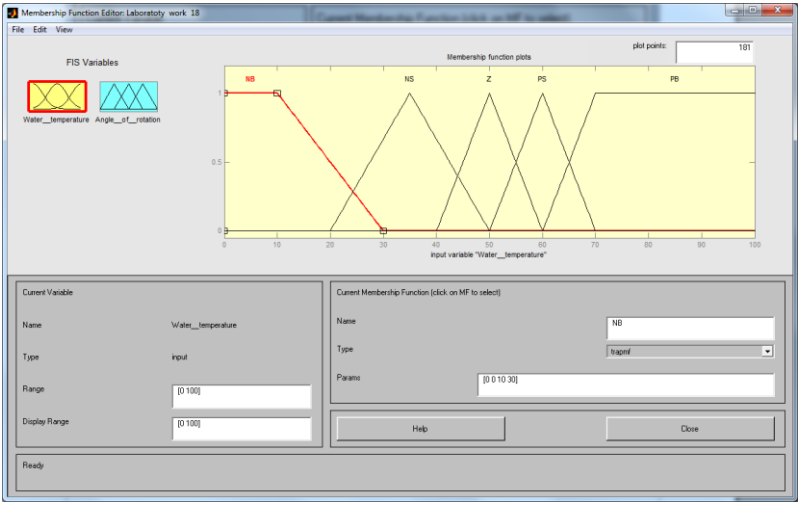


Рис. 18.2.2. Графіки функції належності для терму лінгвістичної змінної „Температура води”.

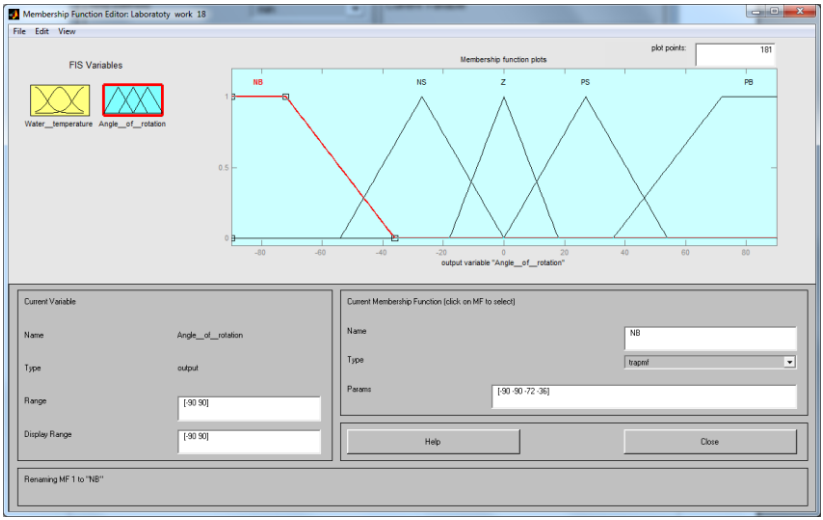


Рис. 18.2.3. Графіки функції належності для терму лінгвістичної змінної „Кут повороту вентиля крану”.

При цьому температура води вимірюється у градусах Цельсія, а кут повороту – у градусах. Для моделі нечіткого виводу використовуємо алгоритм Мамдані (рис. 18.2.4).

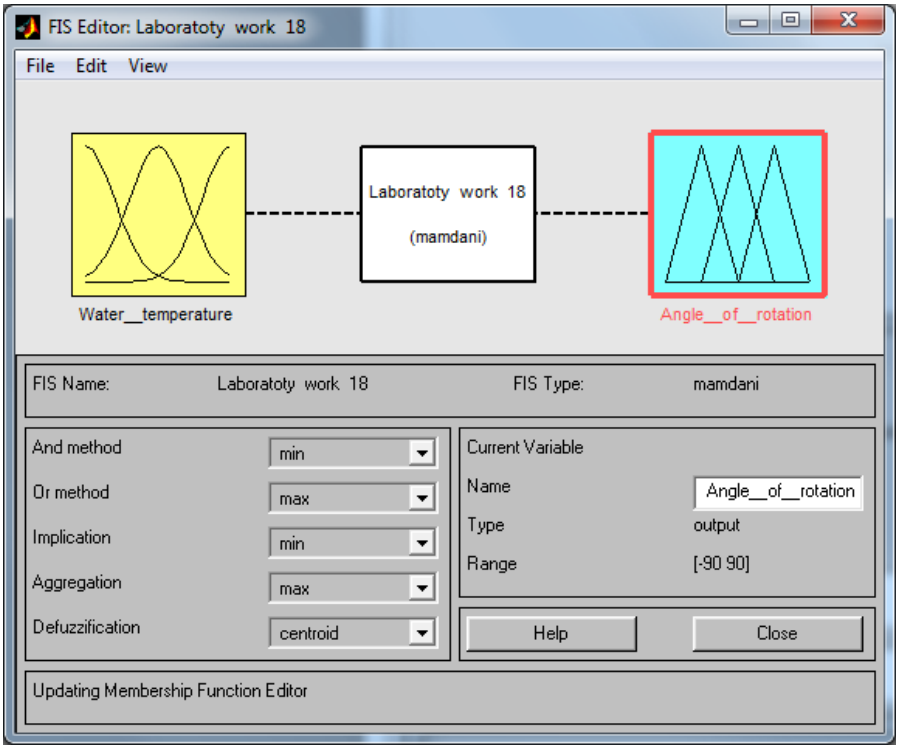


Рис. 18.2.4. Вікно FIS Editor модуля fuzzy системи MATLAB.

База нечітких правил представлена на рис. 18.2.5. Розглянемо випадок коли температура води на вході змішувача рівна 55 $^{o}$C. У цьому випадку газифікація вхідної лінгвістичної змінної приводить до значення степенів істинності 0,5 для правил нечітких продукцій із номерами 2 і 3. Ці правила вважаються активними і використовуються у даному процесі нечіткого виводу (рис. 18.2.6).

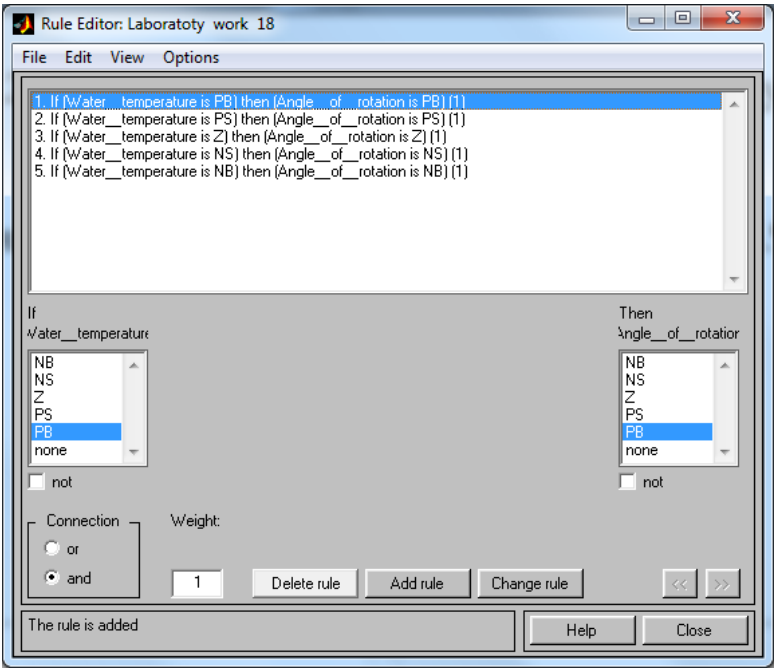


Рис. 18.2.5. Вікно Rule Editor модуля fuzzy системи MATLAB.

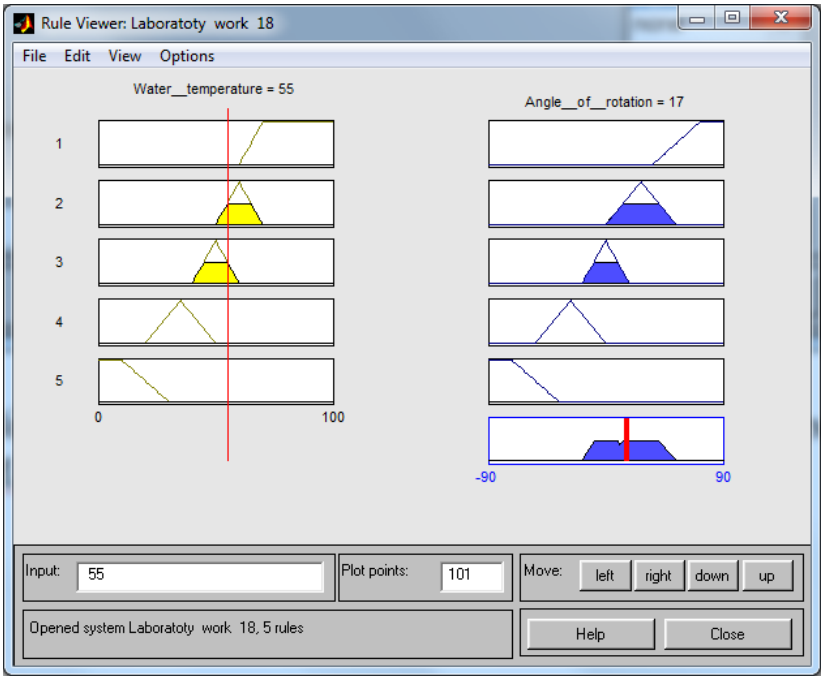


Рис. 18.2.6. Вікно Rule Viewer модуля fuzzy системи MATLAB.

## 18.3. Завдання для виконання

1. Ознайомитися із теоретичними відомостями.
2. У системі MATLAB запустити модуль fuzzy.
3. Реалізувати нечітку модель управління змішувачем води при прийнятті душу.
4. Провести моделювання при різних значеннях температури води на вході змішувача.
5. Оформити звіт про виконання лабораторної роботи.

## Лабораторна робота № 19
## „Розв'язок задач нечіткої кластерізації в MATLAB"

### 19.1. Теоретичні відомості

Терміном кластер ний аналіз прийнято позначати сукупність методів, підходів та процедур, які розроблені для проблеми формування однорідних класів у довільній проблемній області.

Виявлення або знаходження кластерів у множині даних у сукупності, що використовується, повинна задовольняти таким вимога:

– кожний кластер повинен представляти собою концептуально однорідну категорію і містити подібні об'єкти із близькими значеннями властивостей або ознак;
– сукупність всіх кластерів повинна бути вичерпаною, тобто охоплювати всі об'єкти досліджуваної сукупності;
– кластери повинні бути взаємно однозначними, тобто ні один із об'єктів досліджуваної сукупності не повинен одночасно належати двом різним кластерам.

#### 19.1.1. Загальна формальна постановка задачі нечіткого аналізу

Нехай досліджувана сукупність даних представляє собою скінчену множину елементів $A = \{a_1, a_2, ..., a_n\}$, яку можна назвати множиною об'єктів кластерізації. Також визначимо скінчену множину атрибутів $P = \{p_1, p_2, ..., p_n\}$,кожен з яких кількісно представляє деяку властивість або характеристику елементів проблемної області, що розглядається. При цьому натуральне число $n$ визначає загальну кількість об'єктів даних, а натуральне число $q$ – загальну кількість ознак об'єктів, що вимірюються.

У подальшому будемо вважати, що для кожного із об'єктів кластерізації деяким чином виміряні всі ознаки множини $P$ в деякій кількісній шкалі. Тим самим кожному із елементів $a_i \in A$

поставлений у відповідність деякий вектор $\boldsymbol{x}_i = \{x_1^i, x_2^i, ..., x_q^i\}$, де $x_j^i$ – кількісне значення ознаки $p_j \in P$ для об'єкту даних $a_i \in A$. Для визначеності будемо вважати, що всі $x_j^i$ приймають деякі дійсні значення, тобто $x_j^i \in \boldsymbol{R}$.

В загальному проблема вимірювання властивостей або ознак у об'єктів даних є нетривіальною і має самостійне значення. Зокрема, процес вимірювання властивостей може бути реалізований у різних шкалах, кожна із яких характеризується допустимим перетворенням даних. У зв'язку із цим для уточнення особливостей процесу вимірювання розрізняють декілька типів шкал вимірювань.

**Шкала найменувань** або номінальна шкала. Є найбільш простою із всіх шкал вимірювань, оскільки може бути використана для встановлення відношення еквівалентності елементів відносно ознаки, що розглядається. В даному випадку у процесі вимірювання деякої ознаки об'єкту ставиться у відповідність деякий символ або номер, який лише відрізняє одне значення ознаки від другої. Бінарну шкалу, яка складається із двох елементів, які позначаються довільними символами, наприклад: $\{0, 1\}$ або $\{+, -\}$, зручно вважати частковим випадком шкали найменувань.

**Порядкова шкала** або шкала порядку. У додаток до різнорідності елементів за значенням ознак дозволяє встановити відношення порядку елементів відносно ознаки, що розглядається. В цьому випадку у процесі вимірювань ознаки об'єкту ставиться у відповідність, як правило, деяке натуральне число або ціле число, яке може бути інтерпретовано як значення ознаки у балах.

**Інтервальна шкала** або шкала інтервалів. У додаток до порядку елементів за значенням ознак дозволяє встановити рівність інтервалів значень ознаки, що розглядається. В цьому випадку у процесі вимірювання ознаки об'єкту ставиться у відповідність, як правило, деяке дійсне число, рівне значенню цієї ознаки.

**Шкала відношень**. У додаток до рівності елементів за значенням ознак дозволяє встановити рівність відношень значень ознаки, що розглядається. В цьому випадку у процесі вимірювання ознаки об'єкту ставиться у відповідність також деяке дійсне число, рівне значенню цієї ознаки.

Якщо повернутися до вимірюванню ознак у об'єктів кластерізації, то саму множину ознак слід вибирати таким чином, щоб $x_j^i$ були виміряні в шкалі відношень або шкалі інтервалів. Сама в цьому випадку результати нечіткої кластерізації мають змістовну інтерпретацію, адекватну проблемі знаходження нечітких кластерів.

Вектори значень ознак $\boldsymbol{x}_i = \{x_1^i, x_2^i, ..., x_q^i\}$ зручно представляти у виді матриці даних **D** розмірності ($n \times q$), яка рівне значенню вектора $\boldsymbol{x}_i$.

Задача нечіткого кластерного аналізу формулюється таким чином: на основі вихідних даних **D** визначити таке нечітке розбиття або покриття множини *A* на задане число *c* кластерів, яке доставляє екстремум деякої цільової функції серед всіх нечітких розбиттів або екстремум цільової функції серед всіх нечітких покриттів.

Одним із варіантів задачі нечіткого кластерного аналізу, для розв'язку якої може бути використана спеціальна функція `fcm` системи MATLAB, базується на алгоритмі розв'язку методом нечітких *c*-середніх.

### 19.1.2. Розв'язок задачі нечіткої кластеріазації в командному рядку системи MATLAB

Функція командної строчки `fcm` передбачена для розв'язку задачі кластерізації із використанням алгоритму FCM. Вона може бути визначена одним із наступних форматів:

```
[center,U,obj_fcn]=fcm(data,cluster_n)
```

або

```
[center,U,obj_fcn]=fcm(data,cluster_n,options).
```

Вхідними аргументами цієї функції є:

– `data`: матриця вихідних даних **D** кластерізації, *i*-строчка в якій представлено інформацію про об'єкт нечіткої кластерізації $a_i \in A$ у форматі вектора $\boldsymbol{x}_i = \{x_1^i, x_2^i, ..., x_q^i\}$, де $x_j^i$ – кількісне значення ознаки $p_j \in P$ для об'єкта даних $a_i \in A$;

– `cluster_n`: число шуканих нечітких кластерів $c$ ($c \in \boldsymbol{N}, c > 1$).

Вихідними аргументами цієї функції є:

– `center`: матриця центрів шуканих нечітких кластерів $\nu_j^k$ ($\forall k \in \{2,..., c\}, \forall p_j \in P$), кожна строчка якої представляє координати центру одного із нечітких кластерів у формі вектора $\nu_k$ ($\forall k \in \{2,..., c\}$).

– `U`: матриця значень функції належності шуканого нечіткого розбиття $\mu_k(a_i)$ ($\forall k \in \{2,..., c\}, \forall a_i \in A$);

– `obj_fcn`: значення цільової функції $f(A_k, \nu_j^k)$ на кожній із ітерацій роботи алгоритму.

Функція `fcm(data,cluster_n,options)` може бути викликана за додатковими аргументами `options`, які призначені для управління процесом нечіткої кластерізації, а також для зміни критерію зупинки роботи алгоритму та/або відображення інформації на екрані монітора.

Ці додаткові аргументи мають такі значення:

– `options(1)`: експоненціальна вага *m* для розрахунку матриці нечіткого розбиття `U` (по замовчуванню *m* = 2);

– `options(2)`: максимальне число ітерацій *s* (по замовчуванню *s* = 100);

– `options(3)`: параметр збіжності алгоритму ε (по замовчуванню *s* = 0,00001);

– `options(4)`: інформація про поточну ітерацію, яка відображається на екрані монітора (по замовчуванню це значення рівне 1).

Якщо будь яке із значень додаткових аргументів рівне `NaN`, то для цього аргументу використовується значення по замовчуванню. Функція `fcm` закінчує свою роботу, коли алгоритм FCM виконає максимальну кількість ітерацій *s*, або коли різниця між значеннями цільвих функцій на двох послідовних ітераціях буде менше заданого апріоі значення параметра збіжності алгоритму ε.

## 19.2. Приклад розв'язку задачі нечіткої кластерізації

В якості прикладу розглянемо множину даних, які містяться в системі MATLAB і використовуються як приклад для тестової сукупності об'єктів нечіткої кластерізації. Ці дані представляють собою матрицю даних **D** розмірності (140×2), які містяться у файлі fcmdata.dat та інсталюються разом із системою MATLAB. В нашому випадку матриця даних **D** відповідає 140 об'єктам, для кожного із яких виконані вимірювання за двома ознаками, що є зручно для візуалізації у двовимірному просторі на площині.

Розв'язок задачі реалізований у m-файлі `Laboratory_work_19`.

```
load fcmdata.dat;
plot(fcmdata(:,1),fcmdata(:,2),'o','color',
'k');
[center,U,obj_fcn]=fcm(fcmdata,2)
maxU=max(U);
index1=find(U(1,:)==maxU);
index2=find(U(2,:)==maxU);
line(fcmdata(index1,1),fcmdata(index1,2),'line
style',
'none','marker','o','color','g');
```

```
line(fcmdata(index2,1),fcmdata(index2,2),'line
style','none','marker','x','color','r');
hold on;
plot(center(1,1),center(1,2),'ko','markersize'
,10,'LineWidth',2);
plot(center(2,1),center(2,2),'ko','markersize'
,10,'LineWidth',2);
```

Результат розв'язку задачі нечіткої кластерізації для 2-х нечітких кластерів з використанням вказаної послідовності команд може бути візуалізований (рис. 19.2.1–19.2.3)

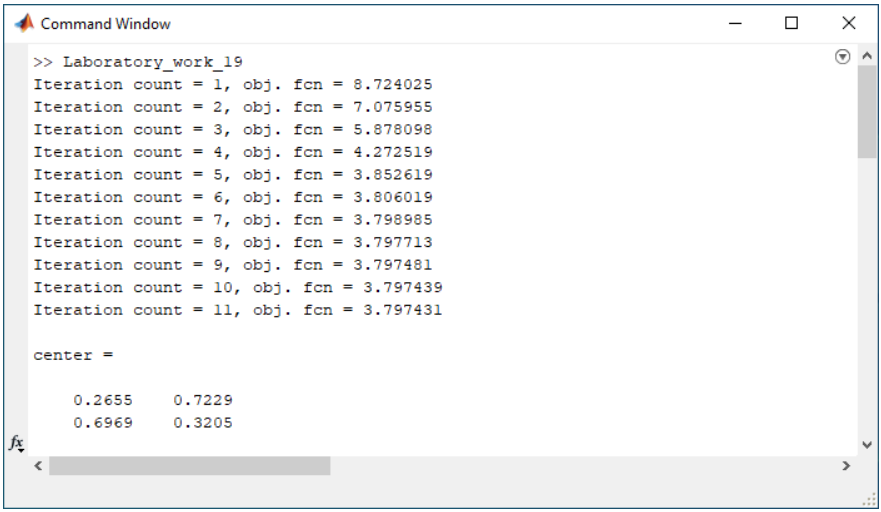


Рис. 19.2.1. Результат розв'язку задачі нечіткої кластерізації у вікні Command Windows системи MATLAB.


Рис. 19.2.2. Результат розв’язку задачі нечіткої кластерізації у вікні Command Windows системи MATLAB.

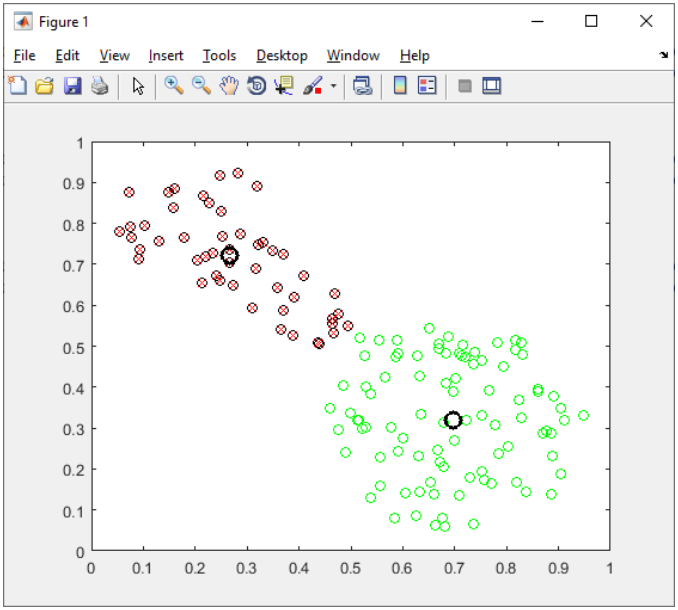


Рис. 19.2.3. Результат розв’язку задачі нечіткої кластерізації для матриці даних із файла `fcmdata.dat`.

### 19.3. Завдання для виконання

1. Ознайомитися із теоретичними відомостями.
2. У системі MATLAB розв'язати задачу нечіткої кластерізації для матриці даних із файла `fcmdata.dat` для 3, 4-кластерів.
3. Порівняти та проаналізувати результати розв'язку.
4. Оформити звіт про виконання лабораторної роботи.

## Список літератури


1. Томашевський В. М. Моделювання систем. Київ : Видавнича група BHV, 2005. 352 с.
2. Лазарєв Ю. В. MATLAB і моделювання динамічних систем : навчальний посібник. Глава 2. Програмування у Matlab. Київ : НТУУ „КПІ", 2009. 60 с.
3. Лазарєв Ю. В. MATLAB і моделювання динамічних систем : навчальний посібник. Глава 3. Пакет програм Simulink. Київ : НТУУ „КПІ", 2009. 79 с.
4. Лазарєв Ю. В. MATLAB і моделювання динамічних систем : навчальний посібник. Глава 4. Засоби взаємодії Matlab з Simulink. Київ : НТУУ „КПІ", 2009. 63 с.
5. Лазарєв Ю. В. MATLAB і моделювання динамічних систем : навчальний посібник. Глава 5. Моделювання динамічних систем. Київ : НТУУ „КПІ", 2009. 65 с.
6. Желдак Т. А. Коряшкіна Л. С., Ус С. А. Нечіткі множини в системах управління та прийняття рішень : навчальний посібник. Дніпро : НТУ ДП, 2020. 386 с.
7. Івахів О. Наконечний М. Основи побудови систем керування з нечіткою логікою: навчальний посібник. Львів : Растр-7, 2017. 129 с.

8. Баськова А. В. Лотиш В. В. Клітинковий автомат Дж. Конуея «Життя» та конфігурація «Сад Едему». *Комп'ютерно-інтегровані технології: освіта, наука, виробництво*, 2013. № 12. С. 86–89. URL: http://nbuv.gov.ua/UJRN/Kitonv_2013_12_19.
9. Оленич І. Б. Нечітка логіка та нечітке моделювання. Львів : ЛНУ імені Івана Франка, 2022. 210 с.