ІННОВАЦІЙНІ ТЕХНОЛОГІЇ КОМП'ЮТЕРНОГО МОДЕЛЮВАННЯ ФІЗИЧНИХ ТА ІНФОРМАЦІЙНИХ ПРОЦЕСІВ

Синергетика Інформаційно-Комунікаційних Систем

Михайло Мар'ян
Володимир Шебень
Наталія Юркович

ІННОВАЦІЙНІ ТЕХНОЛОГІЇ КОМП'ЮТЕРНОГО МОДЕЛЮВАННЯ ФІЗИЧНИХ ТА ІНФОРМАЦІЙНИХ ПРОЦЕСІВ

Синергетика Інформаційно-Комунікаційних Систем

INNOVATIVE TECHNOLOGIES OF COMPUTER MODELING FOR PHYSICAL AND INFORMATION PROCESSES

Synergetics of Information and Communication Systems

INOVATÍVNE TECHNOLÓGIE POČÍTAČOVÉHO MODELU FYZIKÁLNYCH A INFORMAČNÝCH PROCESOV

Synergetika Informačných a Komunikačných Systémov

Михайло Мар'ян/Mykhaylo MAR'YAN/ Michajlo MARIAN
Володимир Шебень/Vladimir SEBEN/ Vladimír ŠEBEŇ
Наталія Юркович/Nataliya YURKOVYCH/ Natalia JURKOVIČ

Prešov 2019

University of Prešov in Prešov,
Faculty of Humanities and Natural Sciences

Prešovská univerzita v Prešove
Fakulta humanitných a prírodných vied

Title / Názov: **INNOVATIVE TECHNOLOGIES OF COMPUTER MODELING FOR PHYSICAL AND INFORMATION PROCESSES: Synergetics of Information and Communication Systems**

INOVATÍVNE TECHNOLÓGIE POČÍTAČOVÉHO MODELU FYZIKÁLNYCH A INFORMAČNÝCH PROCESOV:
Synergetika Informačných a Komunikačných Systémov

Authors / Autori: **Prof. Mykhaylo Mar'yan, DrSc**./
Prof. Michajlo Marian, DrSc.
Dr.h.c. doc. PaedDr. Vladimir Seben, PhD./
Dr.h.c.doc. PaedDr. Vladimír Šebeň, PhD.
Ass Prof. Nataliya Yurkovych, PhD./
Doc. Natalia Jurkovič, PhD.

Reviewers / Recenzenti: Prof. Vasyl Rubish, DrSc.
Prof. Oleksandr Grabar, DrSc.

Vydala: © Prešovská univerzita v Prešove, 2019
Vydanie: Prvé

ISBN: 978-80-555-2278-4

ЗМІСТ

ПЕРЕДМОВА

Інформаційні технології (Information Technology - IT) , інформаційно-комунікаційцні технології та системи (Information and Communication Technologies - ICT) визначають сукупність методів, процесів і програмно-технічних засобів, інтегрованих з метою збирання, обробки, зберігання, розповсюдження, відображення та використання інформації. Термін “Технологія“ походить із грецької мови від слова Τεχνολογία. Його утворено злиттям двох слів: «техне» (τεχνο)— майстерність, уміння та «логос» (λογία)— учення, поняття. Слово «Інформація» в перекладі з латинської мови визначає та містить відомості про навколишній світ. З цієї точки зору, інформаційні технології формують алгоритми та способи сприймання, розвитку оточуючого інформаційного середовища – науки, освіти, економіки, екології та вдосконалення самої людини (інтелектуальний, емоційний, казуальний розвиток). Розглянуті нами в даній монографії умови та способи формування самоорганізованих структур в системах різноманітної природи в рамках синергетичного підходу передбачають саме створення та наявність «інформаційних технологій». Це означає, що рівні структурування та фрактальність інтелектуальних середовищ (інтелектуальних матеріалів, програмного та апаратного забезпечення та інших) може бути притаманна також інформаційним технологіям та середовищам, в яких вони функціонують. І як наслідок, розвиток інформаційно-комунікаційних систем (розумний будинок, розумне авто, розумні речі та аксесуари, інші) містить елементи способів та алгоритмів формування інтелектуальних систем. Це питання надзвичайно актуальне і для екології, освіти, науки, проблем енергозбереження.

Наступний аспект сучасних інформаційно-комунікаційних систем полягає в тому, що наукові області та напрямки, а саме - програмування та комп’ютерне моделювання, експериментальні та теоретичні дослідження, інформаційні системи та освіта, є сумісні, інтегровані, синхронізовані, об’єднані завдяки “одному об’єкту” та інформаційному середовищу, які

формуються та динамічно розвиваються при застосуванні синергетичного підходу та фрактальності. *(Це прекрасно проявляється в сучасних інформаційних технологіях: синхронізація застосунків для різних пристроїв та платформ з використанням хмарних технологій).* Разом з тим, завдяки цілісності синергетики проявляються кардинально нові властивості, які ідентифікують нове інформаційне середовище та нові об'єкти, нові способи без дисипативного функціонування.

Процеси самоорганізації відігріють надзвичайно важливу роль у системах живої та неживої природи. Значна кількість фундаментальних задач у фізиці, екології, економіці, програмуванні, інформаційних технологіях пов'язана із дослідженням явищ у відкритих за енергією, масою та інформацією системах за допомогою міждисциплінарного напрямку, який особливо бурхливо розвивається на протязі останніх десятиліть – синергетики. Одне з чільних місць в синергетиці займає вивчення особливостей функціонування відкритих систем (інформаційно-комунікаційних, екологічних, освітніх систем, алгоритмічних мов та систем програмування, функціонування та платформ штучного інтелекту) та процесів самоорганізації у них. Синергетика вивчає самоузгоджену поведінку відкритих систем, що містять велике число компонент (підсистем, частин). В буквальному розумінні термін синергетика перекладається як сумісна дія, виділяючи таким чином самоузгодженість функціонування частин, що проявляється в поведінці системи як цілого.

Синергетика виступає з'єднуючою ланкою між неорганічними та органічними системами. Найбільш очевидна особливість об'єктів живої природи полягає в тому, що вони здатні до самоорганізації, тобто до спонтанного утворення та розвитку складних структур з певним рівнем функціональної організації. Необхідна передумова ефектів самоорганізації полягає у відкритості системи: обмін із зовнішнім середовищем енергією, масою, інформацією. Саме завдяки цьому потоку із зовнішнього середовища система стає активною, тобто отримує здатність до спонтанного утворення структур.

Процеси самоорганізації не є виключною властивістю біологічних об’єктів, а спостерігаються і в інформаційно-комунікаційних системах, матеріалах штучного інтелекту, зокрема в некристалічних матеріалах, і супроводжуються формуванням самоорганізованих структур на макро-, мікро-, нанорівнях.

Цей підхід реалізується та розвивається в монографії для систем різноманітної природи.

І.ІННОВАЦІЙНІ ТЕХНОЛОГІЇ КОМП'ЮТЕРНОГО МОДЕЛЮВАННЯ ФІЗИЧНИХ ПРОЦЕСІВ

1.1. Середовище візуального програмування Delphi: моделювання взаємодії електромагнітного випромінювання з некристалічними тілами

1.1.1. Самоорганізовані структури і системи

У відкритих системах, тобто системах, в яких можливий обмін речовиною та енергією з навколишнім середовищем, в ході неперервного процесу з просторово-однорідного стану може самостійно сформуватися більш складна просторова або тимчасова структура (наприклад, коливання концентрацій реагентів у часі). Процес самостійного формування структури більш складної, ніж початкова, називають самоорганізацією. Структури, що утворюються в процесі самоорганізації, - це самоорганізовані структури.

Практично всі найбільш складні структури навколо нас - це структури самоорганізовані. Найпростіший, класичний приклад дисипативної самоорганізованої структури - це комірки Бенара - правильні шестигранні конвективні комірки, які виникають в шарі рідини, що підігрівається знизу (Рис. 1.1). Внутрішня структура Землі – це також самоорганізована структура. Структура земної поверхні - результат як внутрішньоземних дисипативних процесів, так і потоку сонячної енергії.

Виникнення дисипативних структур пов'язано з виробленням "позитивної" ентропії - синентропії. Так, наприклад, бенарівська конвекція на кілька порядків більш ефективний спосіб перенесення і розсіювання тепла, ніж теплопровідність. Такі перебудови відбуваються на макроскопічному рівні, і механізм їх полягає в розростанні деяких певних випадково виникаючих флуктуацій. Завдання опису і пояснення виникнення спостережуваних структур

розпадається на два: опис природи і механізму виникнення первинних флуктуацій та опис механізму перетворення їх в макроскопічні структури.

Наочними прикладами дисипативних структур, крім комірок Бенара (Рис.1.1), вихрів Тейлора в потоці Куейта, хімічної реакції Жаботиського-Білоусова, є, наприклад, ще такі природні явища, як циклони, торнадо (смерчі) в атмосфері, а також деякі хмарні структури. Дисипативними структурами в принципі тієї ж природи, хоча і незмірно більш складними, є всі живі істоти і екологічні системи, що підтримують своє існування шляхом неперервного обміну речовиною і енергією з зовнішнім середовищем - виділенням з себе ентропії, поглинанням негентропії – синентропії (Mar'yan & Yurkovych, 2018).

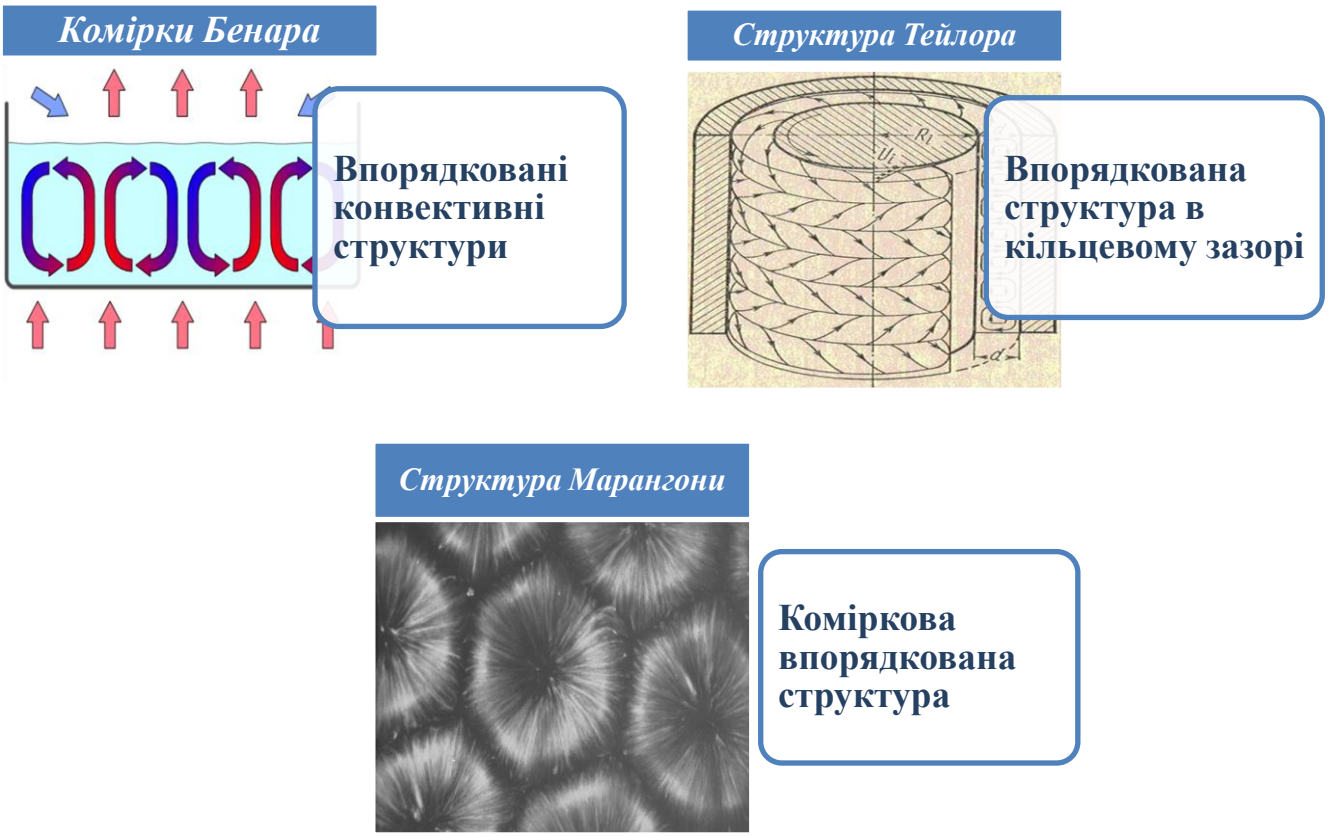


Рис. 1.1. Комірки Бенара - упорядковані конвективні структури у формі циліндричних валів в шарі в'язкої рідини з вертикальним градієнтом температур(а), дисипативна структура Тейлора в кільцевому зазорі між двома коаксіальними циліндрами при обертанні одного з них (б), коміркова впорядкована структура, що виникла в результаті дії сили Марангони(в)

Проблемами дисипативних структур та самоорганізації займалися такі вчені як Герман Хакен, Ілля Пригожин, Грегуар Ніколіс (Prigogine, 1980; Haken, 2006).

Дисипативні (самоорганізовані) структури є результатом розвитку власних внутрішніх нестійкостей в системі. Процеси самоорганізації можливі при обміні енергією і масою з навколишнім середовищем, тобто при підтримці стану поточної рівноваги, коли втрати на дисипацію компенсуються ззовні. Ці процеси описуються нелінійними рівняннями для макроскопічних функцій. Дисипативні структури можна розділити на (Prigogine, 1980):

тимчасові;

просторові;

просторово-часові.

Прикладами тимчасових структур є періодичні, коливальні і хвильові процеси.

Типовими прикладами просторових структур є: перехід ламінарної течії в турбулентну, перехід дифузійного механізму передачі тепла в конвективний. Характерні приклади: турбулентність, комірки Бенара і надгратка пор.

Прикладами просторово-часових структур є режим генерації лазера і коливальні хімічні реакції.

Дисипативні системи - динамічні системи, у яких енергія упорядкованого процесу переходить в енергію неупорядкованого процесу, в кінцевому рахунку в теплову. Приклади дисипативних систем: тверді тіла, між якими діють сили сухого або рідинного тертя; в'язке (або пружньов’язке) середовище; коливання електричного струму в системі контурів і т. д. Практично всі системи, з якими доводиться реально стикатися в земних умовах, є дисипативними системами. Розглядати їх як консервативні, тобто як системи, в яких механічна енергія зберігається, можна лише в окремих випадках, наближено відволікаючись від низки реальних властивостей системи.

До числа самоорганізованих структур відносять просторово-періодичні конвективні системи, системи з хаотичною поведінкою, деякі типи динамічних фрактальних структур, періодичні просторово-часові структури (автоколивання і автохвилі) (Mar'yan & Yurkovych, 2018). Одним з механізмів виникнення самоорганізації є процеси перерозподілу енергії за рахунок формування конвективних потоків. Очевидно, що інтенсивний приплив енергії ззовні призводить до локального "перегріву" частини системи, а отже і до значного падіння ентропії системи в цілому. Крім того, якщо приплив енергії є неперервним, то в стійкому стані теплопотік повинен врівноважуватися передачею енергії в зовнішнє середовище. Перерозподіл енергії всередині системи може здійснюватися за рахунок звичайних механізмів переносу енергії (наприклад, теплообміну). Однак, якщо швидкість теплопередачі виявляється занадто мала, а сумарна ентропія системи виявляється нижче деякого критичного значення, то в системі стає можливим зміна механізму теплопереносу з утворенням конвективних або турбулентних потоків. При цьому, в однорідних, симетричних системах може спостерігатися формування впорядкованих самоорганізовних структур.

Розглянемо як приклад вільну конвекцію в рідинах, яка може бути викликана дією наступних сил:

термогравітаційних - виникають у полі сили тяжіння через різні густини областей рідини, що мають різну температуру.

концентраційних - виникають у полі сил тяжіння, в системі, що має області з різною концентрацією домішок, а значить, і густиною.

магнітогідродинамічних - виникають у рідинах, що містять феромагнітні частинки.

Окрім гравітаційних сил вільну конвекцію ще може викликати дія поверхневої сили - сили Марангони (Рис. 1.1). Ця сила діє в системах, в яких виникає різниця сил поверхневого натягу на поверхні і в глибині нерівномірно нагрітої рідини, так як вони сильно залежать від температури.

1.1.2. Ефект впливу електромагнітного випромінювання на формування самоорганізованих структур в шарах некристалічних матеріалів систем *As-S(Se)*: самоорганізовані структури

Представлена модель впливу електромагнітного випромінювання у видимій області спектру ($\Lambda = 0.63$ *мкм*, густина потужності лазерного випромінювання $P \le 1$ $Вт/см^2$) на формування дисипативних структур в шарах некристалічних матеріалів систем $As-S(Se)$ та її аналіз на основі експериментальних досліджень (Mar'yan & Yurkovych, 2018).

При відсутності засвітки окремі м'які атомні конфігурації N_{20} статистично рівномірно розподілені по об'єму зразка. Розглянуто випадок електромагнітного поля $E(r,t)$ падаючої нормально до поверхні шару некристалічного тіла (НКТ) вздовж осі z хвилі, для якого на поверхні $r(x,y)$ шару виділено напрямок, та випадок аксіально-симетричного поля з розподілом Гауса вздовж перерізу пучка. Динаміка зміни числа атомів у м'яких конфігураціях N_2 та температури T при дії випромінювання задається системою рівнянь:

$$\frac{\partial N_2}{\partial t} = G - \frac{N_2 - N_{20}}{\tau_{rel}} + D \cdot \nabla^2 N_2, \qquad (1.1)$$

$$\rho C \frac{\partial T}{\partial t} = div(\chi \cdot grad(T)) + W(T) - Q(T), \qquad (1.2)$$

де $G = \gamma P \beta V / \hbar\omega$ – швидкість генерації атомів у м'які конфігурації (β – квантовий вихід; V – об'єм системи; $I = I_0 \exp(-\gamma \cdot z)$ – інтенсивність електромагнітної хвилі; γ – коефіцієнт поглинання; $P = E^2/(s_0 t_p)$ – густина інтенсивності опромінення ($s_0 = \pi \cdot r_0^2$ –ефективний переріз пучка опромінення, r_0 – ефективний радіус пучка опромінення, t_p – час опромінення)); τ_{rel} – час

релаксації в метастабільний стан; D – коефіцієнт дифузії; ρ, C і χ – густина, теплоємність та коефіцієнт теплопровідності матеріалу відповідно; $W(T)=\gamma\cdot P$ – потужність лазерного теплового джерела; $Q(T)$ – теплообмін з оточуючим середовищем. Просторова варіація числа атомів у м'яких конфігураціях та температури системи при дії випромінювання впливають зворотнім чином на ширину квазізабороненої зони E_g та коефіцієнт поглинання випромінювання γ (Mar'yan & Yurkovych, 2013). Фотоіндукований приріст коефіцієнта поглинання $\Delta\gamma$ на частоті ω поглинання лазерного випромінювання в лінійному наближенні розраховується згідно співвідношення $\gamma(\omega)=\gamma_0+\Delta\gamma$, де γ_0 – коефіцієнт поглинання до опромінення, і визначається потужністю випромінювання P та структурно-чутливими параметрами матеріалу НКТ через зміну N_2, T: $\Delta\gamma=\frac{1}{\hbar}\left(\frac{\partial\gamma}{\partial\omega}\right)_s\cdot(\beta_{N_2}(N_2-N_{20})+\beta_T(T-T_0))$. Тут коефіцієнти β_{N_2} і β_T враховують зміну коефіцієнта поглинання при переході атомів у м'яку конфігурацію та при нагріванні. Встановлено, що у випадку значного оптичного поглинання (для халькогенідних склоподібних напівпровідників систем $As-S(Se)$ у видимій області спектру $\gamma_0\approx(10^2\div10^3)$ $см^{-1}$) існує область значень хвильового вектора

$$k^2\le k_c{}^2=\frac{(G_0-G_{\min})}{D\tau_{rel}G_{\min}},\quad G_{\min}=\frac{\gamma_0\hbar}{\tau_{rel}\beta_{N_2}\left(\frac{\partial\gamma}{\partial\omega}\right)_s}\left(1-\frac{\beta_T}{a_T\beta_{N_2}}\right),\ a_T=\frac{\chi}{\rho C}, \qquad (1.3)$$

в якій інкремент затухання $\lambda(k)=D(k_c^2-k^2)\ge0$. Це свідчить про втрату стійкості однорідного розподілу атомів у м'яких конфігураціях. Внаслідок цього при $G_0\ge G_{\min}$ формується і стає стійкою одномірна просторова структура розподілу атомів у м'яких конфігураціях у випадку лінійного поля опромінення або неоднорідна радіально-кільцева структура з числом променів $m=\frac{k_c^2r_0^2}{4}-1=\frac{G_0-G_{\min}}{4D\tau_{rel}G_{\min}}\cdot r_0^2-1$ у випадку аксіальної симетрії поля (Mar'yan &

Yurkovych, 2019). Період L_c та час життя τ_{life} утвореної неоднорідної структури визначаються співвідношеннями

$$L_c = 2\pi/k_c = \frac{2\pi}{\sqrt{\frac{(G_0 - G_{\min})}{D\tau_{rel}G_{\min}}}}\ ,\ \tau_{life} = \left(\frac{(G_0 - G_{\min})}{\tau_{rel}G_{\min}} - D\cdot k^2\right)^{-1} \qquad (1.4)$$

і залежать від густини потужності випромінювання.

Проведені дослідження взаємодії електромагнітного випромінювання із шарами НКТ, отриманих в сильно нерівноважних умовах, та враховано можливість формування кластерів м'яких конфігурацій. Динаміка зміни числа атомів у м'яких конфігураціях та функції розподілу за розмірами x кластерів м'яких конфігурацій $g(x,r,t)$ при дії випромінювання в цьому випадку задаються рівняннями:

$$\frac{\partial N_2}{\partial t} = G - \frac{N_2 - N_{20}}{\tau_{rel}} - 4\pi\int_0^\infty x^2 V(x) g(x,r,t)dx + D\cdot\nabla^2 N_2\ , \qquad (1.5)$$

$$\frac{\partial g(x,r,t)}{\partial t} = -\frac{\partial}{\partial x}[V(x)g(x,r,t)] + D(x)\nabla^2 g(x,r,t). \qquad (1.6)$$

Тут $D(x)$ – коефіцієнт дифузії кластерів, який апроксимовано виразом $D(x) = \beta_0/x^2$, β_0 - константа дифузії; швидкість росту кластерів при дії електромагнітного випромінювання визначається приростом числа атомів у м'яких конфігураціях і задається співвідношенням $V(x) = D(N_2 - N_{20})/x$. Динаміка зміни температури T при дії електромагнітного випромінювання описується співвідношенням (1.2). Встановлено, що при $G_0 \geq G_{\min}$, де $G_{\min} = \frac{(4\pi N_d \bar{x})^2 \beta_0}{3\tilde{\alpha}_1}$, в системі випадковим чином розподілених кластерів м'яких конфігурацій під дією випромінювання можливий перехід в неоднорідний стан з формуванням дисипативної структури розподілу кластерів м'яких конфігурацій з періодом L_c:

$$L_c = L_0\sqrt{\frac{G_{\min}}{G_0 - G_{\min}}},\ L_0 = \frac{2\pi\tilde{\alpha}_1}{N_d\bar{x}\left[1 + \frac{4\pi N_d \bar{x}\beta_0\beta_T\omega(\partial\gamma/\partial\omega)_s}{3\tilde{\alpha}_1\gamma_0\rho C}\right]}, \qquad (1.7)$$

де N_d – густина дефектних центрів, $\bar{x}$ – середній розмір кластера, $\tilde{\alpha}_0$ і $\tilde{\alpha}_1$ – коефіцієнти, $\nu = D\cdot(N_{2s} - N_{20})$. Характерний час існування самоорганізованої структури

$$\tau_{life} = \left(\frac{(3\nu\tilde{\alpha}_1 - (4\pi N_d \bar{x} + \tilde{\alpha}_0)\beta_0)^2 + 12\nu\beta_0\tilde{\alpha}_0\tilde{\alpha}_1}{16\pi N_d \bar{x}^3 \beta_0 \tilde{\alpha}_1} \right)^{-1}. \qquad (1.8)$$

Проведено аналіз біфуркації розв'язку та його стійкості при $G_0 \geq G_{\min}$. Виявлено індуковані під дією електромагнітного випромінювання процеси самоорганізації структури при втраті стійкості однорідного просторового розподілу атомів у м'яких конфігураціях на рівні порядку, що перевищує середній порядок (Mar'yan & Yurkovych, 2014). При характерних значеннях параметрів електромагнітного випромінювання $He-Ne$-лазера (густина потужності випромінювання $P = 1\ мВт/см^2 \div 1\ Вт/см^2$, частота випромінювання $\omega = 3\cdot 10^{15}\ c^{-1}$, ефективний радіус пучка $r_0 \approx 0.11 см$) та параметрів некристалічних матеріалів систем $As-S(Se)$ для густини потужності випромінювання $G_0 > G_{\min}$, де $G_{\min} \approx 10^{17} \div 10^{18}\ квант/(см^2\cdot c)$, період самоорганізованої структури порядку $L_c \approx (10^{-2} \div 5) мкм$, час життя структури, яка формується на основі розподілу м'яких атомних конфігурацій $\tau_{life} >> 1\ c$, а для структури на основі розподілу кластерів м'яких конфігурацій $\tau_{life} \approx (10^{-5} \div 10^{-1})\ c$. Встановлено прояв самоорганізованих структур та процесів самоорганізації в особливостях поведінки оптичних та структурно-чутливих параметрів As_2S_3 , показана можливість керованої зміни параметрів Самоорганізованих структур як хімічним складом і умовами синтезу, так і параметрами електромагнітного випромінювання (Mar'yan & Yurkovych, 2018). Зокрема, фотоіндукований приріст коефіцієнта поглинання при розвитку нестійкості має порядок $\delta\gamma/\gamma_0 \approx 10^{-3} \div 4\cdot 10^{-2}$. Електронно-мікроскопічні знімки структур, які формуються в As_2S_3 при дії електромагнітного випромінювання $\lambda = 0.63\ мкм,\ P \leq 1.5\ мВт/см^2$, свідчать про наявність неоднорідних структур на

просторових масштабах $(0.5 \div 0.9)$ *мкм*. Результати розрахунку для випадку аксіальної симетрії випромінювання з густиною потужності $P = 1.5$ *мВт/см*^{2} на шари As_2S_3 визначають число променів радіально-кільцевої структури $m = 34$, період структури $L_c \approx 1.2$ *мкм*, що корелює з експериментальними даними (Mar'yan & Yurkovych, 2019).

1.1.3. Середовище візуального програмування Delphi

Інтегроване середовище розробки Delphi - це середовище, в якому є все необхідне для проектування, запуску і тестування створюваних додатків: редактор вихідного коду, відладчик, інструментальні панелі, редактор зображень, інструментарій баз даних[1]. Така інтеграція забезпечує розробника додатків набором інструментів, що гармонійно доповнюють один одного і полегшують процес створення сучасних додатків (Yurkovych, Seben & Mar'yan, 2017).

Delphi - це потомок середовища програмування Turbo Pascal. Назва середовища розробки додатків Delphi походить від назви міста в Стародавній Греції, де знаходився знаменитий Дельфійський оракул - храм Аполона в місті Дельфи, жерці якого займалися передбаченнями. Середовище розробки Delphi має довгу історію – від найпершої версії Borland Delphi (пізніше відома як Delphi 1) розробленою компанією Borland у 1995 році для 16-розрядних програм для Windows, до самої успішної версії Delphi 7 (2002 рік) та до останньої версії на сьогоднішній день Embarcadero RAD Studio XE 7 уже від компанії Embarcadero. Мова програмування Object Pascal та середовище Delphi входять в першу 10-у найбільш популярних та успішних мов програмування та розробки датків на сучасному етапі разом з такими мовами програмування як Java, Ruby, Swift та інші.

Для подолання труднощів на етапі створення інтерфейсу користувача в кінці XX століття широкого поширення отримало візуальне програмування,

[1] Аналогічно можуть бути застосовані середовища програмування C++, Java, Ruby

озброївши програміста наочними засобами конструювання інтерфейсу. Розглянемо дательно процес візуального програмування на прикладі Delphi (Mar'yan & Yurkovych, 2019).

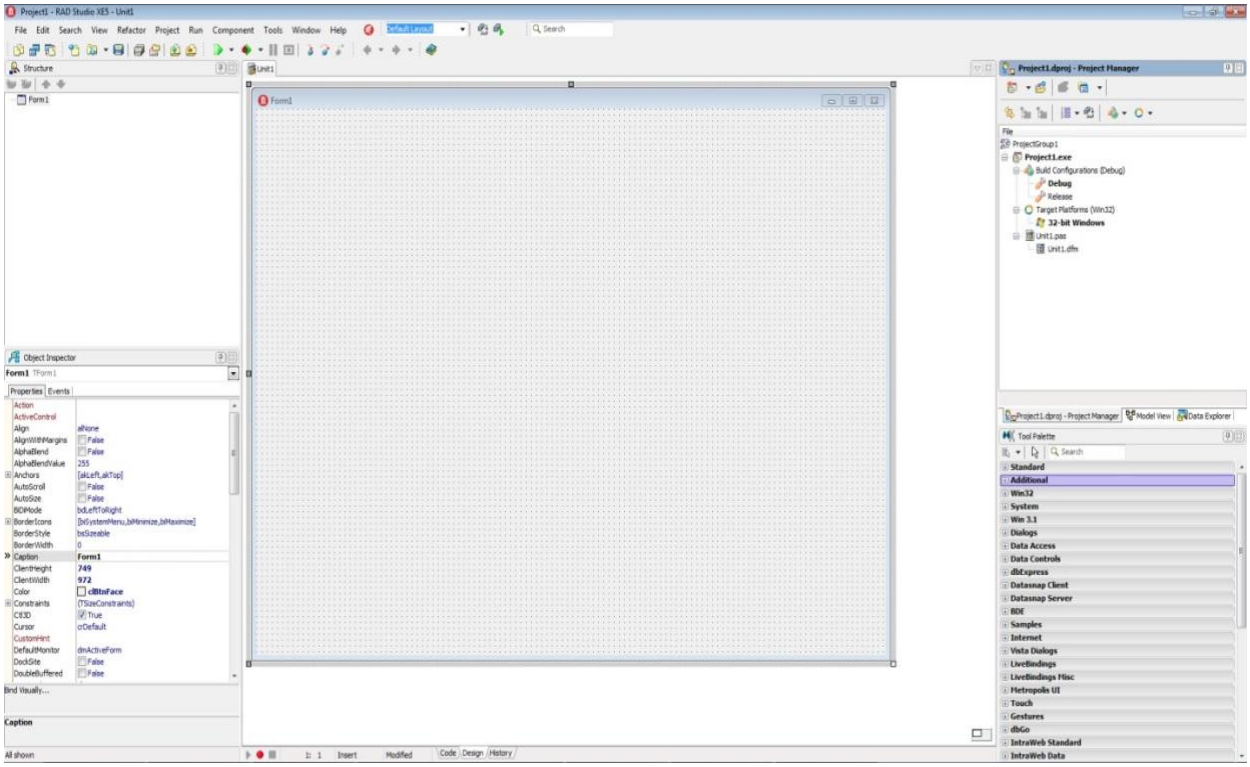


Рис. 1.2. Середовище візуального програмування на мові Delphi – Embarcadero RAD Studio XE 5

Робота проводиться в інтегрованому середовищі розробки (Рис 1.2) (ICP або Integrated Development Environment, IDE) Delphi. Середовище надає програмісту форми, на яких розміщуються компоненти. На форму за допомогою миші поміщаються піктограми компонентів, наявні в бібліотеці Delphi.

За допомогою простих маніпуляцій мишею можна змінювати розміри і положення цих компонентів. При цьому в процесі проектування можна постійно бачити результат - зображення форми і розташованих на ній компонентів. Відпадає необхідність у запуску програми для перевірки того, як виглядає той чи інший компонент вікна, і наступного змінення програмного коду для підбору найбільш вдалих розмірів вікна і других компонентів.

Результати проектування можна бачити, навіть не компілюючи програму, негайно після виконання операції зміни розмірів і положення компонента за допомогою миші.

Але переваги візуального програмування не зводяться тільки до цього. Найголовніше полягає у тому, що під час проектування форми і розміщення на ній компонентів редактор коду Delphi автоматично генерує код програми, включаючи в неї відповідні фрагменти, що описують даний компонент. А потім у відповідних діалогових вікнах користувач може змінити задані за замовчуванням значення властивостей цих компонентів і, при необхідності, написати обробники подій. Тобто проектування зводиться, фактично, до розміщення компонентів на формі, завданням деяких властивостей цих компонентів і написанням, при необхідності, обробників подій (Mar'yan, Seben & Yurkovych, 2018).

1.1.4. Компонент середовища Delphi Chart як засіб графічного відображення даних

Компонент Delphi Chart розміщений на сторінці TeeChart Lite палітри компонентів. Його розробив Девід Бернеда. Це багатий можливостями, дуже потужний компонент, що дозволяє будувати дво- і тривимірні діаграми на основі різних даних.

Він має велику кількість різноманітних властивостей. Частина з них, у свою чергу, є об'єктами і володіють власними властивостями. Являється контейнером об'єктів Series типу TChartSeries - серій даних, що характеризуються різними стилями відображення. Кожна серія буде відповідати одній кривій на графіку.

Розташований компонент на формі буде мати вигляд, як на Рис. 1.3.

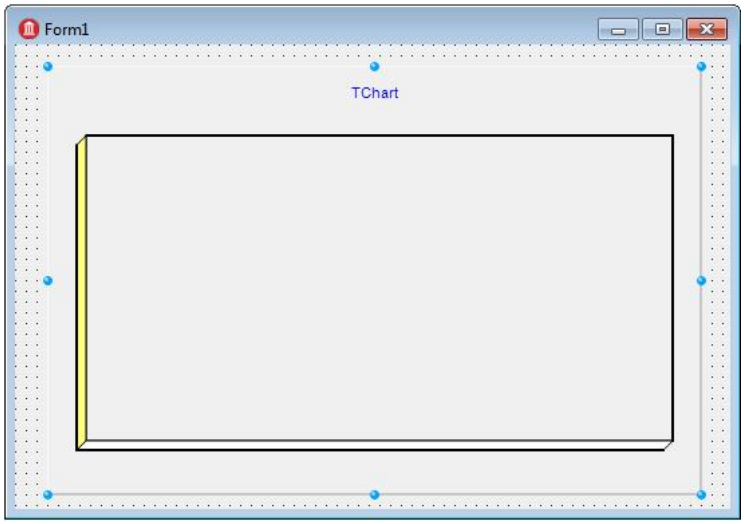


Рис. 1.3. Вигляд розташованого компоненту Chart на формі

Налаштування властивостей компонента TChart відбувається в редакторі Editing Chart (Рис. 1.4).

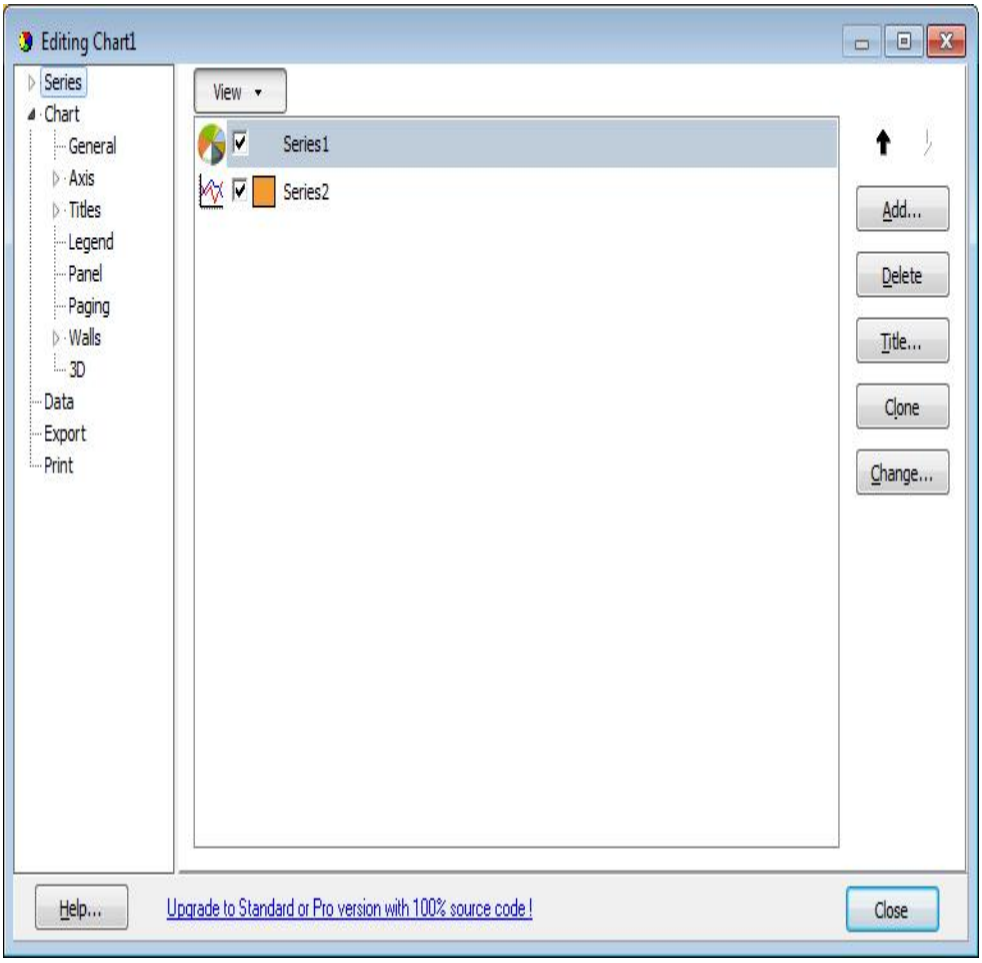


Рис. 1.4. Вигляд властивостей Editing Chart

Основні параметри діаграми визначаються на вкладці Chart (діаграма), вона, у свою чергу, складається з набору додаткових панелей.

Панель Ряд даних (Series) важлива, в неї можна додати на один графік кілька діаграм за допомогою кнопки Add. При цьому над значеннями даних можна виконувати операції, які задаються у вкладці Series -> Data Source вибравши function зі списку function.

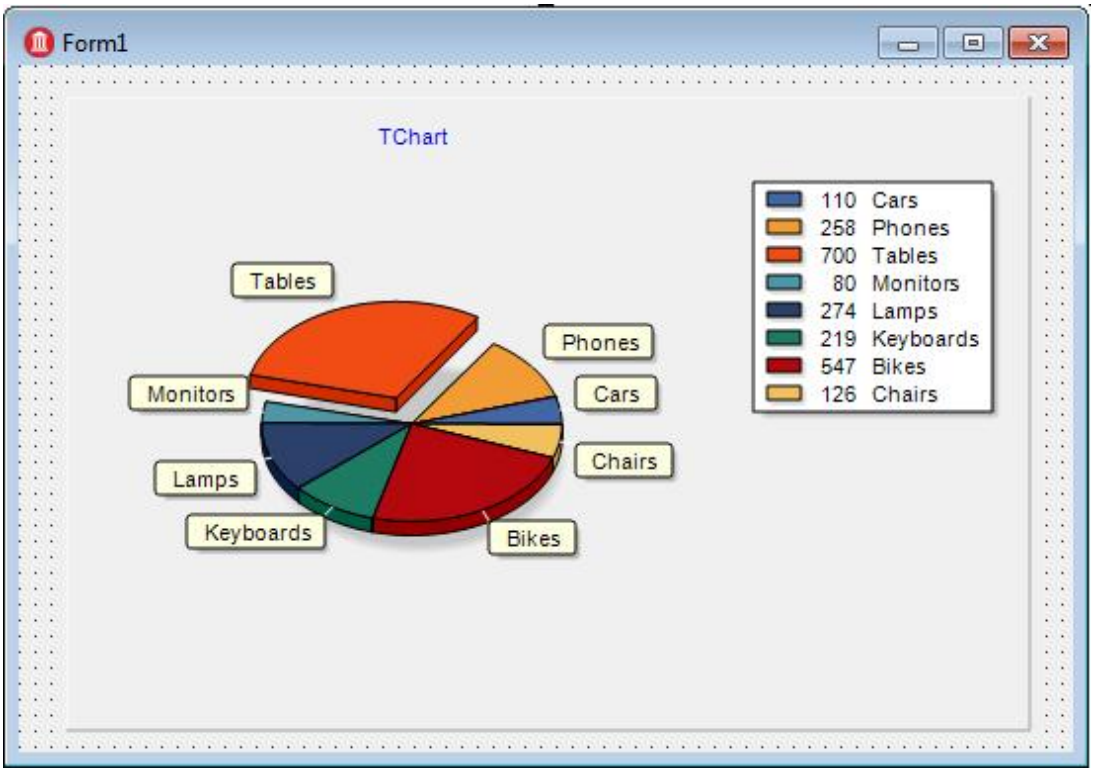


Рис. 1.5. Приклад діаграми, побудованої за допомогою об'єкта TChart

Приклад коду створення серії та додавання її до графіка приведений у лістингу (Додаток А).

```
procedure Series(const _Chart: TChart);
var
  LineSeries: TLineSeries;
  x: double;
  y: double;
  i: integer;   //counter
begin
  LineSeries:= TLineSeries.Create(_Chart);  //create series
```

```
  for i := 0 to 10 do begin
x := i;
y := i * i;
   LineSeries.AddXY(x, y);
  end;
  Chart.AddSeries(LineSeries);
end;
```

Візуальний інтерфейс програми розрахунку параметрів самоорганізованих структур представлено на Рис. 1. 6.

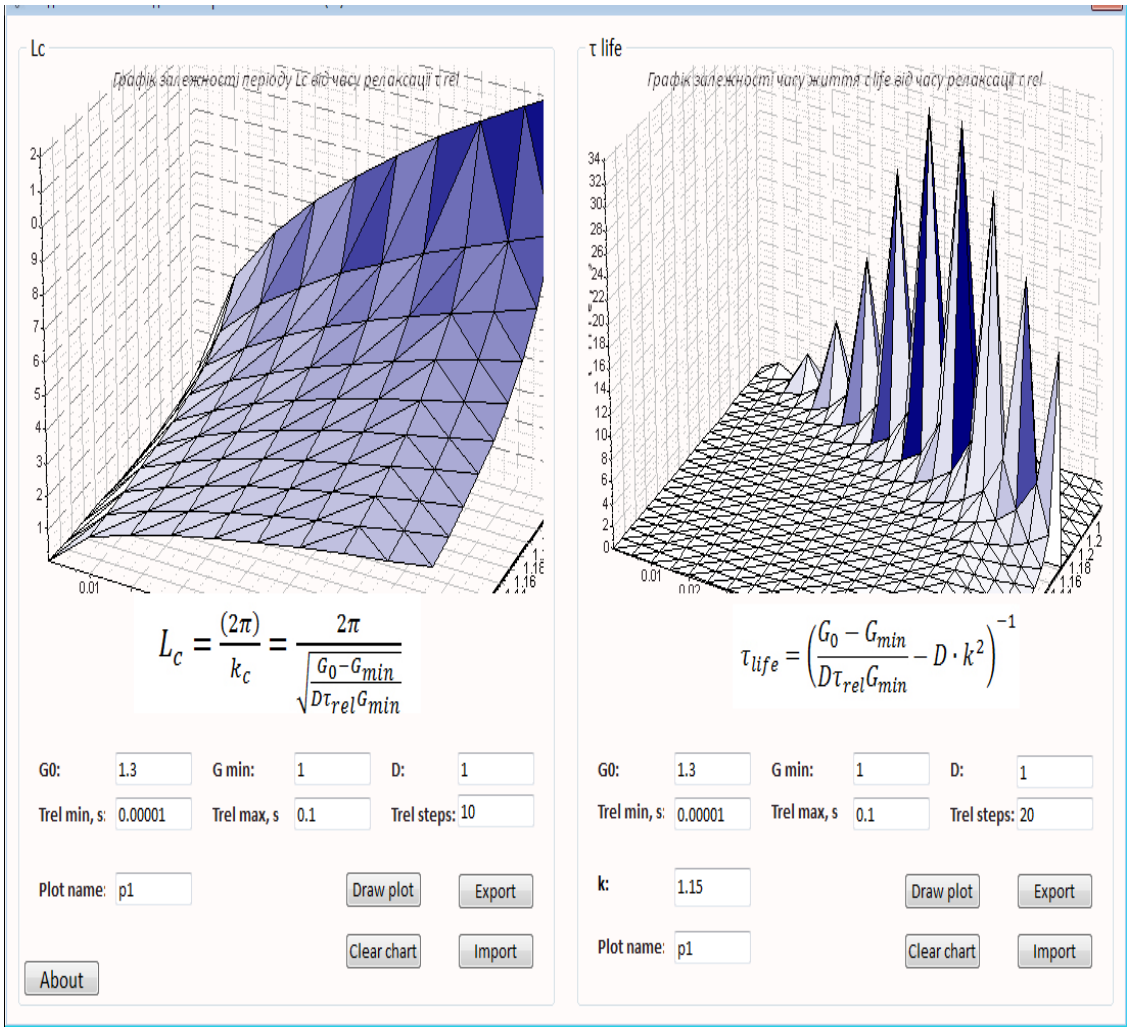


Рис. 1.6. Залежність періоду Lc та часу життя τ_{life} самоорганізованих структур від потужності лазерного опромінення (комп'ютерне моделювання в IDE Delphi, Додаток А)

1.2. Електронно-мікроскопічні дослідження структури некристалічних тіл систем *As-S(Se)* та їх моделювання в середовищі Delphi

1.2.1. Електронно-мікроскопічні дослідження структури некристалічних тіл систем *As-S(Se)*

Delphi — це інтегроване середовище швидкої розробки програмного забезпечення для роботи під Microsoft Windows (Yurkovych, Seben & Mar'yan, 2017). Воно підтримує розробку Windows-застосунків на мові програмування Delphi, яка є наступницею мови Object Pascal .

У сучасній науці і техніці напівпровідникових приладів поряд з кристалічними матеріалами широко використовують різні аморфні речовини. Суттєвою перевагою аморфних матеріалів над кристалічними є їх технологічність, тобто відносна простота й менша енергомісткість технологічного процесу, більша можливість зміни складу і властивостей цих речовин (Mar'yan & Yurkovych, 2019). Аморфні матеріали дають можливість конструювати прилади, які легко піддаються мікромініатюризації. Вони також легко поєднуються з кристалами в одному пристрої. Важливим стимулом розвитку досліджень аморфних речовин є також освоєння космічного простору, що потребує розробки і створення приладів для експлуатації в умовах тривалої дії космічного випромінювання, оскільки аморфні матеріали значно менш чутливі до проникаючої радіації, жорсткого рентгенівського випромінювання. Активність досліджень халькогенідів систем As-S(Se) зв'язана з унікальними властивостями цих речовин, що, разом з розробкою методів одержання тонких плівок великої площі, відкриває нові можливості їх практичних застосувань:

диски оптичної пам'яті з дуже великою щільністю запису інформації, електронні схеми великої площі на тонких гнучких підкладках, швидкодіючі і довговічні фоточутливі барабани для ксерокопіювання та фоторезистивні матеріали для літографії, оптичні елементи лазерної техніки, голограми та елементи інтегральної оптики, дешеві високочутливі термоелектричні перетворювачі та фотоелементи, перемикачі, фототермопластичні, фотоелектретні матеріали (Іваницький, 2011).

Для отримання однозначних, відтворюваних, достовірних результатів експериментальних досліджень структури аморфних плівок важливим є строгий контроль основних параметрів їх конденсації. Тому у процесі проведення всіх технологічних процесів слід враховувати такі найважливіші характеристики: хімічний склад та структурний стан вихідних речовин; вид випарника та його матеріал; температура випарника $T_в$; мас-спектрометричний склад парової фази; вид матеріалу підкладки; температура підкладки $T_{пд}$; віддаль від випарника до підкладки; кут конденсації пари; швидкість конденсації $v_к$; умови зберігання та препарування зразків для досліджень; вплив різних зовнішніх дій (Бобик, Іваницький, Ковтуненко,Сватюк, 2012; Mar'yan & Yurkovych, 2019).

Метод простого термовипаровування використовується лише для матеріалів, хімічний склад яких відтворюється в осаджених плівках. Як правило, це елементарні та деякі подвійні речовини, до яких відносяться і певні матеріали систем As-S(Se) (Іваницький, 2011). Для цього методу важливим є вибір належної конструкції термічних випарників.

Для аналізу структури аморфних речовин важливим є знання хімічного складу досліджуваних зразків. Особливого значення це набуває при електронномікроскопічних дифракційних дослідженнях з використанням методу функцій радіального розподілу атомів (ФРРА).

Контроль хімічного складу проводився на електронному мікроскопі - мікроаналізаторі ЕММА-4. При виконанні кількісних вимірювань використовувався режим хвильової дисперсії, оскільки при цьому досягається

найбільш висока роздільна здатність. Експериментальні значення інтенсивності аналітичних ліній визначалися з урахуванням фону. Остаточні результати РСМА отримували статистичною обробкою даних вимірювань.

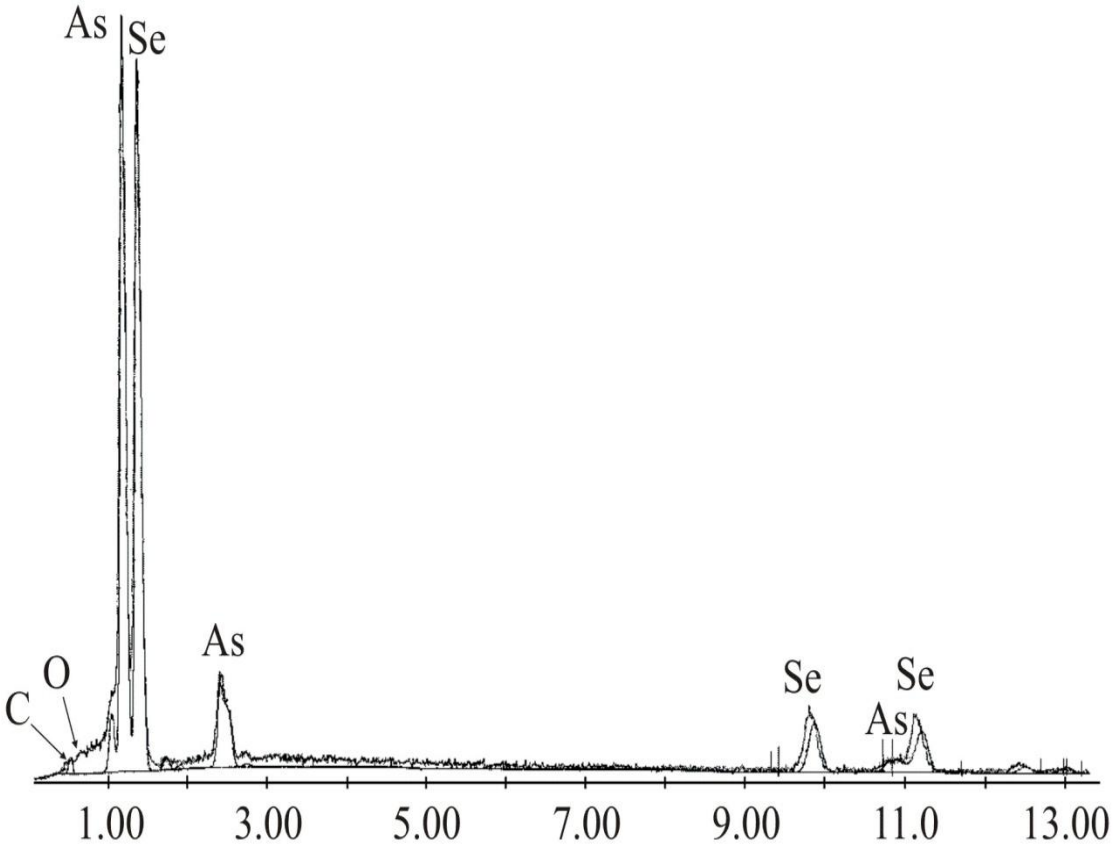


Рис.1.7. Типовий спектр РФЕС для плівок $As_{50}Se_{50}$, отриманих методом ДТВ

Широкі можливості при вирішенні структурних задач дає електронна мікроскопія. Для неї характерні: роздільна здатність на рівні окремих атомів, оперативність, можливість комплексного застосування різних методик, швидкість переходу від одного режиму спостереження до іншого, можливість моделювання різних зовнішніх впливів на зразок безпосередньо в колоні мікроскопа і спостереження при цьому стимульованих структурних змін “in situ” (у процесі дії) і т.д. Електронномікроскопічні дослідження дозволяють швидко і з достатнім ступенем вірогідності оцінити ряд важливих структурно-експлуатаційних параметрів тонкоплівкових систем. Істотним же обмеженням даного методу є товщина плівок: не більше за 100 – 200 нм.

Електронномікроскопічні дослідження в основному проводилися на приладах ЕММА-2, ЕММА-4 і ЕМВ-100Б при прискорюючих напругах 75 і

(або) 100 кВ. Результати досліджень фіксувались на фотопластинках або з допомогою цифрової фотокамери у виді електронномікроскопічних зображень (Іваницький, 2011).

Важлива додаткова інформація про структуру аморфних плівок вилучалася і в ході досліджень процесів їх кристалізації (Бобик, Іваницький, Ковтуненко,Сватюк, 2012). При ідентифікації кристалічних фаз, які виділялися у процесі кристалізації, використовувалися літературні дані про міжплощинні відстані різних кристалічних модифікацій елементарних речовин і їх сполук.

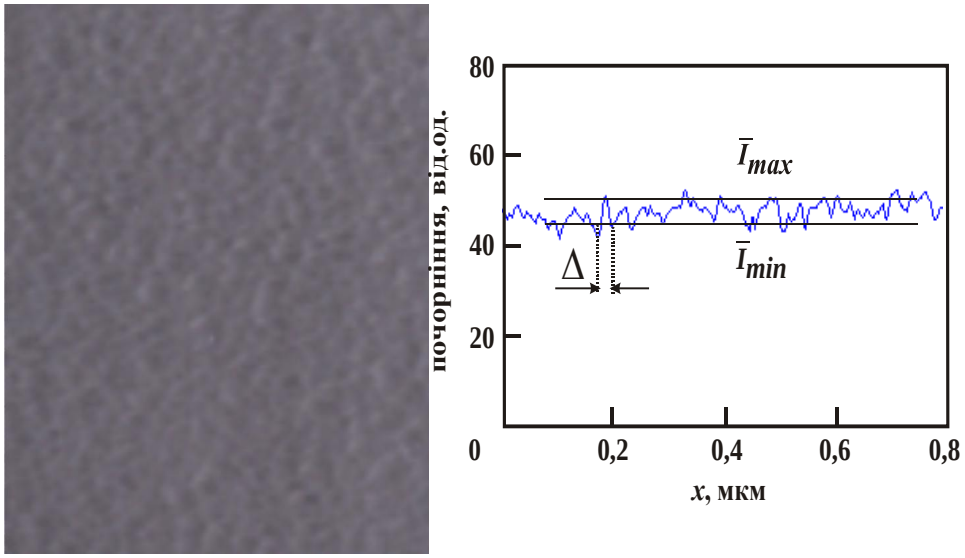


Рис.1.8. Промікрофотометрований електронномікроскопічний знімок «на просвіт» аморфної плівки $As_{33}Se_{67}$ та профіль розподілу фотографічного почорніння на ньому вздовж горизонтальної прямої

1.2.2. Комп'ютерне моделювання структури аморфних плівок

Для кількісного опису неоднорідностей контрасту ЕМ зображень ми використали методику статистичної обробки кривих фотометрування, запропоновану в роботі (Бобик, Іваницький, Ковтуненко,Сватюк, 2012). Базою для такої обробки є ЕМ знімки у вигляді цифрових файлів. Для їх аналізу у середрвищі Delphi була розроблена спеціальна програма одномірної розгортки

та моделювання наноструктурних параметрів аморфних речовин. Відповідно до алгоритму роботи програми компютер здійснює розгортка по координаті *x* або *y* двохмірного цифрового знімку. У результаті отримується крива розподілу інтенсивностей пікселів зображення вздовж *x* або *y* координати, яку називають мікрофотограмою. Шляхом статистичної обробки такої мікрофотограми визначаються кількісні параметри та моделюються наноструктурні характеристики аморфних обєктів.

Алгоритм роботи програми полягає в наступному.

На кожній мікрофотограмі вибирається досить протяжна частина, довжина якої *L* називається базовою довжиною. На цій частині розраховуються певні локальні параметри неоднорідностей ЕМ зображень. Базою для таких розрахунків є середня лінія мікрофотограми, яка визначається таким чином, щоб у межах базової довжини середньоквадратичне відхилення профілю мікрофотограми від даної прямої було мінімальним. У якості математичної умови проведення середньої лінії є рівність суми площ виступів сумі площ впадин профілю відносно середньої лінії (Іваницький, 2011).

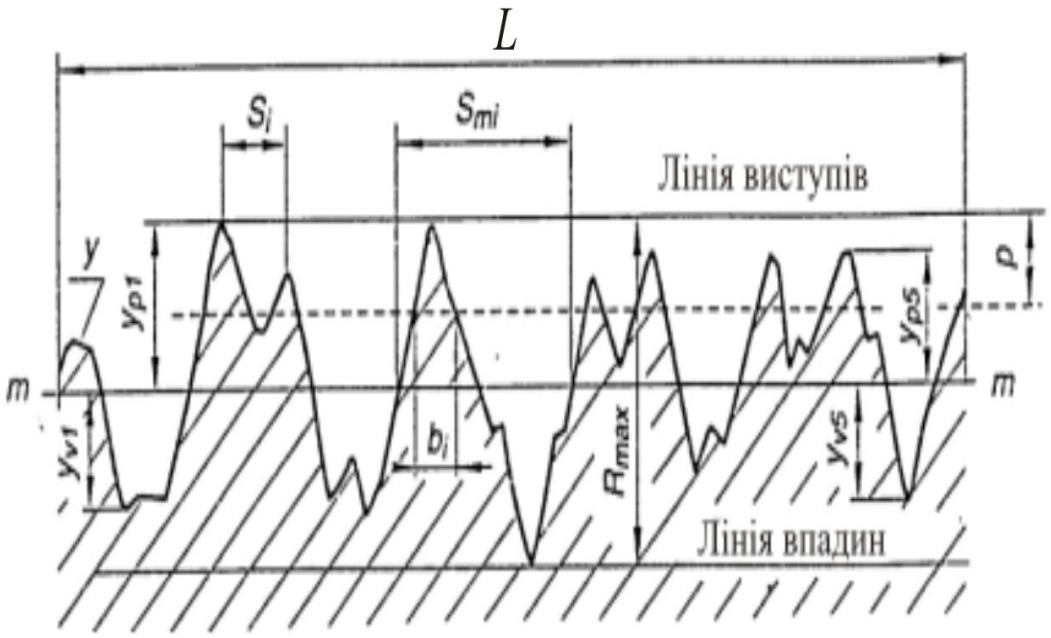


Рис.1.9. Параметри статистичної обробки окремих мікрофотограм ЕМ зображення

За наведеною методикою було проаналізовано топологічні особливості тонких аморфних плівок As-Se із нанонеоднорідною «зернистою» структурою. На Рис.1.8 наведено їх типові мікрофотограми для різних ступенів нанонеоднорідності та різних розмірів нанонеоднорідності d_c. Відповідно з цим рисунком, мікрофотограми плівок закономірно збільшують висоту своїх «осциляцій» при рості ступеня нанонеоднорідності *K*. Розміри ж піків осциляцій безпосередньо відображають зміни середнього діаметра нанонеоднорідностей плівок d_c при зміні їх хімічного складу та технологічних умов осадження.

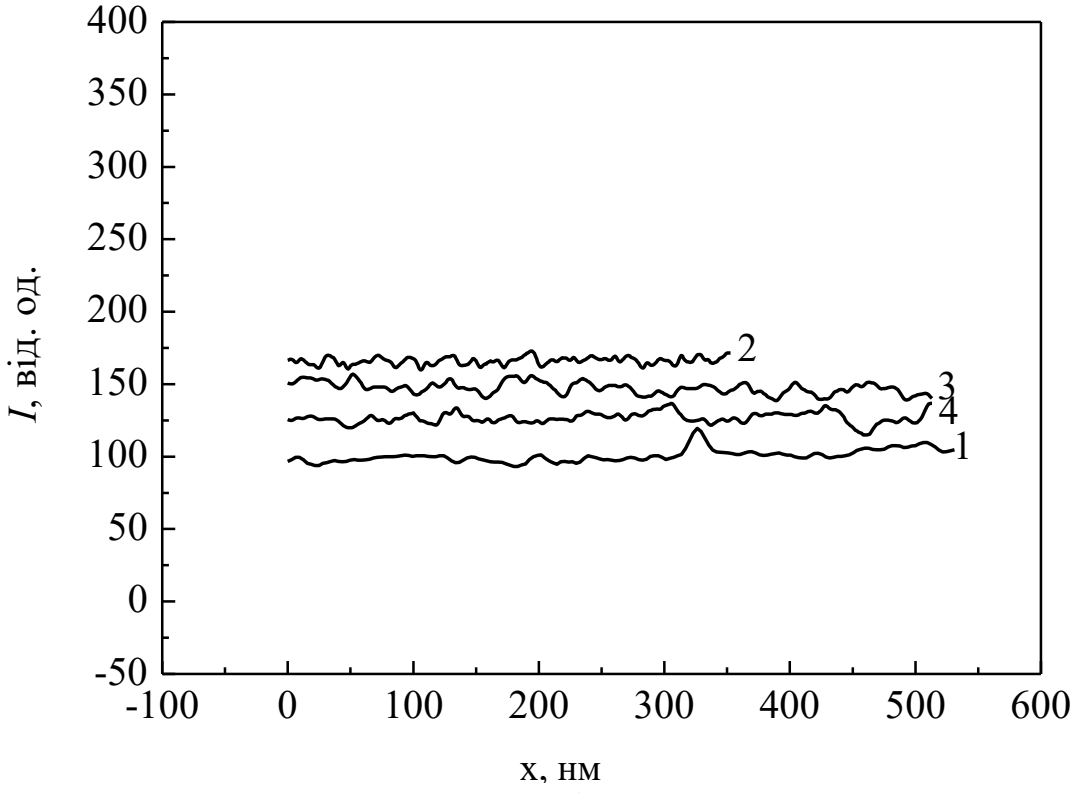


Рис. 1.10. Профілограми інтенсивності електронномікроскопічних зображень плівок $As_{20}Se_{80}$ з d_c = 30 нм (2), $As_{40}Se_{60}$ з d_c = 40 нм (4), $As_{50}Se_{50}$ з d_c = 50 нм (1), $As_{10}Se_{90}$ з d_c = 70 нм (4)

Головне меню програми для цифрової обробки та моделювання наноструктури аморфних плівок приведено на Рис. 1.11.

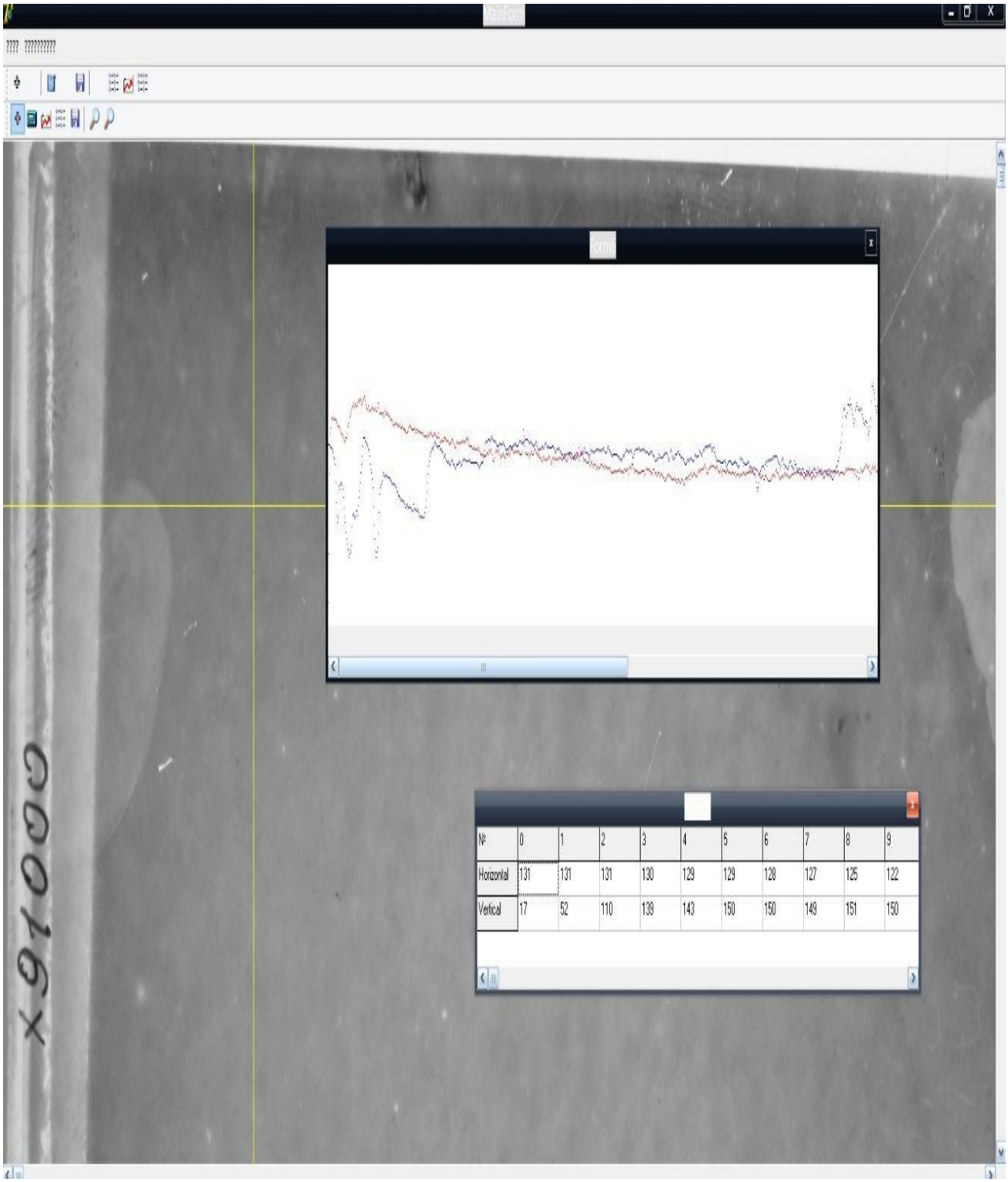


Рис. 1.11. Обробка електронно-мікроскопічних досліджень НКТ: візуальний інтерфейс програми

1.3. Явище перколяції: моделювання в середовищі Delphi з використанням компоненти Chart

Перколяція (percolation – англ.) в перекладі означає протікання. В літературі також можна зустріти і теорія перколяції, і теорія протікання і навіть теорія теорія просочування. Назва виникла в зв'язку з тим, що ряд перших робіт в цьому напрямку був присвячений процесам просочування (протікання) рідин та газів через пористе середовище. Теорія перколяції знаходить застосування в описі різних систем і явищ, в тому числі таких, як розповсюдження епідемій , надійність комп'ютерних та нейронних мереж, тощо. В фізиці модель перколяції часто використовують для моделювання явищ пробою в діалектриках, дифузії, процесів гелеутворення, переходу в некристалічний стан (Mar'yan & Yurkovych, 2019).

1.3.1. Коміркова перколяція

Найбільш поширеними задачами теорії перколяції являються коміркові задачі :

- ***задача вузлів***;
- ***задача зв'язків***.

Коміркові моделі в першу чергу представляють інтерес з теоретичної точки зору, саме для них доведено ряд строгих тверджень і відношень. На даний час процеси протікання на комірках достатньо добре вивчені. З іншого боку, ці задачі мають і практичне значення: навіть такої простої моделі достатньо, щоб описати, наприклад, фазовий перехід парамагнетик-феромагнетик.

Розглянемо нескінченну квадратну ґратку. Назвемо точки перетину ліній вузлами. Самі лінії будемо називати зв'язками. В задачі зв'язків шукають відповідь на питання: яку частину зв'язків потрібно прибрати (перерізати), щоб ґратка розпалась на дві частини? В задачі вузлів блокують вузли (прибирають вузол, тим самим перерізають всі вхідні у вузол "дефекту") і досліджують, при якій частці блокованих вузлів ґратка розпадеться (Рис 1.12). Зрозуміло, що квадратна ґратка являється тільки однією з можливих моделей. Можна розглядати перколяцію і на трикутній, шестикутній ґратках, тривимірних структурах, тощо.

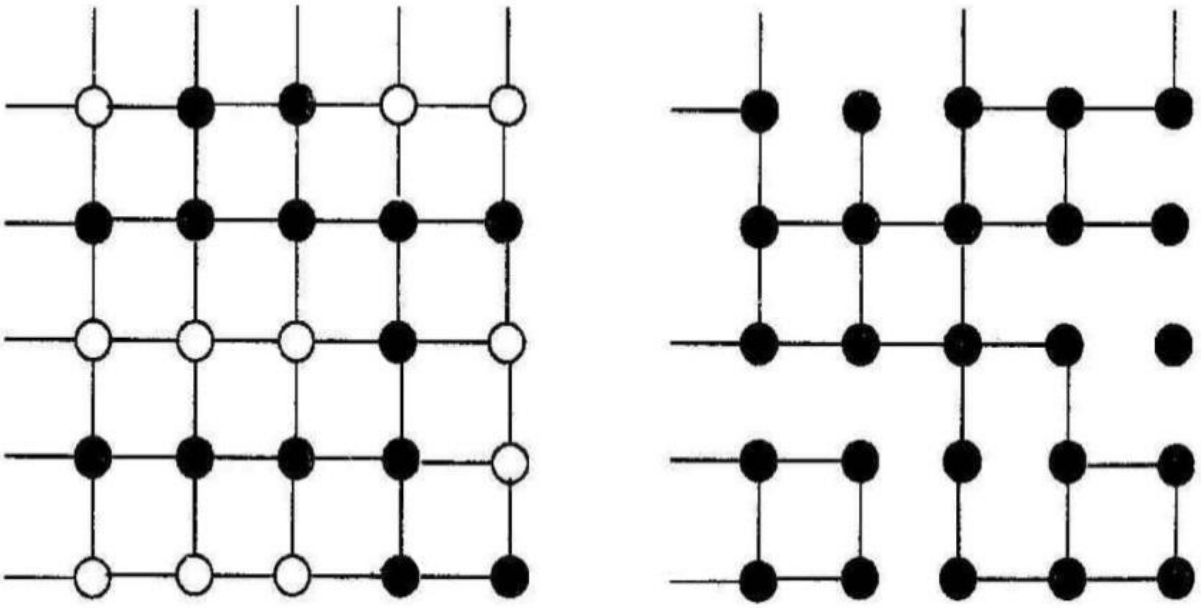


Рис. 1.12. Задача вузлів (зліва) і задача зв'язків (справа) на квадратній ґратці

Коли всі вузли (або всі зв'язки) закриті, ґратка є моделлю ізолятора. Коли вони всі відкриті і по провідним зв'язкам через відкриті вузли може поширюватись струм, ґратка буде моделлю метала. При якомусь критичному значенні відбудеться перколяційний перехід, що є геометричним аналогом переходу метал-ізолятор.

В комірковій перколяції двовимірна система розділена на комірки (чим більше комірок тим більша наближеність до реального об'єкта). В якості найпростішого прикладу можна розглянути модель протікання, наприклад, електричний пробій на двовимірній квадратній ґратці, що складається з вузлів, які можуть бути провідними або непровідними .

У початковий момент часу всі вузли ґратки є непровідними. З часом джерело замінює непровідні вузли на провідні, і число провідних вузлів поступово зростає. При цьому вузли заміщуються випадковим чином, тобто вибір кожного з вузлів для заміщення є рівноймовірним для всієї поверхні ґратки. Перколяцією називають момент появи такого стану ґратки, при якому існує хоча б один неперервний шлях через сусідні провідні вузли від одного краю до протилежного. Очевидно, що зі зростанням числа провідних вузлів, цей момент

настане раніше, ніж вся поверхня ґратки складатиметься виключно з провідних вузлів.

Позначимо непровідний і провідний стани вузлів нулями і одиницями відповідно. В двовимірному випадку фізичному середовищу буде відповідати бінарна матриця. Послідовність заміни нулів матриці на одиниці адекватна ідентифікації джерела протікання (Gould & Tobochnik, 1988).

У початковий момент часу матриця складається виключно із непровідних елементів (Рис. 1.13):

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Рис.1.13. Матриця коміркової перколяції у початковий момент часу

При впливі зовнішнього джерела в матриці починають додаватися провідні елементи, однак спочатку їх недостатньо для перколяції (Рис. 1.14):

0	0	0	1
1	0	1	0
0	0	1	0
0	0	1	0

Рис.1.14. Матриця коміркової перколяції після додавання провідних елементів

У міру збільшення числа провідних вузлів настає такий критичний момент, коли відбувається перколяція, як показано на Рис. 1.15:

0	0	0	1
1	1	0	0
0	1	1	0
0	0	1	1

Рис. 1.15. Момент настання протікання

Видно, що від лівої до правої границі останньої матриці є ланцюжок елементів, який забезпечує протікання струму по провідним вузлам (одиницям), які неперервно слідують одна за одною (Mar'yan, Seben & Yurkovych, 2018).

Більш наглядно це можна продемонструвати на наступному прикладі, який є подібним до попереднього: гратка складається з провідних елементів та ізоляторів (Рис. 1.16). До двох сторін ґратки припаяні металічні контакти, які приєднані до джерела живлення. При деякому критичному значенні кількості провідних елементів відбудеться перколяційний перехід і коло провідності замкнеться (Gould & Tobochnik, 1988; Mar'yan & Yurkovych, 2019).

Рис. 1.16. Перколяція на квадратній ґратці

1.3.2. Неперервна перколяція

Перколяція може спостерігатися як на ґратках, так і інших геометричних конструкціях, у тому числі неперервних, що складаються з великого числа подібних елементів або неперервних областей відповідно, які можуть знаходитися в одному з двох станів. Такі математичні моделі називаються неперервними або континуальними (Mar'yan & Yurkovych, 2019).

Як приклад перколяції в неперервному середовищі може виступати проходження рідини крізь об'ємний пористий зразок (наприклад, води через губку з піноутворюючого матеріалу), в якому відбувається поступове надування бульбашок до тих пір, поки їх розміри не стануть достатніми для просочування рідини від одного краю зразка до іншого[2] (Gould & Tobochnik, 1988).

[2] Явище перколяції має багато спільного з розглядуваними нами в розділах 3, 4 нейронними мережами та розробкою моделей самоорганізації в них

Інтуїтивно поняття перколяції переноситься на будь-які конструкції або матеріали, які називаються перколяційними середовищами. Для них має бути визначено зовнішнє джерело протікання, спосіб протікання, а також можливість їх елементів (фрагментів) перебувати в різних станах, один з яких (первинний) не задовільняє даному способу проходження, а інший (вторинний) задовільняє. Під способом протікання також розуміють певну послідовність виникнення елементів або зміну фрагментів середовища в потрібний для протікання стан, який забезпечується джерелом. Джерело ж поступово переводить елементи або фрагменти зразка з одного стану до іншого, поки не наступить момент перколяції.

Сукупність елементів, по яких відбувається перебудова, називається **перколяційним кластером**. В залежності від конкретної реалізації він може мати різну форму. Тому прийнято характеризувати його загальний розмір. **Порогом протікання** називається кількість елементів перколяційного кластера, віднесена до загальної кількості елементів аналізованого середовища.

Зважаючи на імовірнісний характер перемикань станів елементів середовища, в кінцевій системі чітко визначеного порога (розміру критичного кластера) не існує, але є так звана критична область значень, в яку потрапляють порогові значення перколяції, отримані в результаті різних випадкових реалізацій. Зі збільшенням розмірів системи область звужується в точку.

Не дивлячись на те, що розгляд коміркових перколяційних задач дуже зручний як для аналітичного, так і для чисельного вивчення, багато реальних невпорядкованих систем не мають ґраткової структури та потребують іншого підходу. В такому випадку доцільним буде розгляд моделі неперервної, або як її ще називають континуальної перколяції. Континуальна задача перколяції має різні формулювання, серед яких найбільш популярні наступні дві.

1) **Модель пустот** або **модель швейцарського сиру**. Сферичні пустоти (однакового розміру або які мають деякий розподіл за розмірами) випадковим чином поміщаються в середину провідного середовища. Сферичні пустоти можуть перекривати одна одну (Рис. 1.17). При критичній частці об’єму, занятого такими пустотами, з’являється нескінчений кластер з’єднаних одна з одною

пустот, і система втрачає провідні властивості чи механічну жорсткість. Ця модель зручна для опису еластичних властивостей матеріалів.

Рис. 1. 17. Неперервна перколяція для об’єктів однакового і різного радіусів та задача сфер для неперервної перколяції

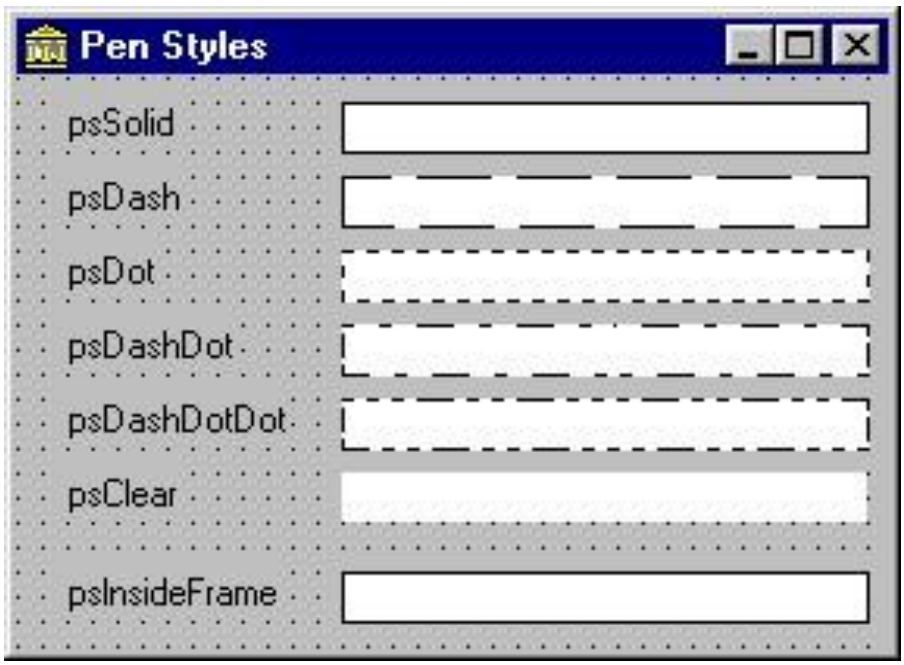
2) **Задача дисків, що перекриваються** або **модель випадкових вузлів**.

У цій задачі диски являються провідними і знаходяться в непровідному середовищі (Рис. 1.17). Можна ввести близькодіючу кореляцію між дисками – сили відштовхування між дисками, що приведе до зсуву порога перколяції, але залишить незмінними критичні показники. При деякому критичному об’ємі, занятому сферами які перекриваються, в системі виникає нескінчений провідний кластер. Модель використовується для опису стрибкової провідності в

домішкових напівпровідниках, фазових переходах в феромагнетиках (Gould & Tobochnik, 1988).

1.3.3. Графічні класи в середовищі Delphi: моделювання явища дифузії

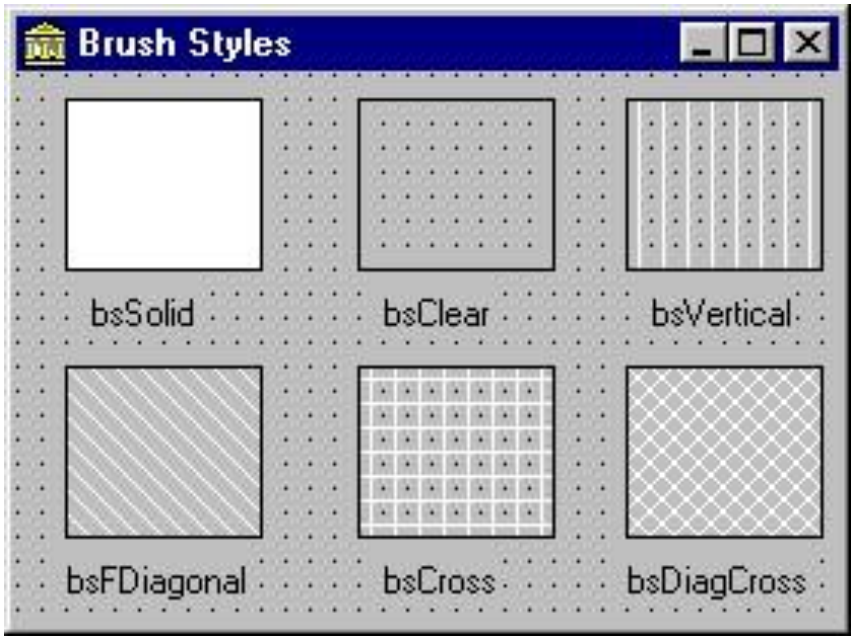
1) Клас **Pen** (перо)- призначений для сторвення ліній і границь геометричних фігур. Можна задавати такі властивості як колір, стиль а також товщину:
Колір : Pen.Colour:=ClBlack; - клас Pen буде виведено чорним кольором.
Тип : Pen.Style:=psSolid;
Товщина : Pen.Width:=2;



2) Клас **Font**(шрифт) - призначений для виводу тестової інформації.
Властивості (приклади) :
Колір: Font.Colour:=ClRed;
Розмір: Font.Size:=10;
Ім'я шрифта: Font.Name:="Arial";
Тип: Font.Style:=fsbold;
3) Клас **Brush** (пензлик)- призначений для зафарбовування і штриховки фігур.
Властивості(приклади)

1) **Колір:**Brush.Color:=ClGreen;

2) **Тип:** Brush.Style:=bsSolid;

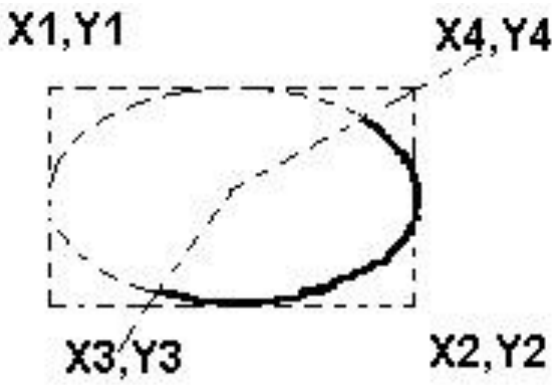


Клас **Canvas** (полотно) містить властивосі класів **Pen, Font, Brush** і методи, які виконують побудову різного роду геометричних об'єктів (дуги,лінії, еліпси, прямокутники, кола і тп.)

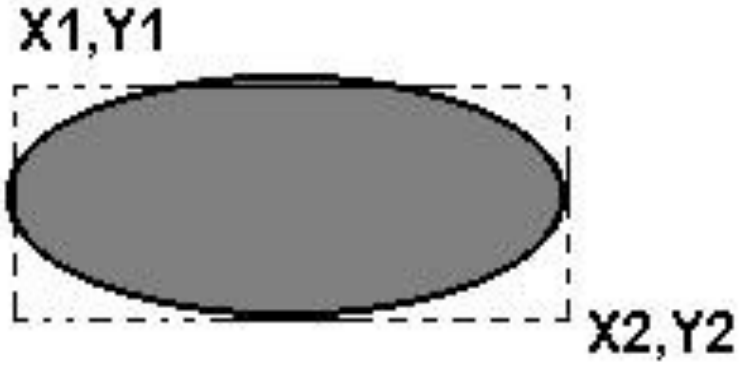
MoveTo(X, Y: Integer)-перехід до точки **X, Y**

LineTo(X+dX, Y+dY : Integer) - побудова лынії з кординатами зміщеними на **X+dX, Y+dY**

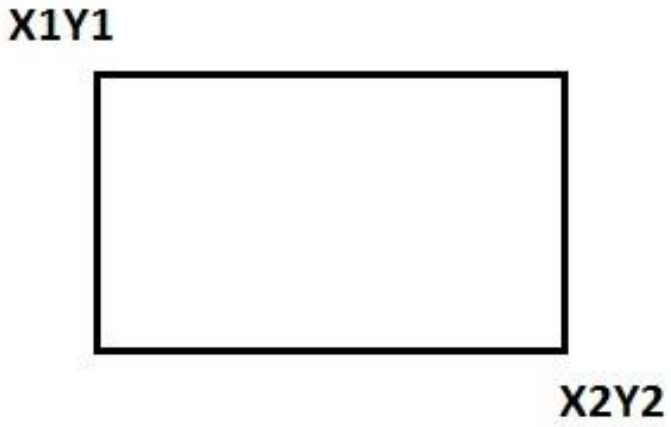
Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Integer) - малює дугу еліпса, вписаного в прямокутник з координатами **(X1, Y1) і (X2, Y2).** Дуга визначається двома радіусами еліпса, проходячими через точки **(X3, Y3) і (X4, Y4).** Дуга малюється проти годинникової стрілки від точки перетину еліпса з першим радіусом до точки перетину з другим радіусом :



Ellipse (X1, Y1, X2, Y2: Integer) - малює еліпс, вписаний в прямокутник з лівим верхнім кутом в точці **(X1, Y1**) і нижнім правим кутом в точці **(X2, Y2).** Прямокутник малюється поточними пером і пензлем:



Rectangle (X1, Y1, X2, Y2: Integer) - малює прямокутник з лівим верхнім кутом в точці **(X1, Y1**) і нижнім правим кутом в точці **(X2, Y2) :**



Доповнити програму можна таким чином, щоб вона моделювала явище дифузії в некристалічних тілах. Для цього ввести додатково перколяційні елементи прямокуної форми, концентрація яких буде в два рази більша ніж елементів -дисків (Рис.1.18).

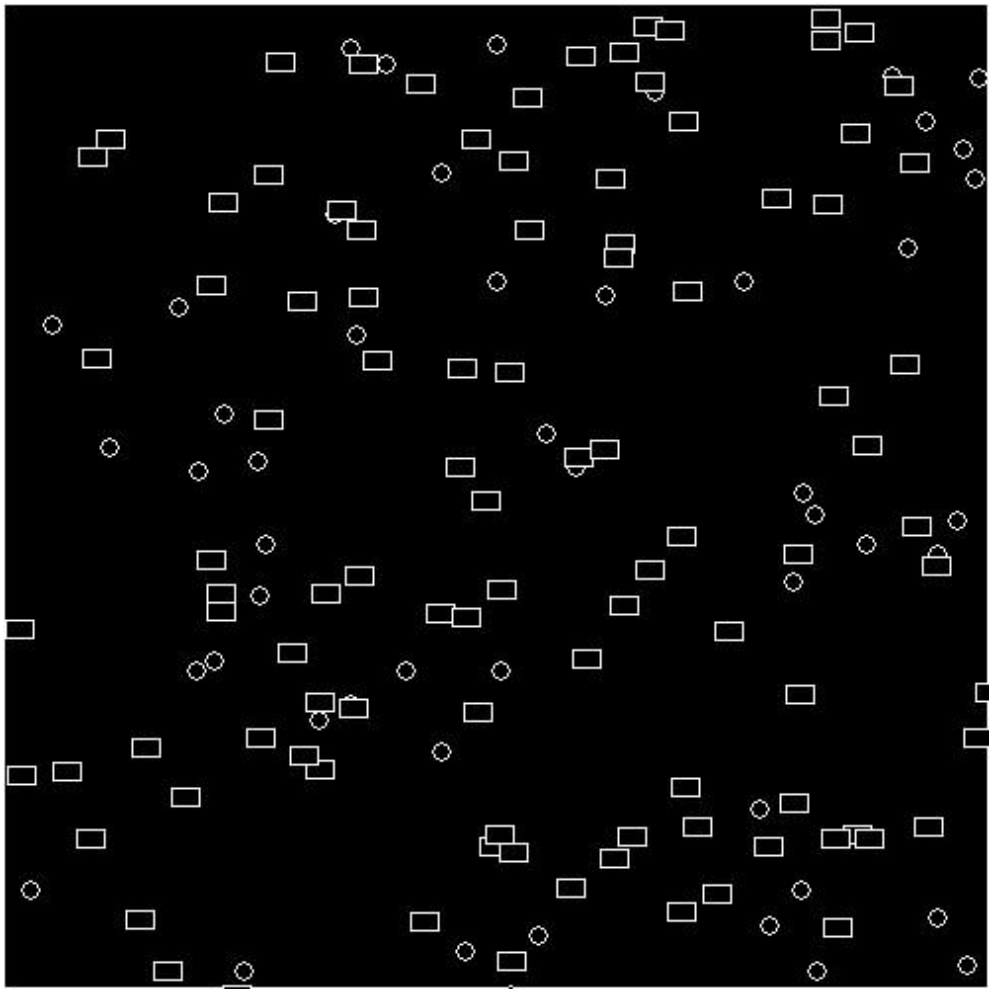


Рис.1.18. Моделювання явища дифузії

За допомгою компоненти **Chart** можна побудувати графік залежності імовірності[3] утворення перколяційного кластера **P** [4] від кількості елементів **N**.

[3] Імовірність утворення перколяційного кластера **P** визначається таким чином: в скількох із 10 випробувань для даної кількості елементів **N** утвориться перколяційний кластер. Наприклад: якщо в 4 із 10 то : P = 4/10 = 0.4 тобто 40%

[4] Перколяційний кластер - ланцюжок подібних елементів, що перекриваються або мають спільну сторону, яка неперервно проходить з одного кінця досліжуваного середовища до іншого

Розглянута модель дає змогу отримати оцінку впливу умов синтезу на коефіцієнт дифузії. Процес дифузії в методі Монте-Карло характеризується співвідношенням (Mar'yan & Yurkovych, 2019):

$$< \overline{R}^2(t) > \approx 2 \cdot D_a \cdot d \cdot t \,. \qquad (1.9)$$

Вираз (1.9) зв'язує між собою час t та приведене середньоквадратичне зміщення $< \overline{R}^2(t) >$ до міжатомної відстані a_0 з початковим положенням при $t = 0$, d - розмірність простору, $D_a = D / a_0^{\,2}$, D – коефіцієнт дифузії. Результати розрахунку коефіцієнта дифузії методом Монте-Карло за час (кількість кроків) $t = 10^3 \div 10^5 c$ (D_a визначається як границя при $t \to \infty$, $a_0 \approx 3 \cdot 10^{-8} см$), показують, що коефіцієнт дифузії некристалічних тіл $D = (0.135 \div 0.185) \cdot 10^{-15} см^2 / сек$.

ІІ. СИНЕРГЕТИКА ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ СИСТЕМ

2.1. Нейронні мережі як інформаційно-комунікаційні системи

Інформація формує та ідентифікує міру визначеності та завершеності системи, яка проявляється через фізико-хімічні та комунікаційні зв'язки. Слово інформація походить від латинського слова ***informatio*** і в перекладі з латинської мови означає відомості, стан. **Комунікація** – це один із способів поширення інформації в різноманітних середовищах з формуванням зворотних сигнально-кодових зв'язків та створенням нових інформаційних середовищ. Слово комунікація походить від грецького слова *επικοινωνία* та визначає трансформацію, поширення. **Система** – це самодостатня, цілісна форма організації структури. Слово «система» має грецьке походження *συστήματος* й означає «ціле, утворене з окремих частин». Інформація присутня на всіх рівнях, завдяки чому і забезпечується комунікація та цілісність системи (Рис. 2.1).

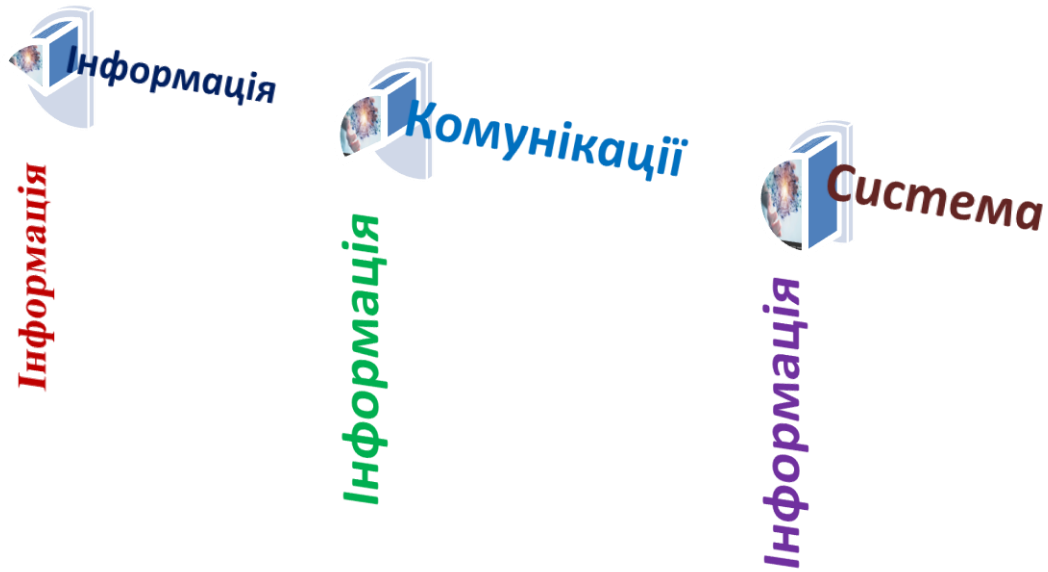


Рис. 2.1. Інформаційно-комунікаційна система

Нейрон та нейронні мережі виступають як продукт інформації та її поширення. З цієї точки зору вони співвимірні та когерентні з інформаційно-комунікаційними системами. Нейрон містить аксон (сприймання інформації), синаптичні зв'язки (поширення інформації), сому (операційна система обробки інформації) та дендрити (поширення та розвиток інформації). Середовище, в якому функціонують нейрон та нейронна мережа, є інформаційним (Рис. 2.2), завдяки чому забезпечується його структурування та системність (Mar'yan & Yurkovych, 2018).

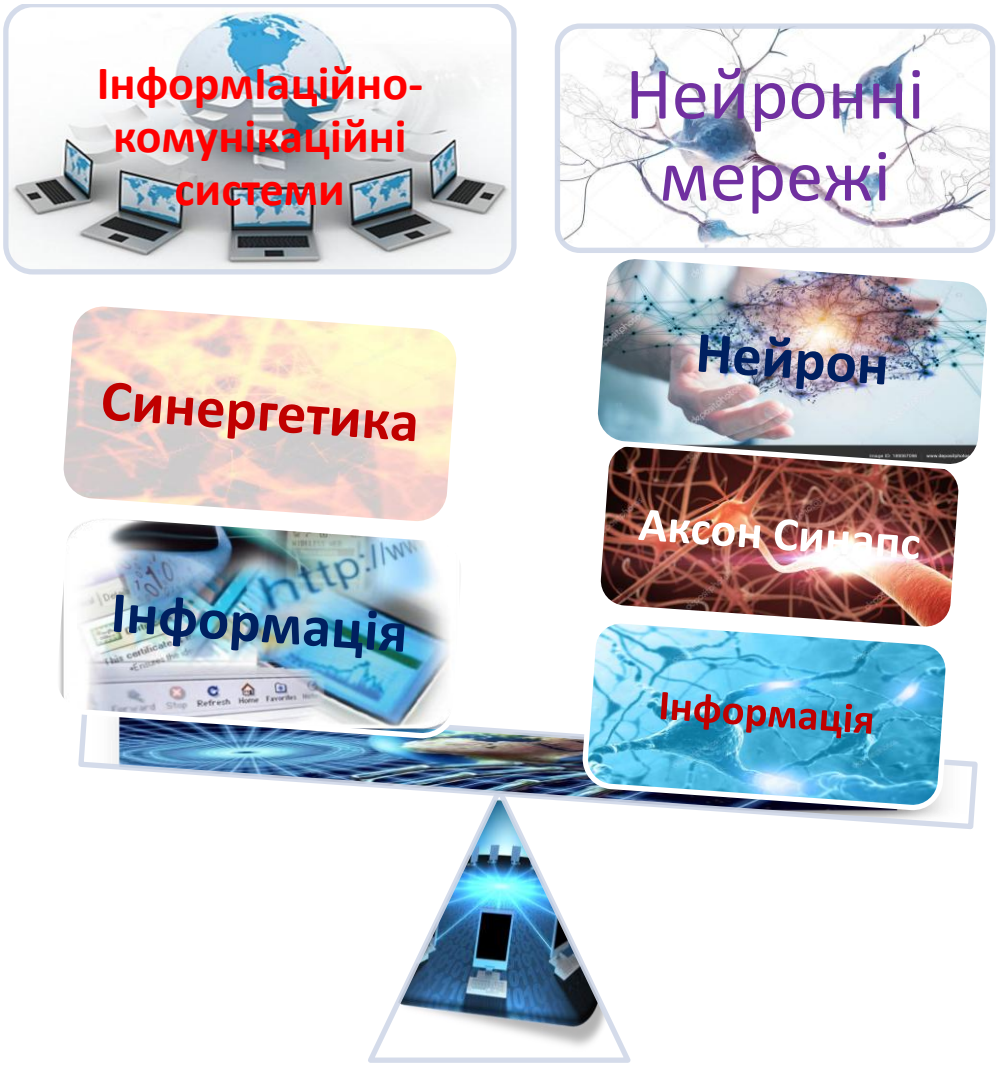


Рис. 2.2. Нейронні мережі як інформаційно-комунікаційні системи

2.2. Обєктно-орієнтоване моделювання та програмування

Розв'язок будь-якої задачі (фізичної, математичної, економічної, екологічної, освітньої та програмної зокрема) вимагає певного рівня абстрагування та абстракції (Mar'yan, Seben & Yurkovych, 2018). Унікальність синергетичного підходу проявляється в тому, що інформаційні технології, зокрема ООП, HTML функціонують не тільки в середовищах програмування або при розробці Web-сторінок. Вони реалізуються в оточуючому інформаційному середовищі при наукових дослідженнях, при вирішенні освітніх, екологічних, економічних проблем (Mar'yan & Yurkovych, 2019). Синергетика при цьому стирає грані, які обмежують систему та створюють деструкцію, реалізує умови функціонування негетропійних середовищ та реальності (Рис. 2.3).

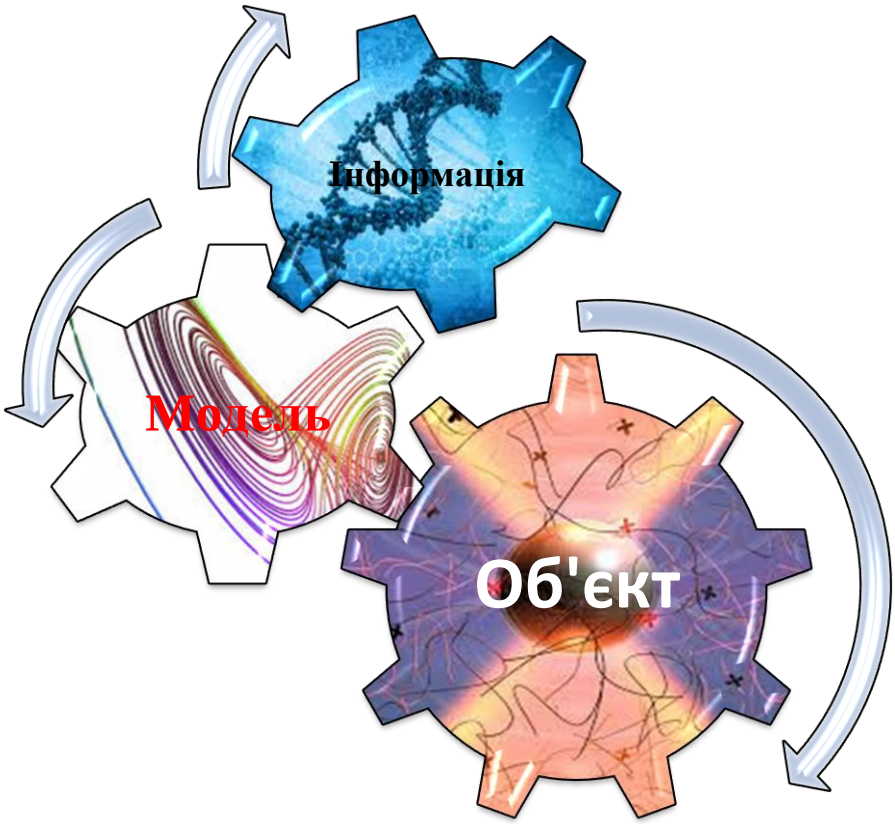


Рис.2. 3 . Об'єктно-орієнтоване моделювання та інформація

Об'єктно-орієнтоване програмування (ООП) — одна з парадигм сучасного програмування, яка розглядає програму як множину “об'єктів”, що взаємодіють між собою. Основу ООП складають три основні концепції: інкапсуляція, успадкування та поліморфізм (Рис. 2.4). Мови програмування високого рівня C++, Java, Object Pascal, Object Lisp, Visual Basic дають змогу піднятись на більш високий рівень абстракції порівняно з машинними кодами, які є основою мови асемблер. Ця можливість пов’язана з тим, що всі мови програмування високого рівня дозволяють окрім використання стандартних типів даних, створювати свої власні типи даних, які б найбільш повно відповідали характеру задачі. Окрім того, принцип модульності дозволяє програмну реалізацію задачі формулювати в термінах незалежних модулів, які використовують певну структуру даних задачі.

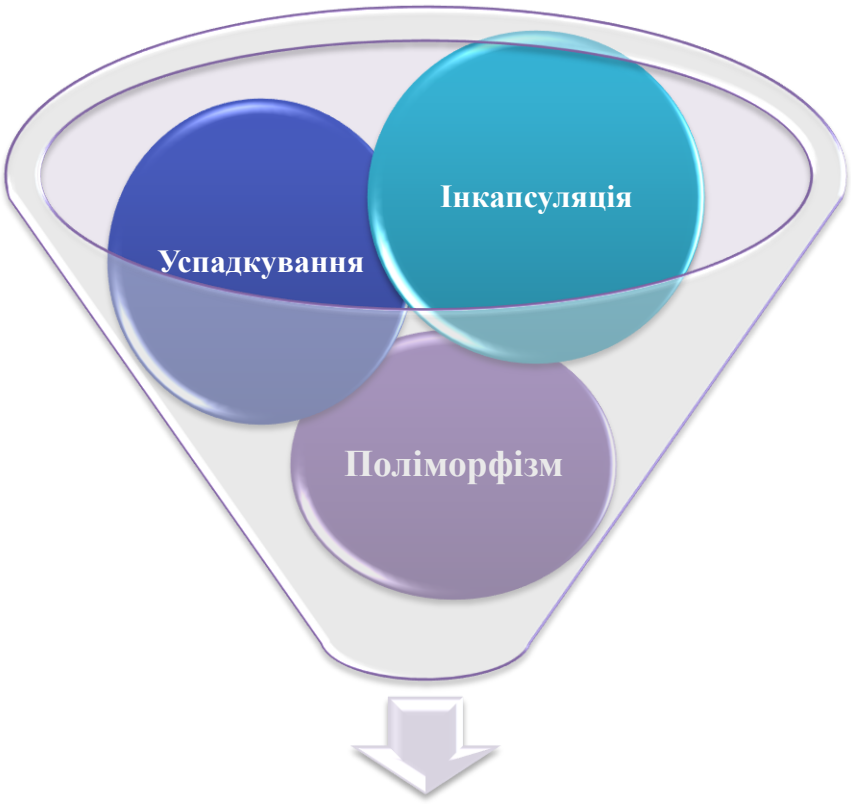


Рис. 2.4. Принципи об’єктно-орієнтованого програмування та моделювання

Принципи структурного програмування є основою для організації, з одного боку, оптимальної структури програми, а з іншого боку, для конструювання необхідної структури даних для збереження і обробки інформації в програмі. Сучасна концепція програмування - об'єктно-орієнтованим програмування (ООП), визначається як технологія створення комплексного програмного забезпечення на основі подання програми у вигляді сукупності об'єктів, кожен з яких є екземпляром певного типу (класу), а класи утворюють ієрархію зі спадкуванням властивостей.Таким чином, кожен об'єкт може містити не лише дані, які характеризують його стан, а і методи їх обробки (функції), які демонструють поведінку об'єкту. Об'єктно-орієнтований підхід заснований на тому, що будь-який елемент оточуючого нас світу можна вважати об'єктом. Наприклад, на рівні об'єктної абстракції можна розглядати довільні математичні поняття. Скажімо, комплексне число можна визначити як об'єкт, який містить дані – значення дійсної та уявної частин комплексного числа, і визначає методи – допустимі дії над такими об'єктами (обчислення модуля та аргументу, функції з комплексного числа, тощо). Довільний технічний прилад також можна вважати об'єктом. Наприклад, телевізор дозволяє сприймати зображення та звук (це інформація, або дані), а також виконувати певні дії – перемикати канали, змінювати гучність, яскравість (ці дії є методами даного об'єкту). Останній приклад демонструє також захист вмісту об'єкту, адже він використовується як "чорна скринька" – об'єкт лише реагує на зовнішні повідомлення (запити): перемкнути канал, збільшити контрастність, вимкнутись. При цьому користувач навіть не повинен знати, як саме (на технічному рівні) обробляються його запити. В цьому і полягає концепція модульності програми, а саме застосування видимої (інтерфейсної) частини, доступної основному проекту, та невидимої (імплементаційної) частини реалізації алгоритму.

Переваги ООП:

- *інтуїтивна близькість до довільної предметної області;*
- *моделювання як завгодно складних предметних областей;*

- *подієва орієнтованість.*

При об'єктно-орієнтованому підході програма являє собою опис об'єктів, їх властивостей (або атрибутів), сукупностей (або класів), відносин між ними, способів їх взаємодії та операцій над об'єктами (або методів). Безперечною перевагою даного підходу є концептуальна близькість до предметної області довільної структури та призначення. Механізм спадкоємства атрибутів і методів дозволяє будувати похідні поняття на основі базових і таким чином створити комплексну модель як завгодно складної предметної області з заданими властивостями (Mar'yan & Yurkovych, 2019).

Спадкування – це процес, шляхом якого деякий клас може успадковувати властивості та методи деякого існуючого класу, додаючи до них деякі особисті риси. Спадкування використовується у випадку, коли треба створити клас з функціональністю, схожою на функціональність вже існуючого класу. При успадкуванні утворюється клон існуючого (батьківського, базового) класу, який є коренем ієрархії успадкування, і цей клон (нащадок) відповідним чином модифікується. Ці два класи можуть мати спільні характеристики і поведінку, але один з них (нащадок) може мати більше властивостей і обробляти більше повідомлень (або обробляти їх інакше). У одного батьківського класу може бути декілька нащадків, при цьому батьківський (базовий) клас містить всі властивості і методи, спільні для його нащадків.

Розглянемо застосування об'єктно-орієнтованого моделювання до розробки нейронних мереж, побудованих на алгоритмах самоорганізації.

ІІІ. НЕЙРОННІ МЕРЕЖІ: ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ

Інтелектуальні системи на основі штучних нейронних мереж дозволяють з успіхом вирішувати проблеми розпізнавання образів, виконання прогнозів, оптимізації, асоціативної пам'яті і керування. Традиційні підходи до рішення цих проблем не завжди надають необхідну гнучкість. Багато додатків виграють від використання нейромереж (Аксенов, Новосельцев, 2006).

Штучні нейромережі є електронними моделями нейронної структури мозку, який головним чином навчається з досвіду. Природній аналог доводить, що множина проблем, які поки що не підвладні розв'язуванню наявними комп'ютерами, можуть бути ефективно вирішені блоками нейромереж. Тривалий період еволюції додав мозку людини багато якостей, що відсутні в сучасних комп'ютерах з архітектурою фон Неймана. **До них відносяться**:

- *розподілене представлення інформації і паралельні обчислення;*
- *здатність до навчання і здатність до узагальнення;*
- *адаптивність;*
- *толерантність до помилок;*
- *низьке енергоспоживання.*

Прилади, побудовані на принципах біологічних нейронів мають перелічені характеристики, що можна вважати суттєвим здобутком у індустрії обробки даних. Досягнення в галузі нейрофізіології надають початкове розуміння механізму природного мислення, де збереження інформації відбувається у вигляді образів, деякі з яких є складними (Ежов, Шумский, 1998). Процес зберігання інформації як образів, використання образів і вирішення поставленої проблеми визначають нову галузь в обробці даних, яка не використовуючи традиційного програмування, забезпечує створення паралельних мереж та їх навчання. В лексиконі розробників та користувачів нейромереж присутні слова, дуже відмінні від традиційної обробки даних,

зокрема, "вести себе", "реагувати", "самоорганізовувати", "навчати", "узагальнювати" та "забувати".

3.1. Нейронні мережі: принципи самоорганізації

3.1.1 Історія нейронних мереж: аналогія з мозком

Вивченню людського мозку - тисячі років. З появою сучасної електроніки, почались спроби апаратного відтворення процесу мислення. Перший крок був зроблений у 1943 р. З виходом статті нейрофізіолога **Уоррена Маккалоха (*Warren McCulloch*)** і математика **Уолтера Піттса (*Walter Pitts*)** про роботу штучних нейронів і представлення моделі нейронної мережі на електричних схемах (Аксенов, Новосельцев, 2006).

В 1949 р. опублікована книга **Дональда Хебба (*Donald Hebb*)** "Організація поведінки". В ній досліджена проблематика налаштування синаптичних зв'язків.

В 1950-х рр. з'являються програмні моделі штучних нейромереж. Перші роботи провів **Натаніел Рочестер (*Nathanial Rochester*)** з дослідної лабораторії IBM. І хоча пізніші реалізації були успішними, його модель зазнала невдачі, оскільки бурхливий зріст традиційних обчислень залишив у затінку нейронні дослідження.

В 1956 р. Дартмутський дослідний проект з штучного інтелекту забезпечив підйом досліджень нейромереж, зокрема штучних нейронних мереж. Стимулювання досліджень штучного інтелекту розгалузилось в двох напрямках: промислові застосування систем штучного інтелекту (експертні системи) та моделювання мозку (Вороновский, Махотило, Петрашев, Сергеев, 1997).

В 1958 р. **Джон фон Нейман (*John fon Neumann*)** запропонував імітацію простих функцій нейронів з використанням телеграфної передачі або вакуумних трубок. В 1959 р. **Бернард Відров (*Bernard Widrow*)** та **Марсіан Хофф (*Marcian Hoff*)** розробили моделі *ADALINE* та *MADALINE* (Множинні

Адаптивні Лінійні Елементи (*Multiple ADAptive LINear Elements*)). *MADALINE* діяла, як адаптивний фільтр, що усував відлуння на телефонних лініях. Ця нейромережа досі наявна в комерційному використанні (Ясницкий, 2008).

Нейробіолог **Френк Розенблатт (*Frank Rosenblatt*)** почав роботу над перцептроном. Одношаровий перцептрон був збудований апаратно і вважається класичною нейромережею. На той час перцептрон використовувався у класифікації множини вхідних сигналів в один з двох класів (Ясницкий, 2008).

У 1982 р. відновлення інтересу спричинило декілька подій. **Джон Хопфілд (*John Hopfield*)** представив статтю до національної Академії Наук США. Підхід Хопфілда створював докорінно нові підходи до моделювання.

У той самий час у Кіото (Японія) відбулась Об'єднана американо-японська конференція по нейронних мережах, які оголосили досягненням п'ятої генерації. Американські періодичні видання підняли цю історію, акцентуючи, що США можуть залишитись позаду, що привело до зросту фінансування в галузі нейромереж. З 1985 р. Американський Інститут Фізики розпочав щорічні зустрічі - "Нейронні мережі для обчислень". В 1989 р. на зустрічі "Нейронні мережі для оборони" **Бернард Відров** повідомив аудиторії про початок четвертої світової війни, де полем бою є світові ринки та виробництва (Ежов, Шумский, 1998; Хайкин, 2006).

У 2000 р. перехід на субмікронні та нанотехнології, а також успіхи молекулярної та бімолекулярної технології призводять до принципово нових архітектурних і технологічних рішень щодо створення нейрокомп'ютерів (Хайкин, 2006).

Сьогодні, обговорення нейронних мереж відбуваються скрізь. Перспектива їх використання видається досить яскравою, в світлі вирішення нетрадиційних проблем і є ключем до цілої технології[5]. На даний час більшість розробок нейронних мереж принципово працюючі, але можуть існувати процесорні обмеження. Дослідження скеровані на програмні та апаратні

[5] На сучасному етапі актуальність застосування нейронних мереж підтверджена розробками та впровадженнями відомих фірм Google, Tesla, Samsung, LG та цілого ряду інших

реалізації нейромереж. Компанії працюють над створенням трьох типів нейрочіпів: цифрових, аналогових та оптичних, що обіцяють бути хвилею близького майбутнього (Ежов, Шумский, 1998).

Точна робота мозку людини - все ще таємниця. Проте деякі аспекти цього дивовижного процесора відомі. Базовим елементом мозку людини є специфічні клітини, відомі як нейрони, що здатні запам'ятовувати, думати і застосовувати попередній досвід до кожної дії, що докорінно відрізняє їх від решта клітин тіла.

Кора головного мозку людини є протяжною, утвореною нейронами поверхнею товщиною від 2 до 3 мм і з площею близько 2200 $см^2$, що вдвічі перевищує площу поверхні стандартної клавіатури. Кора головного мозку містить близько 10^{11} нейронів, що приблизно дорівнює числу зірок Чумацького шляху. Кожен нейрон зв'язаний з 10^3 - 10^4 іншими нейронами. У цілому мозок людини містить приблизно від 10^{14} до 10^{15} взаємозв'язків (Хайкин, 2006).

Сила людського розуму залежить від числа базових компонент, різноманіття з'єднань між ними, а також від генетичного програмування і навчання (Вороновский, Махотило, Петрашев, Сергеев, 1997).

Індивідуальний нейрон є складним, має свої складові, підсистеми та механізми керування і передає інформацію через велику кількість електрохімічних зв'язків. Налічують біля сотні різних класів нейронів. Разом нейрони та з'єднання між ними формують недвійковий, нестійкий та несинхронний процес, що різниться від процесу обчислень традиційних комп'ютерів. Штучні нейромережі моделюють лише найголовніші елементи складного мозку, що надихає науковців та розробників до нових шляхів розв'язування проблеми.

3.1.2. Біологічний та штучний нейрони

Біологічний нейрон. Нейрон (нервова клітка) складається з тіла клітини - соми (**soma**), і двох типів зовнішніх деревоподібних відгалужень: аксона

(***axon***) і дендритів (***dendrites)***. Тіло клітини вміщує ядро (***nucleus***), що містить інформацію про властивості нейрону, і плазму, яка продукує необхідні для нейрону матеріали. Нейрон отримує сигнали (імпульси) від інших нейронів через дендрити (приймачі) і передає сигнали, згенеровані тілом клітки, вздовж аксона (передавач), що наприкінці розгалужується на волокна (***strands***). На закінченнях волокон знаходяться синапси (***synapses***).

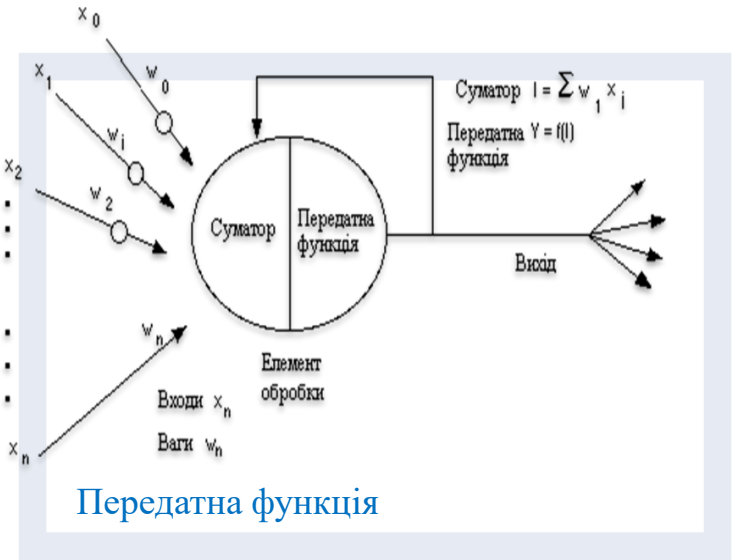
Рис. 3.1. Схема біологічного нейрона

Синапс є функціональним вузлом між двома нейронами (волокно аксона одного нейрона і дендрит іншого). Коли імпульс досягає синаптичного закінчення, продукуються хімічні речовини, названі нейротрансмітерами. Нейротрансмітери проходять через синаптичну щілину, збуджуючи або загальмовуючи, у залежності від типу синапсу, здатність нейрона-приймача генерувати електричні імпульси. Результативність синапсу налаштовується минаючими через нього сигналами, тому синапси навчаються в залежності від активності процесів, у яких вони приймають участь. Нейрони взаємодіють за

допомогою короткої серії імпульсів. Повідомлення передається за допомогою частотно-імпульсної модуляції (Ежов, Шумский, 1998).

Останні експериментальні дослідження доводять, що біологічні нейрони структурно складніші, ніж спрощене пояснення, наведене вище і значно складніші, ніж існуючі штучні нейрони, які є елементами сучасних штучних нейронних мереж. Оскільки нейрофізіологія надає науковцям розширене розуміння дії нейронів, а технологія обчислень постійно вдосконалюється, розробники мереж необмежений простір для вдосконалення моделей біологічного мозку (Уоссермен, 1992).

Штучний нейрон. Штучний нейрон моделює чотири основні функції природного нейрона (Рис. 3.2).



штучний нейрон

Рис. 3.2. Базовий штучний нейрон

Вхідні сигнали x_n зважені ваговими коефіцієнтами з'єднання w_n додаються суматором, який дає на виході *I*, проходять через передатну (активаційну) функцію *Y*, генерують результат *OUT* і виводяться (Ежов, Шумский, 1998).

У наявних на цей час пакетах програм штучні нейрони називаються "елементами обробки" і мають набагато більше можливостей, ніж простий штучний нейрон, описаний вище. На Рис. 3.3 зображена детальна схема спрощеного штучного нейрону (Каллан,2003).

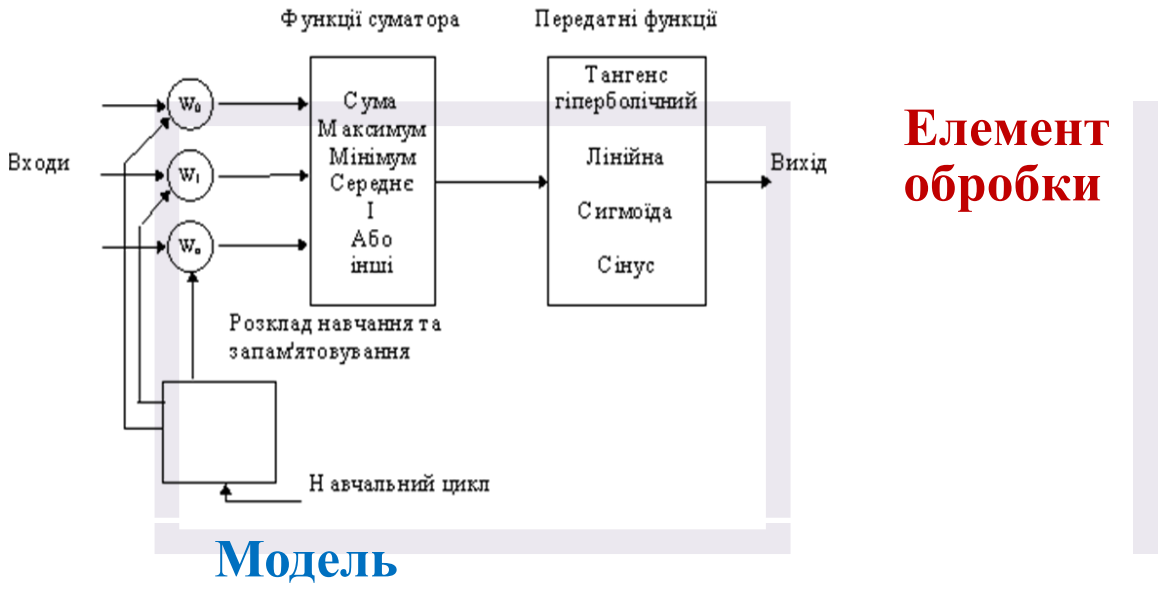


Рис. 3.3. Модель "елемента обробки"

Модифіковані входи передаються на функцію сумування, яка переважно тільки сумує добутки. Проте можна обрати багато різних операцій, такі як середнє, найбільше, найменше, ***OR, AND***, тощо, які могли б виробляти деяку кількість різних значень. Окрім того, більшість комерційних програм дозволяють інженерам-програмістам створювати власні функції сумування за допомогою підпрограм, закодованих на мові високого рівня (C++, Object Pascal, Java).

В любому з цих випадків, вихід функції сумування надсилається у передатну функцію і скеровує весь ряд на дійсний вихід (0 або 1, -1 або 1, або яке-небудь інше число) за допомогою певного алгоритму (Каллан,2003).

Передатна (активаційна) функція *Y(I)* може бути:

1. Пороговою бінарною функцією (рис. 3.4)

$$OUT = \begin{cases} 1, I \geq T \\ 0, I < T \end{cases} \qquad (3.1)$$

де T - деяка постійна порогова величина, або ж функція, що точніше моделює нелінійну передавальну характеристику біологічного нейрона (Люгер, 2003).

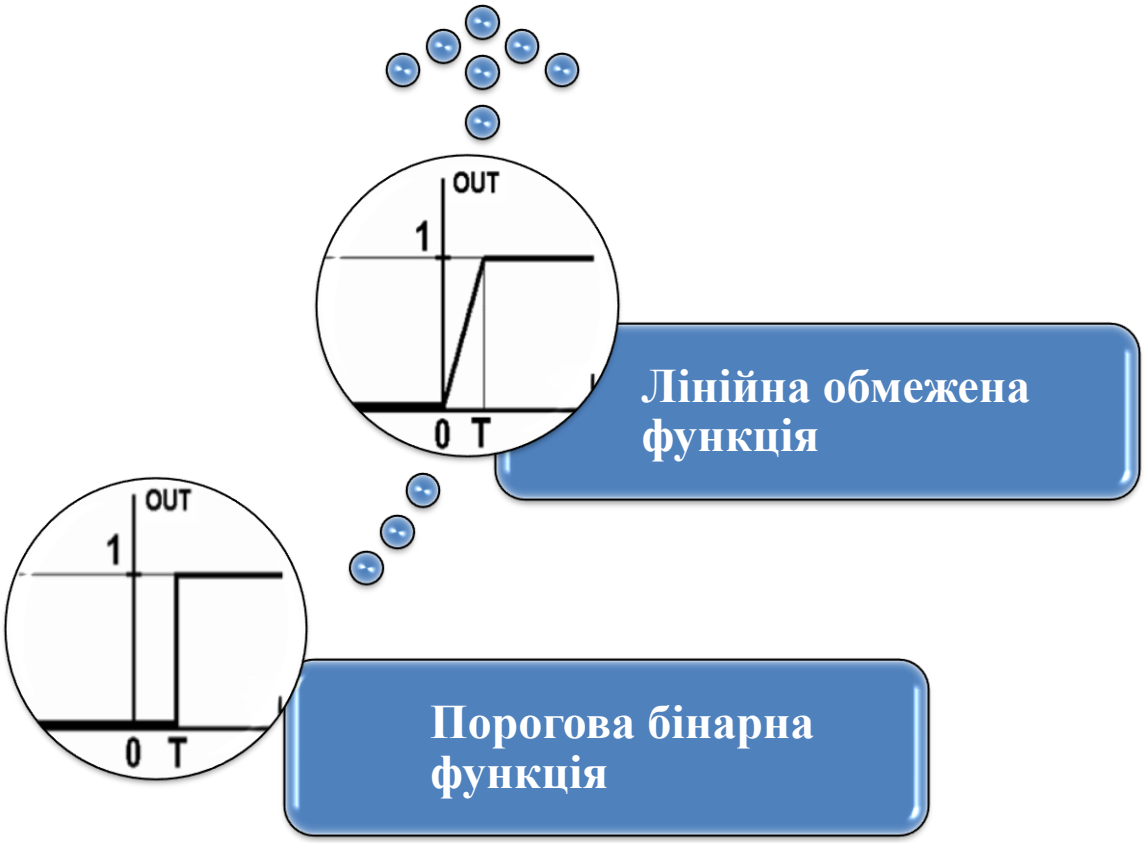


Рис. 3.4. Порогова бінарна функція та лінійна обмежена функція

2. Лінійною обмеженою функцією (Рис. 3.4)

$$OUT = \begin{cases} 1, I \geq T \\ I, 0 \leq I < T \\ 0, I < 0 \end{cases} \qquad (3.2)$$

3. Функцією гіперболічного тангенса (Рис. 3.5)

$$\mathrm{OUT} = th(C \cdot I) = \frac{\exp(C \cdot I) - \exp(-C \cdot I)}{\exp(C \cdot I) + \exp(-C \cdot I)} \qquad (3.3)$$

де $C > 0$ – коефіцієнт ширини сигмоїди по осі абсцис (звичайно C=1).

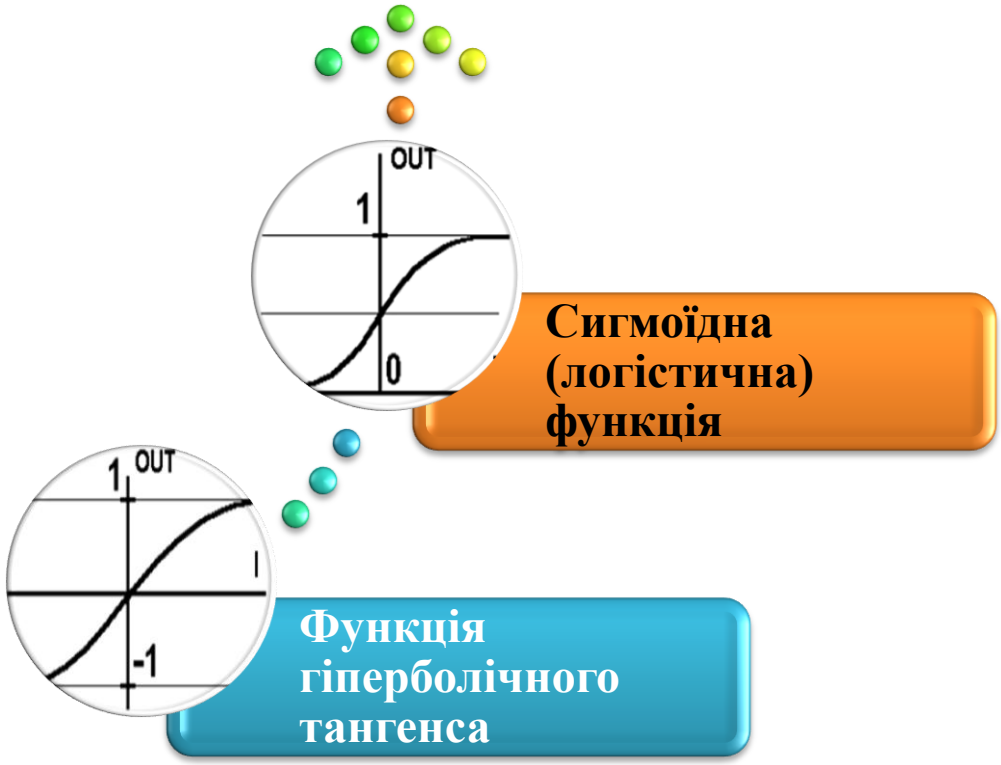


Рис. 3.5. Функція гіперболічного тангенса та сигмоїдна (логістична) функція

4. Сигмоїдною (***S***-подібною) або логістичною функцією (Рис. 3.5)

$$\mathrm{OUT} = \frac{1}{1 + e^{-C \cdot I}} \qquad (3.4)$$

Найбільш популярною є сигмоїдна функція. Це зумовлено наступними її властивостями (Вороновский, Махотило, Петрашев, Сергеев, 1997):

- здатність підсилювати слабкі сигнали сильніше, ніж великі, і опиратися „насиченню” від потужних сигналів;
- монотонність і диференційованість на всій осі абсцис;
- простий вираз для похідної

$$Y'(I) = C \cdot Y(I) \cdot (1 - Y(I)) \quad , \qquad (3.5)$$

що дає можливість використовувати широкий спектр оптимізаційних алгоритмів (Каллан,2003).

Якщо активаційна функція *Y* звужує діапазон зміни величини *I* так, що при будь-яких значеннях *I* значення ***OUT*** належать деякому кінцевому інтервалу, то *Y* називається «стискаючою» функцією. Як «стискаюча» функція часто використовується сигмоїдна функція. Після обробки сигналу, нейрон на виході має результат передатної функції ***OUT***, який надходить на входи інших нейронів або до зовнішнього з'єднання, як це передбачається структурою нейромережі.

Всі штучні нейромережі конструюються з базового формуючого блоку - штучного нейрону. Існуючі різноманітності і фундаментальні відмінності, є підставою мистецтва талановитих розробників для реалізації ефективних нейромереж (Каллан,2003; Люгер, 2003).

3.2 Одно- і багатошарові нейронні мережі з прямим поширенням

Одношарові нейронні мережі з прямим поширенням (перцептрони). Мережа, в якій усі вхідні елементи сполучені безпосередньо з вихідними елементами, називається одношаровою нейронною мережею, або мережею перцептрона. В такій мережі кожен вихідний елемент є незалежним від інших, тобто кожна вага впливає тільки на один з виходів (Рис. 3.6).

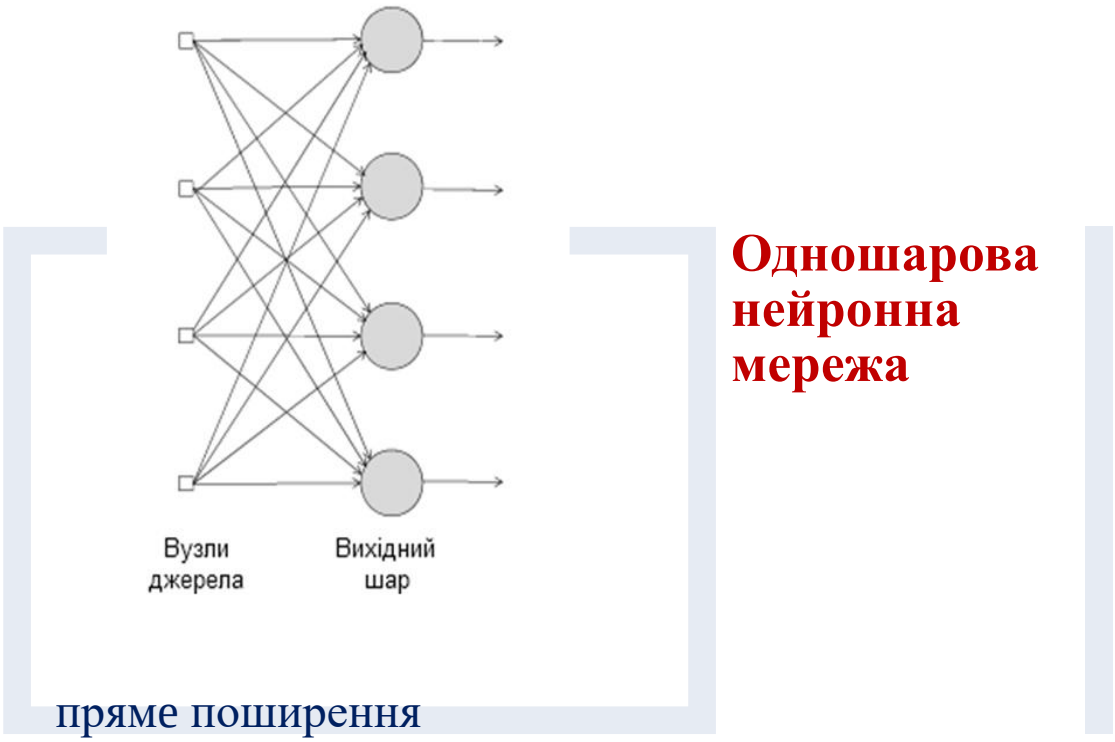


Рис.3.6. Одношарова нейронна мережа прямого поширення - перцептрон

Одношарова мережа прямого поширення є окремим випадком багатошарової мережі.

Багатошарові нейронні мережі з прямим поширенням. Багатошарові мережі прямого поширення характеризуються наявністю одного або декількох прихованих шарів (Рис. 3.7).

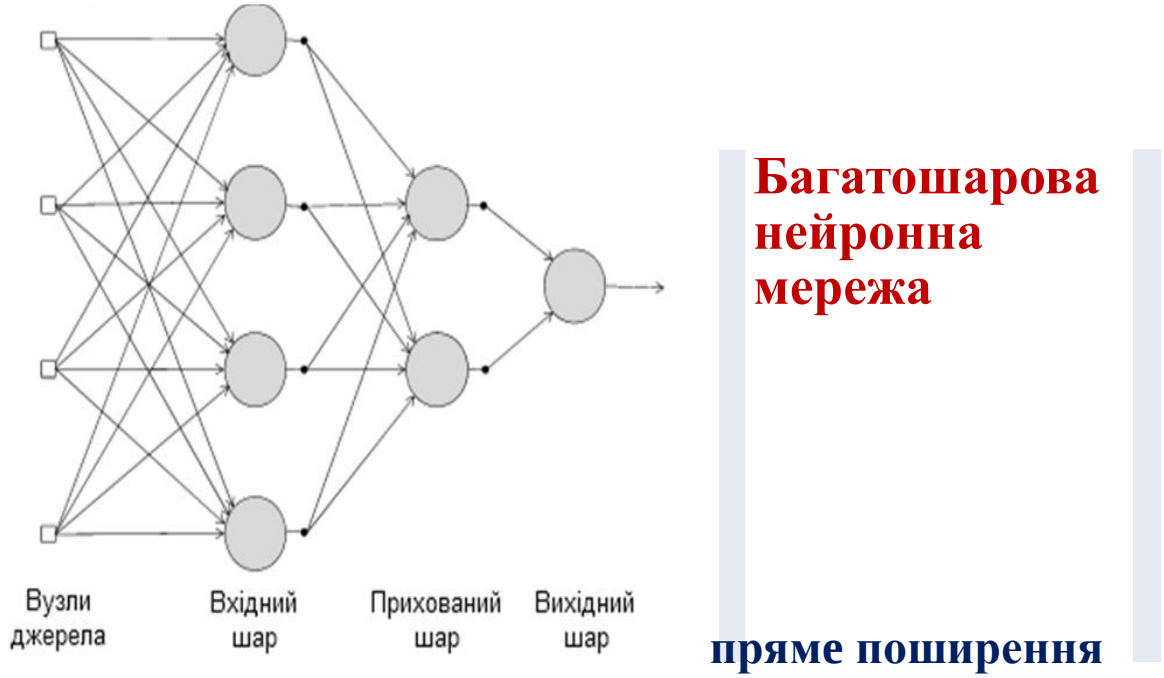


Рис. 3.7. Багатошарова нейронна мережа прямого поширення

Вузли прихованих шарів називаються прихованими нейронами. Додаючи один або декілька прихованих шарів можна виділити глобальні закономірності в даних, що особливо істотно при великій розмірності вхідних даних.

У багатошарових мережах прямого поширення сигнал поширюється пошарово: вузли джерела формують вхідний сигнал для першого (вхідного) шару нейронів, вихідний сигнал першого шару нейронів використовується як вхідний сигнал для другого (прихованого) шару нейронів і т. д. аж до останнього (вихідного) шару нейронів (Каширина, 2005).

Мережа, в якій кожен нейрон прихованого шару з'єднується з усіма нейронами сусідніх шарів, є повнозв'язним. Якщо деякі синаптичні зв'язки відсутні, мережа є неповнозв'язною. (Тут прослідковується аналогія з явищми перколяції, Розділ I).

3.3. Навчання нейронних мереж

3.3.1. Навчання за допомогою алгоритму Хебба

Більшість сучасних алгоритмів навчання виросла з концепцій Хебба, який запропонував модель навчання без вчителя.

Алгоритм навчання Хебба названий так на честь нейрофізіолога, який в нейробіологічному контексті запропонував наступне правило модифікації синаптичних ваг (Каширина, 2005):

*Якщо аксон клітини **A** знаходиться на досить близькій відстані від клітини **B** і постійно або періодично бере участь в її збудженні, спостерігається процес метаболічних змін в одному або обох нейронах, що виражається в тому, що ефективність нейрона А як одного зі збудників нейрона В зростає.*

В контексті штучних нейронних мереж це правило можна сформулювати у вигляді двох тверджень:

- якщо два нейрони, пов'язані синаптичним зв'язком, збуджуються одночасно (синхронно), то міцність цього зв'язку (відповідна синаптична вага) зростає;
- якщо два нейрони по обидві сторони синапсу збуджуються асинхронно, такий синапс слабшає (синаптична вага зменшується).

Можна сформулювати наступні властивості синапсу Хебба:

- *Залежність від часу*. Зміна синаптичної ваги залежить від точного часу виникнення передсинаптичного і постсинаптичного сигналів.
- *Локальність*. На зміну синаптичної ваги чинять дію сигнали, що знаходяться в просторово-часовій близькості.
- *Інтерактивність*. Зміна синаптичної ваги визначається сигналами на обох його кінцях.
- *Кореляція*. Механізм зміни синаптичної ваги визначається наявністю кореляції між передсинаптичним і постсинаптичним сигналом.

Метод навчання Хебба полягає в зміні ваг за наступним правилом:

$$W_{ij}(t) = W_{ij}(t-1) + \alpha\, y_i^{(n-1)}\, y_i^n \qquad (3.6)$$

де $y_i^{(n-1)}$ - вихідне значення нейрона і шару *n-1*, y_i^n - вихідне значення нейрона *j* шару *n*, $W_{ij}(t)$ і $W_{ij}(t-1)$ - ваговий коефіцієнт синапсу, що з'єднує ці нейрони, на ітераціях *n* і *n-1* відповідно; α - коефіцієнт швидкості навчання. Тут і далі, для спільності, під *n* мається на увазі довільний шар мережі. При навчанні за цим методом посилюються зв'язки між збудженими нейронами.

Існує також і диференціальний метод навчання Хебба (Каширина, 2005):

$$W_{ij}(t) = W_{ij}(t-1) + \alpha \left[y_i^{(n-1)}(t) - y_i^{(n-1)}(t-1)\right]\left[y_j^{(n)}(t) - y_j^{(n)}(t-1)\right] \qquad (3.7)$$

Тут $y_i^{(n-1)}$ - вихідне значення нейрона *i* шару *n-1* відповідно на ітераціях *t* і *t-1*, y_i^n - те ж саме для нейрона *j* шару *n*. Як видно з формули (3.7), найсильніше навчаються синапси, що з'єднують ті нейрони, виходи яких найбільш динамічно змінилися в бік збільшення.

Повний алгоритм навчання із застосуванням вищенаведених формул буде виглядати так:

1. На стадії ініціалізації всім ваговим коефіцієнтам привласнюються невеликі випадкові значення.

2. На входи мережі подається вхідний образ, і сигнали збудження поширюються по всім шарам згідно принципам класичних прямопоточних (feedforward) мереж, тобто для кожного нейрона розраховується зважена сума його входів, до якої потім застосовується активаційна (передатна) функція нейрона, в результаті чого виходить його вихідне значення $y_i^{(n)}$, $i = 0..M_i - 1$, де M_i - число нейронів у шарі *i*, *n = 0 ... N-1*, а *N* - число шарів у мережі.

3. На підставі отриманих вихідних значень нейронів по формулі (3.6) або (3.7) виробляється зміна вагових коефіцієнтів.

4. Цикл із кроком 2, поки вихідні значення мережі не стабілізуються із заданою точністю. Застосування цього способу визначення завершення навчання, обумовлено тим, що значення синапсів, що підлаштовуються фактично не обмежені.

На другому кроці циклу послідовно задаються всі образи з вхідного набору.

Слід зазначити, що вид відгуків на кожен клас вхідних образів не відомий заздалегідь і буде являти собою довільне поєднання станів нейронів вихідного шару, обумовлене випадковим розподілом ваг на стадії ініціалізації. Разом з тим, мережа здатна узагальнювати схожі образи, відносячи їх до одного класу. Тестування навченої мережі дозволяє визначити топологію класів у вихідному шарі. Для приведення відгуків навченої мережі до зручного поданням можна доповнити мережу одним шаром, який, наприклад, по алгоритму навчання одношарового перцептрона необхідно змусити відображати вихідні реакції мережі в необхідні образи (Каширина, 2005; Mar'yan & Yurkovych, 2018).

3.3.2. Символьне кодування з алгоритмами Хебба та прямого поширення: розробка програми в середовищі C++

Розроблена програма на алгоритмічній мові C++, проведене її тестування, навчання та шляхи модифікації. Призначенням програми є розпізнавання образів цифр та латинських букв. Реалізована нейронна мережа в якості передатної функції використовує бінарну порогову функцію (0 і 1).

Інтерфейс програми зображений на Рис. 3.8.

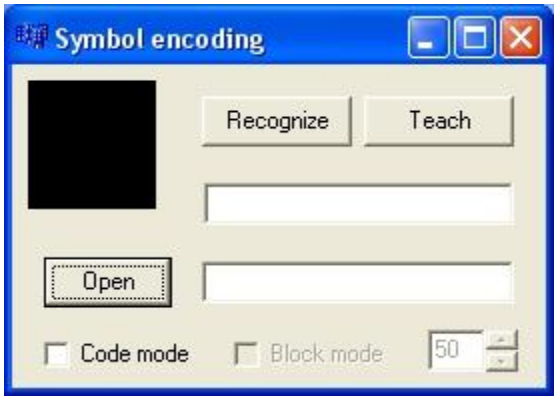


Рис. 3.8. Візуальний інтерфейс програми

Інтерфейс складається з наступних компонентів:

1. Поле зображення SymbolImage;
2. Кнопка вибору зображення через провідник OpenPicture;
3. Кнопка розпізнавання зображення Recognize;
4. Поле виводу результату розпізнавання SymbolEdit;
5. Кнопка проведення навчання нейронної мережі Teach.

Окрім вказаних компонентів існують також поле для вводу коду CodeEdit, перемикачі CodeCheckBox і BlockCheckBox, а також поле BlockEdit із стрілками BlockUpDown. Ці компоненти являють собою другу частину програми, що активується перемикачем CodeCheckBox. SymbolImage показує зображення, з яким в даний момент працює програма. Підтримуються зображення формату .bmp та розміром 64x64 пікселі.

OpenPicture відкриває вікно провідника (Рис. 3.9) та дозволяє вибрати зображення з жорсткого диску для розпізнавання.

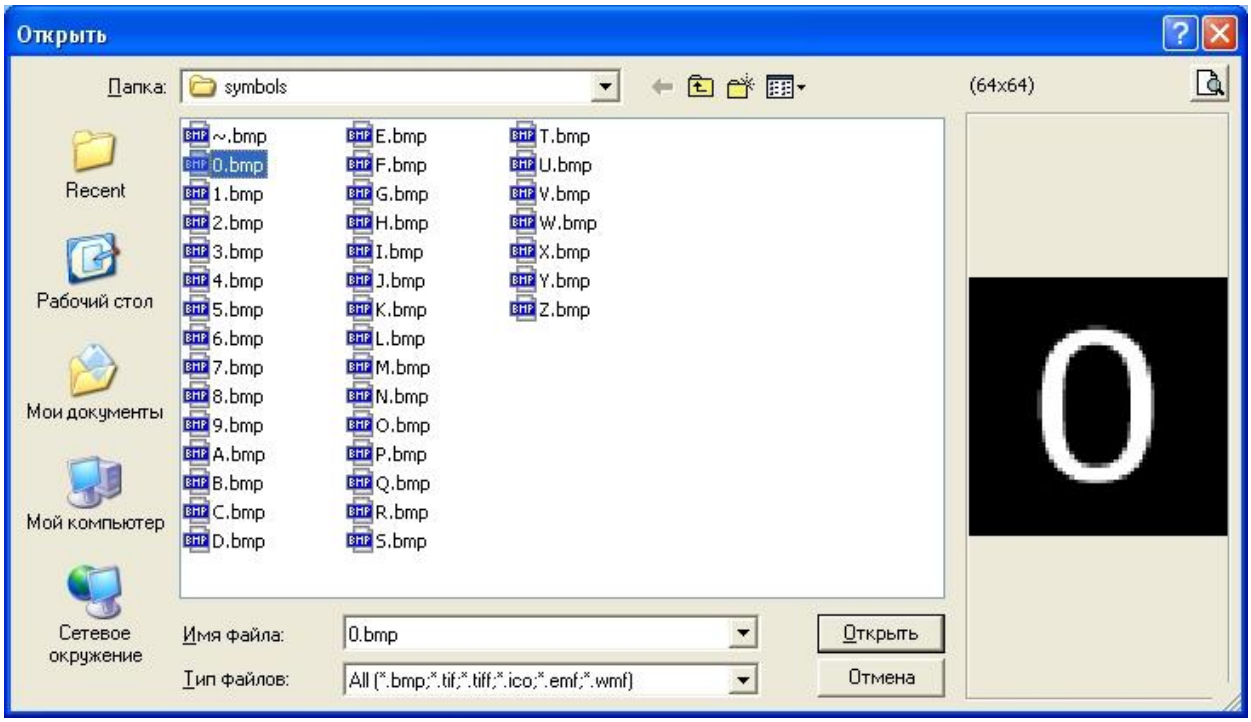


Рис. 3.9. Вікно вибору зображення для розпізнавання

Recognize викликає алгоритм розпізнавання зображення та вивід результату в поле SymbolEdit.

SymbolEdit використовується для зворотного зв'язку з користувачем. Він виводить результати обчислень (в тому числі і помилки). Також він використовується в процесі навчання як поле вводу символу, що зображений на малюнку.

Teach викликає алгоритм навчання нейронної мережі. Це дозволяє налаштувати мережу таким чином, щоб вона виводила очікувані результати при обчисленнях. Для спрацювання алгоритм потребує зображення символу та ручне введення правильної відповіді в поле SymbolEdit.

Нейронна мережа моделюється двома класами: RecognizeNeuron і RecognizePerceptron. Окрім них є також ще два класи: CodeNeuron і CodePerceptron. Вони являють собою іншу нейронну мережу і складають другу частину програми.

Клас RecognizeNeuron (Додаток В). Цей клас є програмним представленням штучного нейрона. Конструктор цього класу задає початкові вагові коефіцієнти і крок їх зміни при навчанні. Ваги синапсів нейрона зберігаються в його атрибуті weights[]. Вхідні дані, що являють собою масив пікселів, перемножаються на відповідний кожному пікселю ваговий коефіцієнт, після чого отримані добутки сумуються. Це реалізовується методом activate(int[]). В результаті роботи нейрона на виході отримується певне число (вказана вище сума), яка характеризує збудження нейрона на зображення. Нейрон з найбільшим збудженням вказує результат розпізнавання.

Навчання нейрона реалізується методом changeWeights(int[]). На вхід подається масив пікселів зображення, якому буде відповідати даний нейрон. Він порівнюється із масивом вагових коефіцієнтів нейрона. Якщо піксель зафарбований, коефіцієнт збільшується на значення step. Атрибут totalWeights зберігає фіксовану суму всіх ваг. Він використовується для того, щоб вирахувати, на яке значення мають зменшитись інші вагові коефіцієнти. Такі

маніпуляції з коефіцієнтами необхідні для правильного розпізнавання спотворених зображень.

Клас RecognizePerceptron (Додаток В). Реалізовує нейронну мережу. Складається з масиву нейронів neurons[]. Метод recognize(int[]) подає вхідний масив пікселів кожному нейрону і порівнює їх рівень збудження. Індекс нейрона з найбільшим збудженням і є результатом розпізнавання (мережа працює не з самим символом а з його кодом; індекс кожного нейрона відповідає певному символу).

Навчання мережі реалізовано методом teach(int[],int). Він отримує на вхід масив пікселів зображення та відповідний йому символ, вказаний користувачем. Цей масив передається в метод changeWeights(int[]) нейрона з порядковим номером, що відповідає коду вказаного символу.

3.3.3. Тестування та аналіз отриманих результатів

На момент запуску програми нейронна мережа вже існує, але вона ще не вміє розпізнавати образи символів. Для того, щоб навчити нейронну мережу розпізнавати, необхідно в поле SymbolEdit ввести символ, що зображений на малюнку, і натиснути кнопку Teach. В результаті нейронна мережа запам’ятає символ (створить внутрішню асоціацію між розташуванням пікселів на зображенні та введеним символом) (Рис 3.10).

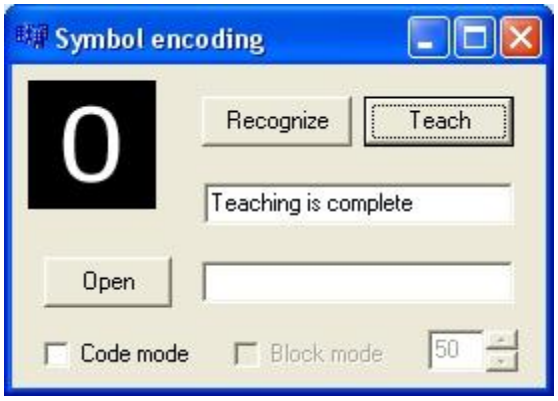


Рис. 3.10. Нейронна мережа навчилася розпізнавати символ

Символ, введений користувачем, перетворюється у свій числовий еквівалент у відповідності із стандартом ANSI. Потім отримане число проходить певні перетворення і приймає значення від 0 до 35 (10 цифр і 26 літер). Ці перетворення потрібні через те, що в інтервалі до літери Z окрім цифр та літер присутні ще й символи керування та розділові символи (цифри знаходяться в інтервалі від 48 до 57, а великі латинські літери – від 65 до 90; відповідно, для цифр відбувається зсув на 48, а для літер – на 55).

Сама нейронна мережа складається із 36 нейронів, кожен із яких відповідає за свій символ. Під час навчання навчається лише нейрон із відповідним індексом.

Якщо для навчання мережі ввести в поле SymbolEdit декілька символів або символ, що не підтримується програмою (не цифру і не латинську букву), нейронна мережа повідомить про некоректність даних для навчання (Рис. 3.11).

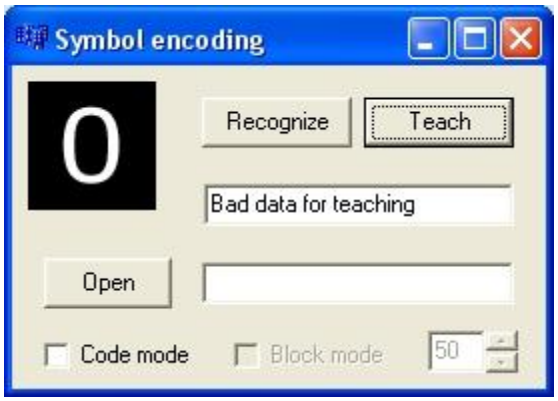


Рис. 3.11. Результат виклику алгоритму навчання по некоректним вхідним даним

Під час навчання зображення розбивається на масив пікселів. Кожен піксель містить інформацію про свій колір (шестизначне шістнадцяткове число). Цей масив переписується таким чином, щоб всі кольори, відмінні від 0 (код чорного кольору за замовчуванням), розглядались як 1. Таким чином фоном є чорний колір, а символ може бути зображений будь-яким іншим кольором.

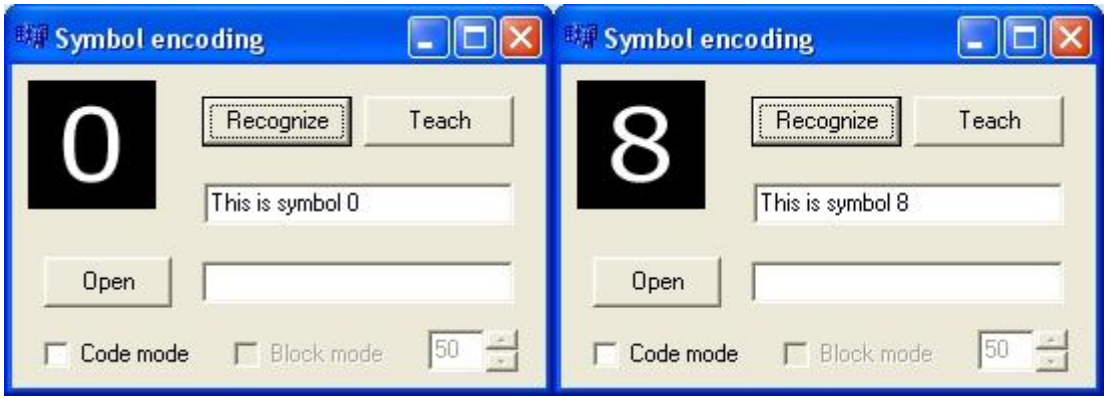


Рис. 3.12. Результат роботи нейронної мережі після навчання

Після проходження навчання для кожного символу, нейронна мережа стає здатною розпізнавати зображення (Рис. 3.12).

3.3.4. Блокування символьного кодування з використанням композиції нейронних мереж

Окремим модулем реалізований алгоритм введення коду доступу та блокування роботи програми при певних умовах. Він підключається перемикачем CodeCheckBox. В результаті принцип роботи програми змінюється. Стають доступні компоненти CodeEdit і BlockCheckBox. Кнопка Teach блокується так як в цьому режимі вона непотрібна. Поле SymbolEdit стає недоступним для редагування – тут виводиться процес вводу коду користувачем.

Ідея модуля полягає в тому, що користувач намагається ввести код доступу (звичайний пароль, відбиток пальця, електронна картка тощо). Програма порівнює введені дані із еталоном. Якщо програма працює у режимі блокування (вмикається перемикачем BlockCheckBox), то при досягненні певного відсотка розпізнавання вхідних даних (тобто співпадіння поданих через зображення символів із еталоном) програма блокується. Таким чином відбувається захист від несанкціонованого доступу зловмисником.

Реалізована програма є спрощеним варіантом вказаної ідеї. В поле CodeEdit вводиться код, що складається з цифр та латинських літер. Це є еталон, з яким будуть порівнюватись дані, введені користувачем через кнопку OpenPicture (тобто вводиться зображення певного символу). Поле SymbolEdit автоматично заповнюється символами «*». Їх кількість рівна кількості символів у еталоні.. При натисканні кнопки Recognize зображення розпізнається у відповідності із навчанням, пройденим в першому режимі роботи програми (перемикач CodeCheckBox вимкнений). Якщо подане на вхід зображення містить символ, вказаний у еталоні, то поле SymbolEdit поновлюється і у відповідне місце записується введений символ (Рис. 3.13).

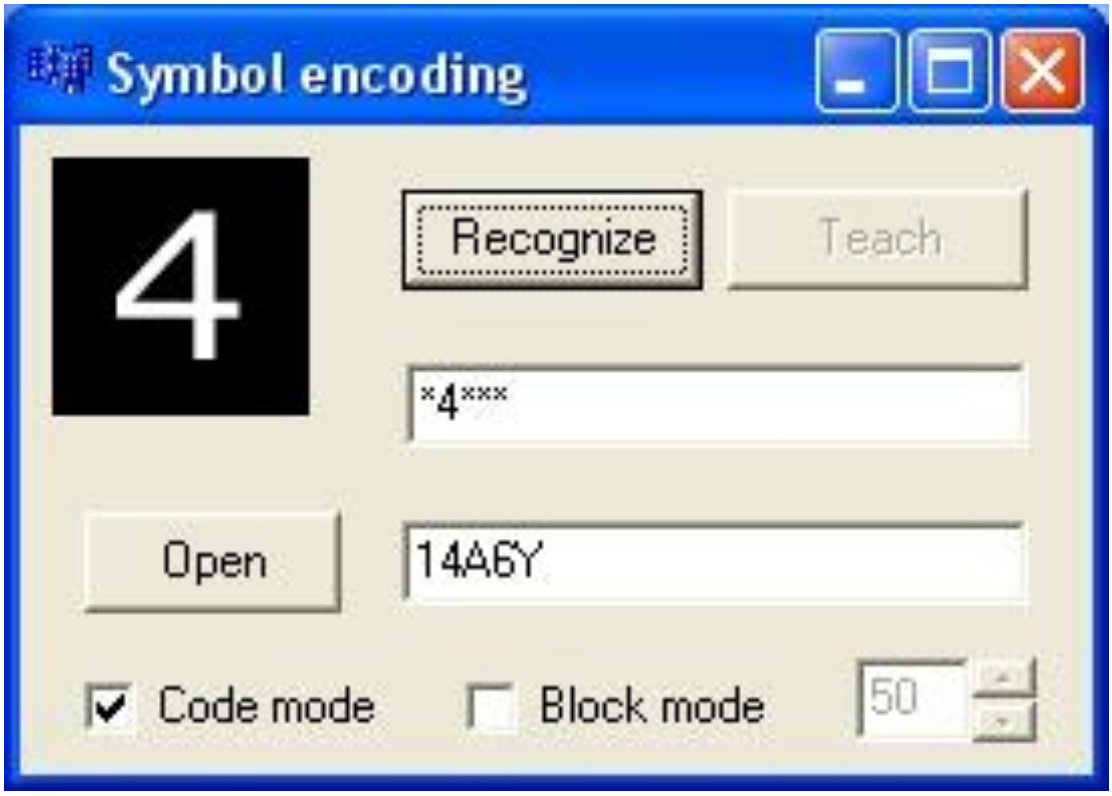


Рис. 3.13. Введення коду доступу користувачем

Таким чином імітується процес ідентифікації користувача.

Робота програми може бути переведена в режим блокування доступу при досягненні певного відсотка розпізнавання. Це досягається вмиканням перемикача BlockCheckBox. Завдяки цьому стають доступними поле BlockEdit і

стрілки BlockUpDown. Вони використовуються для задання граничного значення розпізнавання у відсотках. Тепер, при досягненні вказаного відсотка розпізнавання, програма заблокується (Рис. 3.14).

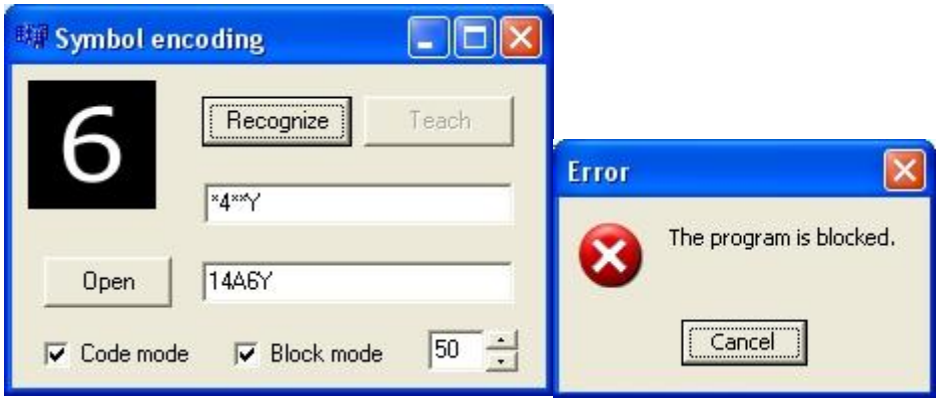


Рис. 3.14. Блокування роботи програми при досягненні вказаного відсотка розпізнавання

Програмно режим коду та блокування реалізований класами CodeNeuron і CodePerceptron.

Клас CodeNeuron (Додаток В) . Цей нейрон зберігає певне еталонне значення в атрибуті value. При виклику функції activate(int) нейрон порівнює вхідне значення із еталонним. Метод getWeights() повертає значення вагового коефіцієнта.

Навчання нейрона реалізується методами setValue(int) і changeWeights(int). Перший задає еталонне значення для нейрона, а другий змінює ваговий коефіцієнт на 1 або -1 (кожен нейрон відповідає за окремий символ коду, значення -1 встановлюється коли нейрон вже спрацював, тобто він вже не приймає участі в подальшій роботі так як відповідний йому символ вже введений користувачем).

Клас CodePerceptron (Додаток В). Містить масив нейронів neurons[]. Метод compare(int) приймає на вхід символ, введений користувачем як частина коду. Далі викликаються методи activate(int) кожного нейрона. Якщо якийсь з них підтвердив співпадіння, то мережа повертає індекс цього нейрона (тобто

порядковий номер символу в коді, з яким співпав введений користувачем символ) і змінює його ваговий коефіцієнт методом changeWeights(int,int) на -1. Метод teach(int,int) задає еталон окремого нейрона.

Алгоритм блокування реалізується методом blocked(int). На вхід він приймає граничне значення відсотка розпізнавання, яке задається користувачем. Цей метод дістає значення вагових коефіцієнтів всіх нейронів і вираховує, скільки з них мають коефіцієнт -1. Визначає їх відсоток і порівнює із заданим граничним значенням.

3.4. Нейронні мережі та синергетика: моделювання фізичних процесів

На основі аналізу літературних даних теорії нейронних мереж та прикладів їх програмних реалізацій проаналізуємо можливість їх застосування в системах технічного захисту інформації, моделювання фізичних процесів. Зокрема, розроблена та представлена програма для символьного розпізнавання з використанням мережі Хебба та прямого поширення на мові C++ , проведене тестування та діагностика моделі, а також її інтерфейс, передбачається її застосування також і в системах різноманітної природи. Реалізована модель нейронної мережі здатна розпізнавати символи на зображенні – цифри та латинські букви. Створено модуль реалізації алгоритму розпізнавання послідовності символів і їх співставлення з конкретним паролем, імітація отримання доступу, та реалізована можливість блокування роботи програми при досягненні заданого відсотка розпізнавання.

Синергетика надає можливість виділити спільні закономірності функціонування (Mar’yan, Seben & Yukovych, 2018; Mar’yan & Yukovych, 2019) таких відокремлених на перший погляд систем – як фізичні, фізико-хімічні та інформаційно-комунікаційні системи (Рис. 3.15). Суттєву роль при цьому відграють нейронні мережі та алгоритми їх розробки, зокрема при розробці інтелектуальних матеріалів, нано, пікорозмірних рівнів структурування самоорганізованих структур (Mar’yan, Yukovych & Seben, 2019.

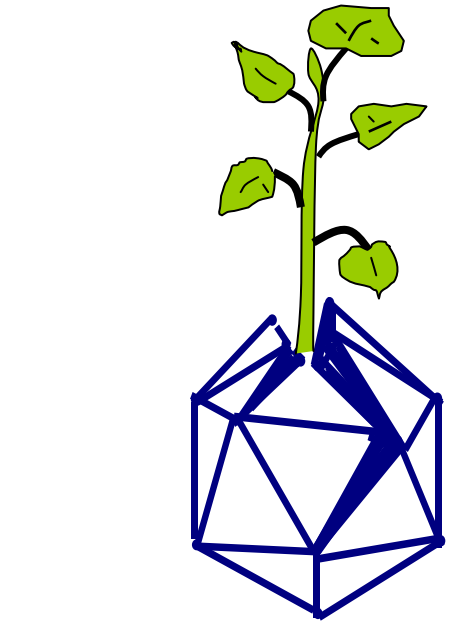


Рис. 3.15. Синергетика систем різноманітної природи

“Збуджений” стан нейрона – це стан одержання (надходження інформації) і кардинальної перебудови системи в околі збудження. Це і закладено в алгоритмах самоорганізації нейронних мереж Хебба, Кохонена. Даний алгоритм проявляється у формуванні та прояві “нейронної структури” розумного будинку, розумного смартфону та інших інтелектуальних інформаційних систем. Це притаманне розвитку Стародавньої Греції (Арістотель та школа Арістотеля це насамперед надходження та поширення інформації, формування інформаційних засад функціонування цивілізації; школи Платона, Архімеда та інші школи так само формувались за принципами, які присутні в алгоритмах самоорганізації нейронних мереж). Інший приклад - Силіконова Долина (США): “збуджений” стан в даному випадку це вихід за межі стандартного дисипативного функціонування університету, перебудова економічних, інформаційних, прикладних зв’язків і формування нової структурованої системи бізнесу, реалізації інформації (Mar’yan & Yurkovych, 2019).

Сигмоїдна функція нейронних мереж, яка відображає процес поширення інформації та перехід у некристалічний стан (Mar'yan & Yurkovych, 2018), для інтелектуальних матеріалів мають спільне та взаємоскорельоване. Це спільне проявляється фрактально, через відповідні атрактори та потоки інформації (Mar'yan & Yurkovych, 2019) і є закономірністю природи. Є спільне в інформаційній структурі нейронних мереж та переходу в некристалічний стан, формування інтелектуальних матеріалів та матеріалів штучного інтелекту. Це насамперед зміна рівня сприймання інформації – мікро, нано, пікопросторові масштаби самоорганізованих структур, які містять інформаційно все необхідне, тому є самодостатні та "інкапсульовані"[6] (Yurkovych, Mar'yan & Seben, 2018).

Залучення технології та алгоритмів нейронних мереж до дослідження фізичних процесів є вкрай актуальним та закономірним, особливо на сучасному рівні розвитку матеріалознавства, розробки інтелектуальних матеріалів. Це комплементарне об'єднання властивостей та способів розробки надає можливість переглянути підходи до дослідження фізичних процесів та визначити кардинально нові напрямки їх сумісного функціонування. Прикладом може бути реалізація ефекту метелика та фрактальності самоорганізованих структур некристалічних матеріалів, транзисторного ефекту та мультистабільності на мікро, нанорівні інтелектуальних систем, кібернетичного ефекту в системах технічного захисту інформації (Mar'yan & Yurkovych, 2019). Важливо, що це не механічне перенесення та об'єднання різних областей, а саме синергетичне переформатування сприймання як фізичних процесів, так і нейронних мереж.

[6] Це проявляється зокрема через формування самоорганізованих інформаційних структур, які реалізують мінімум дисипації енергії на відповідних рівнях структурування (Mar'yan, Yurkovych & Seben,2019)

IV. НЕЙРОННІ МЕРЕЖІ З АЛГОРИТМОМ САМООРГАНІЗАЦІЇ КОХОНЕНА ЯК ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНІ СИСТЕМИ

Останнім часом в самих різних галузях науки і техніки відзначається зростаючий інтерес до застосування штучних нейронних мереж. Багато в чому популярність нейронних мереж пояснюється можливістю їх ефективного використання в задачах погано розв'язуваних "аналітичними" методами. У теоретичних роботах (Круглов, Борисов, 2002), присвячених нейронним мережам, зокрема наголошується, що їх використання доцільно в наступних **завданнях**:

- Класифікації зразків. Завдання полягає у вказівці приналежності вхідного зразка, представленого вектором ознак , одному або декільком попередньо визначеним класам.

- Кластеризації/категоризації. Завдання відрізняється від класифікації зразків тільки тим, що класи заздалегідь не визначені, хоча в багатьох випадках кількість класів таки попередньо вказується.

- Апроксимації функцій. Завдання полягає в знаходженні оцінки функції за відомою вибірці її параметрів і значень. Нейронні мережі рекомендується використовувати у випадках, коли вибірка спотворена шумом і знайти аналітичне рішення скрутно. При цьому попутно вирішується завдання фільтрації, тобто виділення корисного сигналу з фонового шуму.

- Прогнозування. Необхідно на підставі безлічі дискретних відліків $\{f(t_1), f(t_2), \ldots f(t_j)\}$ в послідовні моменти часу передбачити значення $f(t_{j+1})$ в момент часу t_{j+1}.

- Оптимізації, тобто знаходження рішень, які задовольняють системі обмежень і максимізують або мінімізують цільову функцію. Для вирішення цього завдання нейронні мережі рекомендується використовувати при неможливості скласти явні функціональні залежності для обмежень та/або цільової функції .

- Управління з еталонною моделлю. У цих завданнях метою управління є розрахунок такого вхідного керуючого впливу на керовану систему, при якому вона слід за бажаною траєкторії, що диктується еталонною моделлю.

- Створення інформаційно-обчислювальних систем володіють пам'яттю, адресується за змістом, тобто – асоціативної пам'яті. У цьому випадку зміст пам'яті може бути викликано з часткового або спотвореного змісту, що крім іншого позитивно позначається і на живучості таких систем. При цьому асоціативна пам'ять дозволяє вирішувати завдання стиснення інформації та відновлення даних.

4.1. Області застосування та класифікація нейронної мережі Кохонена

Відзначимо, що частково або в комплексі, вирішувати перераховані завдання доводиться при розробці методів і засобів захисту інформації: в компонентах систем оповіщення про атаки (СОА), а також в системах виявлення вразливостей (СВВ). Однак докладного опису механізму такого застосування в літературі відсутньо. Швидше за все, мова йде про використання в зазначених засобах захисту керуючого елемента на базі нейронної мережі - багатошарового перспетрона. З його допомогою вирішується завдання розпізнавання реалізації атаки на комп'ютерну систему, тобто задача розпізнавання образів. СОА на підставі нейронних мереж отримали певне поширення. Проте всі вони мають ряд суттєвих недоліків, які обмежують їх практичну цінність. До зазначених недоліків відносяться: високий рівень помилкових тривог, складність підбору оптимальних граничних параметрів, складність введення в систему нового суб'єкта/об'єкта спостережень, недостатня адаптація до багатьох особливостей сучасного стану галузі інформаційних технологій. Це свідчить про необхідність подальшого удосконалення таких систем. При цьому слід враховувати певний прогрес в розвитку теорії штучних нейронних мереж, що в свою чергу повинно відображатися і на методиці їх використання в задачах захисту інформації на основі використання фізичних процесів. Так, крім перцептрона, вже досить

добре вивчені ще кілька різновидів нейронних мереж, кожна з яких володіє своїми специфічними можливостями. Цим пояснюється актуальність дослідження застосовності різного роду нейронних мереж в задачах захисту інформації.

Штучні нейронні мережі (ШНМ)- математичні моделі, а також їх програмні або апаратні реалізації, побудовані за принципом організації та функціонування біологічних нейронних мереж - мереж нервових клітин живого організму. Це поняття виникло при вивченні процесів, що протікають в мозку, і при спробі змоделювати ці процеси. Першою такою спробою були нейронні мережі У. Маккалок і У. Питтса (Аксенов, Новосельцев, 2006). Після розробки алгоритмів навчання, одержувані моделі стали використовувати в практичних цілях: в задачах прогнозування, для розпізнавання образів, в задачах управління та ін.

ШНМ являють собою систему з'єднаних і взаємодіючих між собою простих процесорів (штучних нейронів). Такі процесори зазвичай досить прості (особливо в порівнянні з процесорами, використовуваними в персональних комп'ютерах). Кожен процесор подібної мережі має справу тільки з сигналами, які він періодично отримує, і сигналами, які він періодично посилає іншим процесорам. І, тим не менш, будучи з'єднаними в досить велику мережу з керованим взаємодією, такі локально прості процесори разом здатні виконувати досить складні завдання.

З точки зору машинного навчання, нейронна мережа являє собою окремий випадок методів розпізнавання образів, дискримінантного аналізу, методів кластеризації та т. п. З математичної точки зору, навчання нейронних мереж - це багатопараметричне завдання нелінійної оптимізації. З точки зору кібернетики, нейронна мережа використовується в задачах адаптивного управління і як алгоритми для робототехніки. З точки зору розвитку обчислювальної техніки та програмування, нейронна мережа - спосіб вирішення проблеми ефективного паралелізму. А з точки зору штучного інтелекту, ШНМ є основою філософської течії коннективізму і основним

напрямком у структурному підході з вивчення можливості побудови (моделювання) природного інтелекту за допомогою комп'ютерних алгоритмів. Нейронні мережі не програмуються в звичному сенсі цього слова, вони навчаються. Можливість навчання - одне з головних переваг нейронних мереж перед традиційними алгоритмами. Технічно навчання полягає в знаходженні коефіцієнтів зв'язків між нейронами. У процесі навчання нейронна мережа здатна виявляти складні залежності між вхідними даними і вихідними, а також виконувати узагальнення. Це означає, що в разі успішного навчання мережа зможе повернути вірний результат на підставі даних, які були відсутні в навчальній вибірці, а також неповних та/або «зашумленних», частково перекручених даних.

Нейронні мережі Кохонена - клас нейронних мереж, основним елементом яких є шар Кохонена. Шар Кохонена складається з адаптивних лінійних суматорів («лінійних формальних нейронів»). Як правило, вихідні сигнали шару Кохонена обробляються за правилом «переможець забирає все»: найбільший сигнал перетворюється в одиничний, інші звертаються в нуль. За способами налаштування вхідних ваг суматорів і за вирішуваними задачами розрізняють декілька різновидів мереж Кохонена. Найбільш відомі з них:

- Мережі векторного квантування сигналів, тісно пов'язані з найпростішим базовим алгоритмом кластерного аналізу (метод динамічних ядер або K-середніх);

- Самоорганізаційні карти Кохонена (Self - Organising Maps, SOM);

- Мережі векторного квантування, яких навчають з учителем (Learning Vector Quantization).

Базова версія. Шар Кохонена складається з деякої кількості n паралельно діючих лінійних елементів. Всі вони мають однакове число входів m і отримують на свої входи один і той же вектор вхідних сигналі $x = (x_1, ...x_m)$. На виході jго лінійного елемента отримуємо сигнал

$$y_j = w_{j0} + \sum_{i=1}^{m} w_{ji} x_i,$$

де w_{ji} - ваговий коефіцієнт iго входу jго нейрона, w_{j0} - пороговий коефіцієнт. Після проходження шару лінійних елементів сигнали посилаються на обробку за правилом «переможець забирає все»: серед вихідних сигналів y_jшукається максимальний; його номер $j_{\max} = \arg\max_j \{y_j\}$. Остаточно, на виході сигнал з номером $j_{\max}$ дорівнює одиниці, інші - нулю. Якщо максимум одночасно досягається для декількох $j_{\max}$, то або приймають всі відповідні сигнали рівними одиниці, або тільки перший у списку (за згодою). “Нейрони Кохонена можна сприймати як набір електричних лампочок, так що для будь-якого вхідного вектора загоряється одна з них”.

Геометрична інтерпретація. Розбиття площини на багатокутники Вороного - Дирихле для випадково вибраних точок (кожна точка вказана у своєму многоугольнике).

Великого поширення набули шари Кохонена, побудовані таким чином: кожному (jму) нейрону зіставляється точка $W_j = (w_{j1}, ...w_{jm})$ в m-вимірному просторі (просторі сигналів). Для вхідного вектора $x = (x_1, ...x_m)$ обчислюються його евклідові відстані $\rho_j(x)$ до точок W_j і «найближчий отримує все» - той нейрон, для якого це відстань мінімально, видає одиницю, решта - нулі. Слід зауважити, що для порівняння відстаней досить обчислювати лінійну функцію сигналу:

$$\rho_j(x)^2 = \|x - W_j\|^2 = \|W_j\|^2 - 2\sum_{i=1}^{m} w_{ji}x_i + \|x\|^2$$

(тут $\|y\|$ - Евклідова довжина вектора: $\|y\|^2 = \sum_i y_i^2$). Останній доданок $\|x\|^2$ однаково для всіх нейронів, тому для знаходження найближчої точки воно не потрібно. Завдання зводиться до пошуку номера найбільшого з значень лінійних функцій:

$$j_{\max} = \arg\max_j \left\{ \sum_{i=1}^{m} w_{ji}x_i - \frac{1}{2}\|W_j\|^2 \right\}.$$

Таким чином, координати точки $W_j = (w_{j1}, ...w_{jm})$ збігаються з вагами

лінійного нейрона шару Кохонена (при цьому значення порогового коефіцієнта $w_{j0} = -\|W_j\|^2/2$).

Якщо задані точки $W_j = (w_{j1}, ...w_{jm})$, то m-мірний простір розбивається на відповідні багатогранники Вороного - Дирихле V_j: багатокутник V_j складається з точок, які ближче до W_j, ніж до інших $W_k (k \neq j)$.

Мережі векторного квантування. Завдання векторного квантування з k кодовими векторами W_j для заданої сукупності вхідних векторів S ставиться як завдання мінімізації спотворення при кодуванні, тобто при заміщенні кожного вектора з S відповідним кодовим вектором . У базовому варіанті мереж Кохонена використовується метод найменших квадратів та спотворення D обчислюється за формулою

$$D = \sum_{j=1}^{k} \sum_{x \in K_j} \|x - W_j\|^2,$$

де K_j складається з тих точок $x \in S$, які ближче до W_j, ніж до інших $W_l (l \neq j)$. Іншими словами, K_j складається з тих точок $x \in S$, що кодуються вектором W_j.

Якщо сукупність S задана і зберігається в пам'яті, то стандартним вибором в навчанні відповідної мережі Кохонена є метод K- середніх. Це метод розщеплення:

- при цьому виборі кодових векторів (вони ж вагові вектори мережі) W_j мінімізацією D знаходимо безлічі K_j - вони складається з тих точок $x \in S$, які ближче до W_j, ніж до інших W_i;;

- при даному розбитті S на безлічі K_j мінімізацією D знаходимо оптимальні позиції кодових векторів W_j - для оцінки за методом найменших квадратів це просто середні арифметичні :

$$W_j = \frac{1}{|K_j|} \sum_{x \in K_j} x,$$

де $|K_j|$ - число елементів в K_j.

Далі ітеруємо. Цей метод розщеплення сходиться за кінцеве число кроків і дає локальний мінімум спотворення (Дебок, Кохонен, 2001).

Якщо ж, наприклад, сукупність S заздалегідь не задана, або з якихось причин не зберігається в пам'яті, то широко використовується онлайн метод. Вектори вхідних сигналів x обробляються по одному, для кожного з них знаходиться найближчий кодовий вектор («переможець», який «забирає все») $W_{j(x)}$. Після цього даний кодовий вектор перераховується за формулою

$$W_{j(x)}^{\text{new}} = W_{j(x)}^{\text{old}}(1-\theta) + x\theta,$$

де $\theta \in (0,1)$ - крок навчання. Решта кодових вектори на цьому кроці не змінюються.

Для забезпечення стабільності використовується онлайн метод з затухаючою швидкістю навчання: якщо T - кількість кроків навчання, то вважають $\theta = \theta(T)$. Функцію $\theta(T) > 0$ вибирають таким чином, щоб $\theta(T) \to 0$ монотонно при $T \to \infty$ і щоб ряд $\sum_{T=1}^{\infty} \theta(T)$ був розбіжним, наприклад, $\theta(T) = \theta_0/T$.

Векторне квантування є загальної операцією, ніж кластеризація, оскільки кластери повинні бути розділені між собою, тоді як сукупності K_j для різних кодових векторів W_j не обов'язково представляють собою роздільні кластери. З іншого боку, за наявності кластерів, що розділяються, векторне квантування може знаходити їх і по-різному кодувати.

Самоорганізаційна карта Кохонена (англ. **Self- organizing map - SOM**) - змагальна нейронна мережа з навчанням без учителя, що виконує задачу візуалізації та кластеризації. Ідея мережі запропонована фінським вченим Т. Кохоненом. Є методом проекціювання багатовимірного простору в простір з більш низькою розмірністю (найчастіше, двовимірне), застосовується також для вирішення задач моделювання, прогнозування та ін. Є однією з версій нейронних мереж Кохонена (Дебок, Кохонен, 2001).

Самоорганізаційна карта складається з компонент, названих вузлами (нейронами). Їх кількість задається аналітиком. Кожен з вузлів описується

двома векторами. Перший - так званий вектор ваги **m**, що має таку ж розмірність, що і вхідні дані. Другий - вектор **r**, що представляє собою координати вузла на карті. Зазвичай вузли розташовують у вершинах регулярної решітки з квадратними або шестикутними осередками (Дебок, Кохонен, 2001).

Спочатку відома розмірність вхідних даних, по ній деяким чином будується початковий варіант карти. У процесі навчання вектори ваги вузлів наближаються до вхідних даних. Для кожного спостереження (семпла) вибирається найбільш схожий по вектору ваги вузол, і значення його вектора ваги наближається до спостереження. Також до спостереження наближаються вектори ваги декількох вузлів, розташованих поруч. Таким чином, якщо в множині вхідних даних два спостереження були схожі, на карті їм будуть відповідати близькі вузли. Циклічний процес навчання, перебираючий вхідні дані, закінчується по досягненні картою допустимої (заздалегідь заданою аналітиком) похибки, або по здійсненні заданої кількості ітерацій (Kohonen, 2001).

Алгоритм мережі.

1. Ініціалізація карти, тобто первісне завдання векторів ваги для вузлів.
2. Цикл:

а. Вибір наступного спостереження (вектора з безлічі вхідних даних).

б. Знаходження для нього кращої одиниці відповідності (best matching unit, BMU, або Winner) - вузла на карті, вектор ваги якого найменше відрізняється від спостереження (в метриці, що задається аналітиком, найчастіше, евклідової).

в. Визначення кількості сусідів BMU і навчання - зміна векторів ваги BMU та його сусідів з метою їх наближення до спостереження.

г. Визначення помилки карти.

Деталізація Алгоритму.

1. Ініціалізація.

Найбільш поширені три способи завдання початкових ваг вузлів:

- Завдання всіх координат випадковими числами.

- Присвоєння вектору ваги значення випадкового спостереження з вхідних даних.

- Вибір векторів ваги з лінійного простору, поширеного на головні компоненти набору вхідних даних.

2. Цикл.

Нехай t - номер ітерації (ініціалізація відповідає номеру 0).

- Вибрати довільне спостереження $x(t)$ з безлічі вхідних даних.

- Знайти відстані від нього до векторів ваги всіх вузлів карти і визначити найближчий за вагою вузол $M_c(t)$. Це - BMU або Winner . Умова на_$M_c(t)$: $\|x(t) - m_c(t)\| \leq \|x(t) - m_i(t)\|$,

для будь-якого $m_i(t)$, де $m_i(t)$ - вектор ваги вузла $M_i(t)$. Якщо знаходиться декілька вузлів, які відповідають умові, BMU вибирається випадковим чином серед них.

- Визначити за допомогою функції h (функції сусідства) сусідів M_c і зміна їх векторів ваги.

3. Задання h.

- Функція визначає "міру сусідства" вузлів M_i і M_c і зміну векторів ваги. Вона повинна поступово уточнювати їх значення, спочатку у більшої кількості вузлів і сильніше, потім у меншої і слабкіше. Часто як функція сусідства використовується функція Гауса:

$$h_{ci}(t) = \alpha(t) \cdot \exp(-\frac{\|r_c - r_i\|^2}{2\sigma^2(t)}),$$

де $0 < \alpha(t) < 1$ - навчальний співмножник, монотонно регресний з кожною наступною ітерацією (тобто визначає наближення значення векторів ваги BMU та його сусідів до спостереження; чим більше крок, тим менше уточнення);

r_i, r_c - координати вузлів $M_i(t)$ і $M_c(t)$ на карті;

$\sigma(t)$ - співмножник, що зменшує кількість сусідів з ітераціями, монотонно спадає.

Параметри α, σ і їх характер убування задаються аналітиком. Простіший спосіб задання функції сусідства:

$h_{ci}(t) = \alpha(t)$,

якщо $M_i(t)$ знаходиться в околі $M_c(t)$ заздалегідь заданого аналітиком радіусу, і 0 у протилежному випадку.

Функція $h(t)$ дорівнює $\alpha(t)$ для BMU і зменшується з віддаленням від BMU .

4. Зміна векторів ваг.

Змінити вектор ваги за формулою:

$$m_i(t) = m_i(t-1) + h_{ci}(t) \cdot (x(t) - m_i(t-1))$$

Таким чином вектори ваги всіх вузлів, які є сусідами BMU, наближаються до розглянутого спостереження.

5. Обчислення помилки карти.

Наприклад, як середнє арифметичне відстаней між спостереженнями і векторами ваги відповідних їм BMU:

$$\frac{1}{N}\sum_{i=1}^{N}\|x_i - m_c\|,$$

де N - кількість елементів набору вхідних даних.

Архітектура нейронних мереж Кохонена в системах захисту інформації.

Багатошаровий перцептрон доцільно використовувати в тих засобах захисту інформації, які базуються на аналізі безлічі взаємокорелюючих дискретних параметрів. До таких засобів захисту відносяться системи розпізнавання атак, системи розпізнавання вразливостей, антивіруси, антикейлогери. Можливі вхідні параметри багатошарового перцептрона для цих засобів захисту показані в таблиці 4.1.

Призначенням мережі Кохонена є кластеризація образів (Kohonen, 2001). Відзначимо, що в будь-якому випадку зазначений алгоритм реалізує принцип навчання без вчителя. Хоча це і розширює адаптивні можливості мережі, але не дозволяє використовувати накопичені знання про досліджуваний процес. Тому

в сучасних нейромережевих системах мережа Кохонена самостійно не використовується (Дебок, Кохонен, 2001).

Таблиця 4.1. Багатошаровий перцептрон

Назва засобів захисту	**Вхідні параметри**
Система розпізнавання атак	**Установки Інтернет запитів і подій в комп'ютерній системі: вхід / вихід користувачів, кількість процесів, доступ до файлів, тимчасові інтервали запитів до об'єктів комп'ютерної системи**
Система розпізнавання вразливостей	**Параметри налаштувань комп'ютерної системи: кількість користувачів, привілеї користувачів, параметри доступу до об'єктів комп'ютерної системи, кількість і номенклатура відкритих портів, запущені мережеві служби, параметри адміністративних налаштувань служб DCOM / COM +**
Антивіруси, антикейлогери	**Параметри подій в комп'ютерній системі: кількість і номенклатура запущених програм і процесів, доступ процесів до файлів, спроби доступу до мережних служб, спроби зміни виконуваних файлів, доступ до API операційної системи**

Модифікація нейронної мережі Кохонена.

Модифікація полягає в додаванні до складу мережі шару Гросберга (зірки Гросберга). У режимі розпізнавання нейрони шару Кохонена визначають кластер, до якого належить вхідний образ. Потім вихідна зірка шару Гросберга, навчаючись "з вчителем", по сигналу нейрона - переможця в шарі Кохонена відтворює на виходах мережі відповідний образ. Навчання ваг шару Кохонена виконується без вчителя на основі самоорганізації. Вхідний вектор спочатку нормується, зберігаючи напрямок. Після виконання однієї ітерації навчання визначається нейрон - переможець, стан її порушення встановлюється рівним одиниці, і тепер можуть бути модифіковані ваги відповідної йому зірки Гросберга. Темпи навчання нейронів Кохонена і Гросберга повинні бути узгоджені. У шарі Кохонена навчаються ваги всіх нейронів в околі переможця, який поступово звужується до одного нейрона.

Рекомендується використання цієї архітектури для швидкого моделювання систем на початкових етапах досліджень з подальшим переходом, якщо це буде потрібно, на значно більше ресурсномісткий, але більш точний метод навчання зі зворотним поширенням помилок.

4.2. Алгоритм програми: загальний опис

Ось так виглядає головне меню програми:

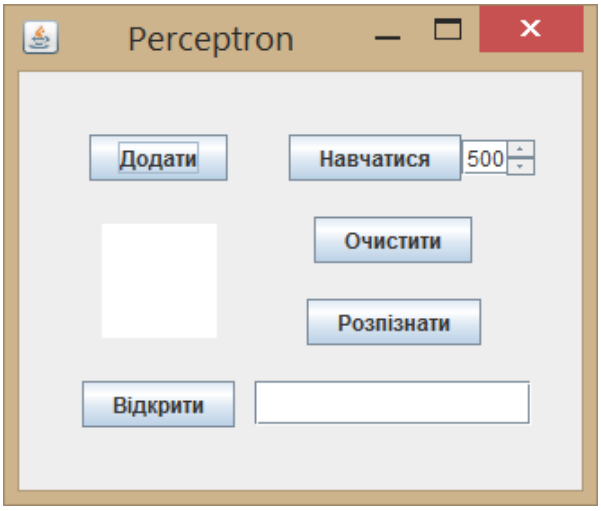


Рис. 4.1. Головне меню програми

Роботу програми починаємо з кнопки "Навчатися". Ця кнопка задає випадкове значення ваговим коефіцієнтам нейронів та перераховує їх у відповідності зі знаннями, які вже доступні мережі. В полі JSpinner задаємо кількість циклів навчання. Цей процес може зайняти деякий час, прогрес виконання можна побачити в консолі.

Тепер мережа готова до розпізнавання зображення. За допомогою мишки малюємо в полі ImageComponent цифру від 0 до 9 і натискаємо кнопку "розпізнати" і в полі JTextField з'являється результат операції (Рис. 4.2):

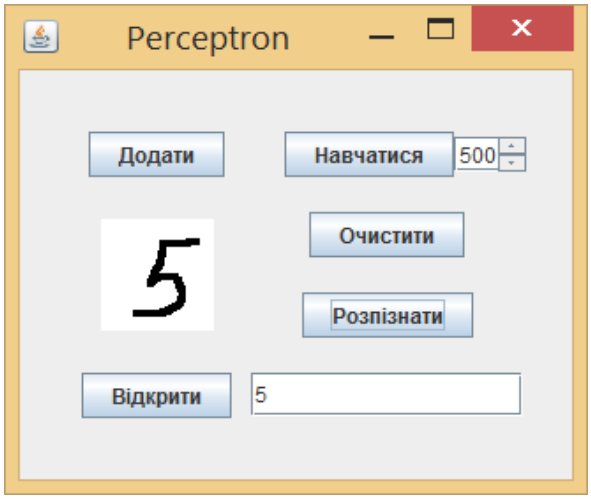


Рис. 4.2. Запуск програми

Для того, щоб очистити поле ImageComponent натискаємо кнопку "Очистити" (Рис. 4.3):

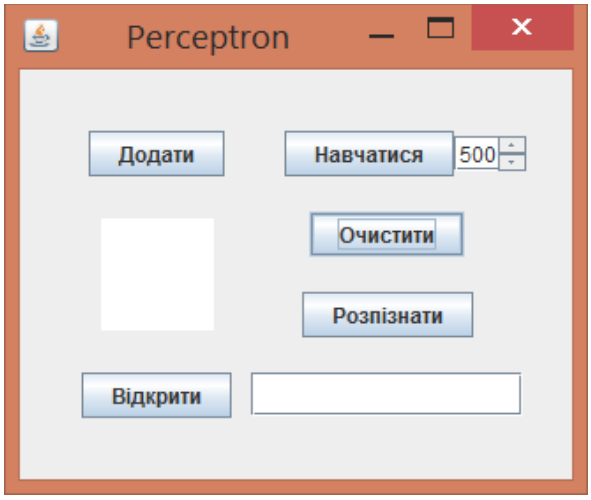


Рис. 4.3. Налаштування програми розпізнавання символів: очистка

Також, ми можемо завантажити готове зображення з комп'ютера. Для цього треба натиснути кнопку "Відкрити" і у діалоговому вікні, що з'явилося, вибрати потрібний файл та натиснути кнопку "Open", і це зображення з'явиться на панелі ImageComponent:

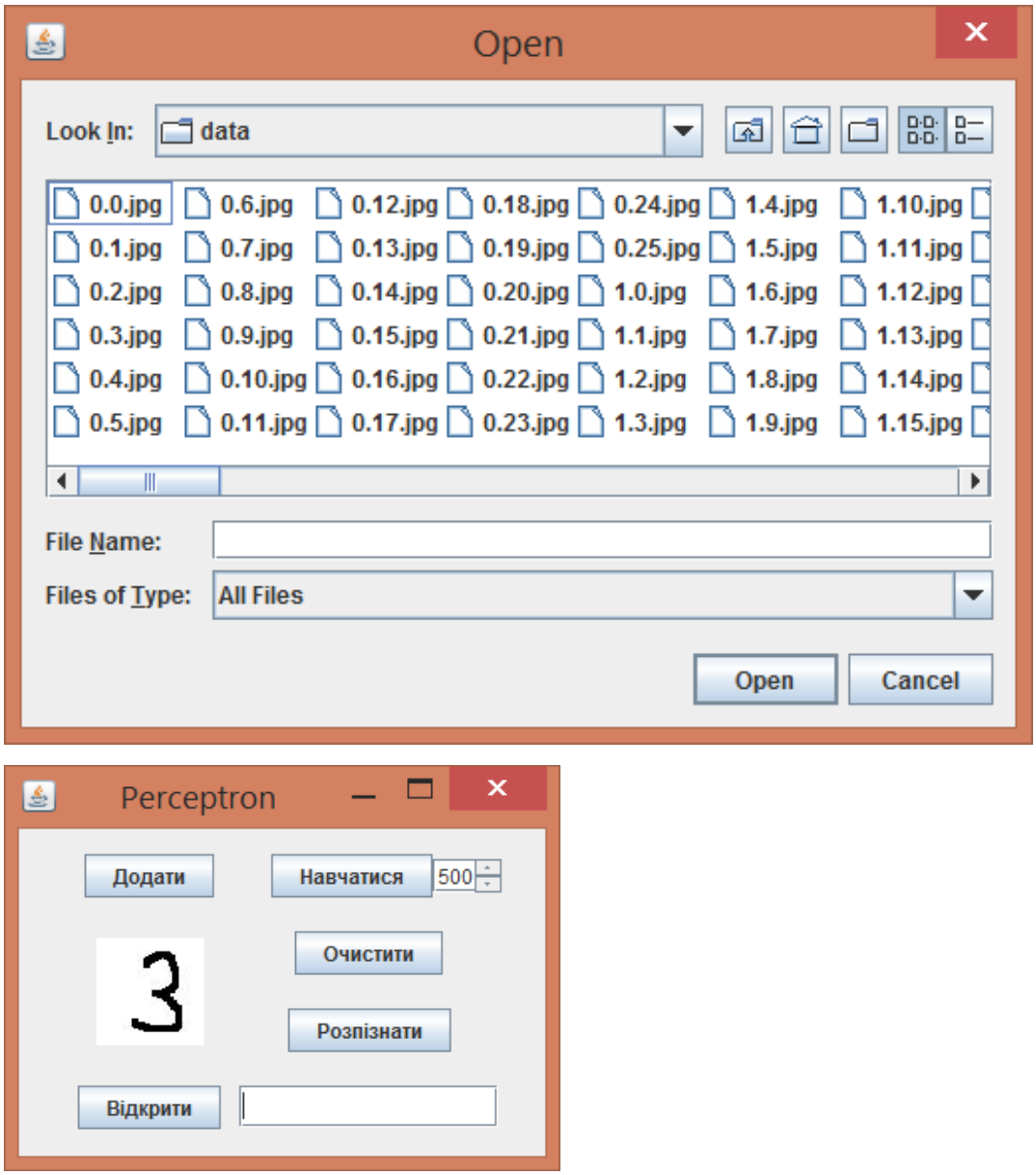


Рис. 4.4. Налаштування програми розпізнавання символів: зміна символів

Так само натискаємо кнопку "Розпізнати" і в полі JTextField отримуємо результат:

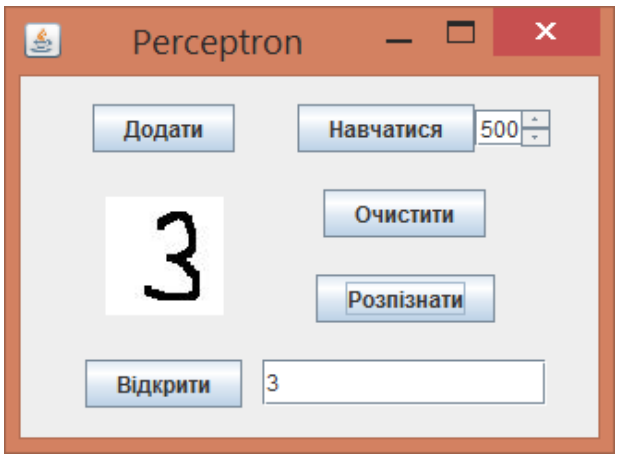


Рис. 4.5. Процес тестування програми

Якщо зображення розпізнається неправильно(а така ймовірність порядку 30%), потрібно ввести в поле JTextField правильну відповідь та натиснути кнопку "Додати":

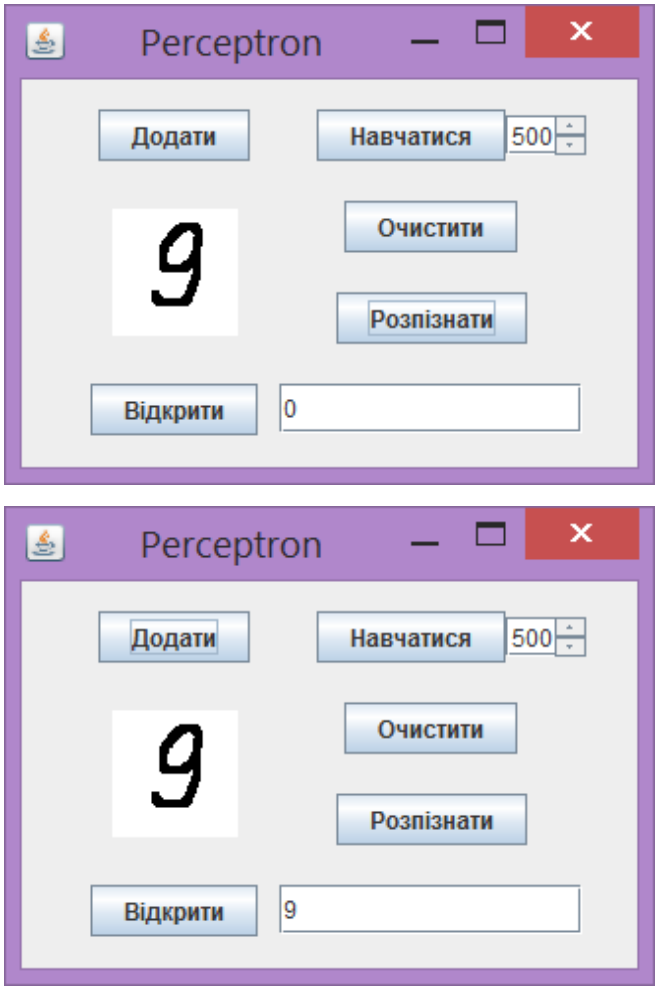


Рис. 4.6. Процес тестування програми

Однак, після цього, потрібно ще раз перерахувати вагові коефіцієнти нейронів, щоб мережа "вивчила" внесені зміни.

Опис класів програми приведено та проаналізовано в Додатку Г. З апробованих нейронних мереж найбільш перспективними для застосування в засобах захисту комп'ютерних систем є багатошаровий перцептрон - мережа Кохонена. Багатошаровий перцептрон, мережу Кохонена доцільно використовувати в керуючих елементах СОА, СВВ, антивірусних системах і в системах захисту від кейлогерів. При цьому, в якості аналізу та застосування можлива імплементація параметрів, представлених в табл. 4.1.

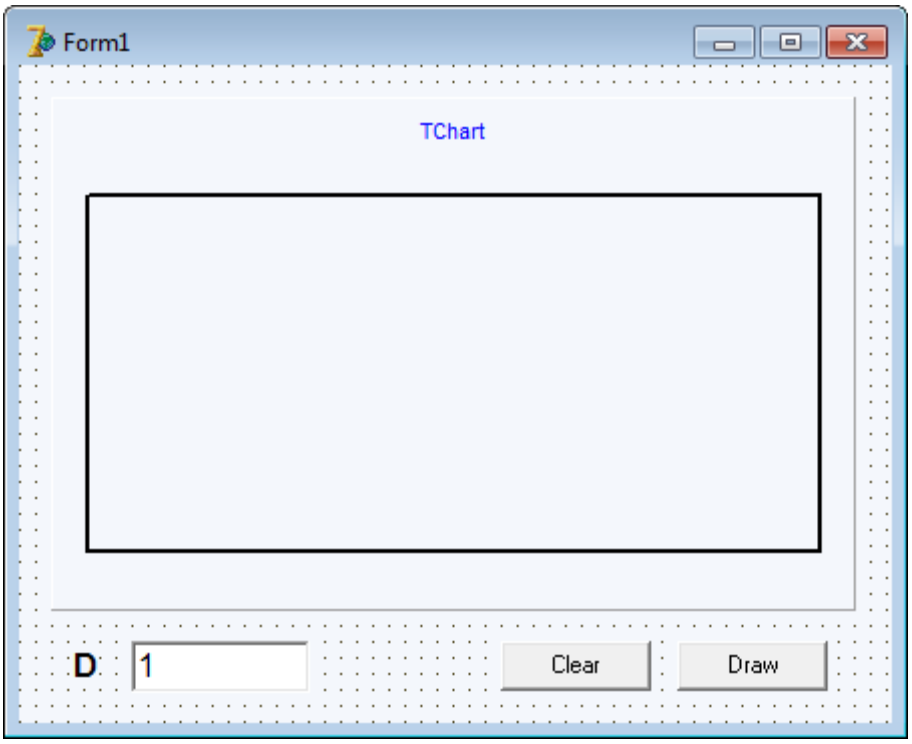
Внаслідок того, що можливості зазначених вище нейронних мереж з алгоритмами самоорганізації Хебба та Кохонена багато в чому доповнюють один одного, перспективним шляхом подальших досліджень є розробка комбінованої на їх основі комплексної нейронної мережі (Mar'yan & Yurkovych, 2018). Її основними характеристиками повинні бути досить висока ємність і точність, низьке енергоспоживання, а також можливість до навчання в процесі функціонування (Yurkovych, Mar'yan & Seben, 2018). Окрім того, є актуальним інтеграція як єдиного цілого фізичних процесів та нейронних мереж із залученням синергетики (Mar'yan & Yurkovych, 2014; Mar'yan & Yurkovych, 2019). Зокрема, алгоритми самоорганізації в некристалічних матеріалах, перколяційних процесів, обробки електронно-мікроскопічних досліджень, розглянуті нами в розділі І, можуть бути адаптовані та імплементовані при розробці програмного забезпечення нейронних мереж. З іншого боку, процеси поширення інформації в нейронних мережах, її обробки та зберігання можуть бути застосовні і до фізичних задач (Mar'yan, Yurkovych & Seben, 2019; Yurkovych, Mar'yan & Seben, 2019). Актуальними, наприклад, є інноваційні технології комп'ютерних програм Чатбот (Chatbot), Target-програм, розроблених на основі нейромереж та технологій машинного навчання в якості віртуальних помічників обробки фізичних процесів та досліджень, використання науково-метричних баз даних.

ДОДАТКИ

Додаток А. Лістинг коду створення серії для лінійного графіка та додавання її до об'єкта TChart

Створення програми для побудови графіка

Візуальний вигляд програми в редакторі:

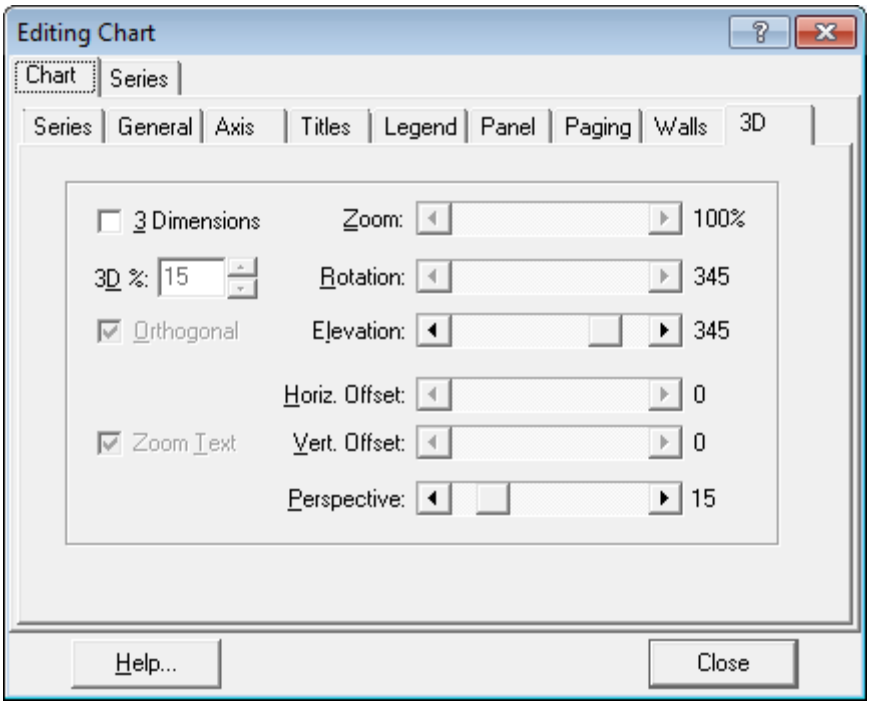


Використовуються компоненти: *TButton*, *TLabel*, *TEdit*, *TChart*. Задамо параметр Name для наступних елементів:

- *TButton* – "BtnClear", "BtnDraw";
- *TEdit* – "EditG0";
- *TChart* – "Chart";

Для побудови графіків у компоненті *TChart* використовується допоміжний контейнер *TLineSeries*. Коротко кажучи, це динамічний масив для пар координат (*x*, *y*). Щоб його використовувати у коді потрібно оголосити його

у фрагменті використовуваних модулів (*uses Series*). Два рази клацаємо на *TChart* , знімаємо галочку із 3D режима, як показано на формі.



Будуємо графік використовуючи формулу (1.1) для періоду дисипативної структури. По осі Y буде період L_c (мкм) а по X час релаксації τ_{rel} (с). Будемо змінювати τ_{rel} в деяких межах, і по його зміні розраховуватимемо період L_c.

Код для події *OnClick BtnClear* (закладка Events):

```
begin
    Chart.RemoveAllSeries;
end;
```

Цей код просто видаляє всі наявні серії в графіку, тобто очищає його.

Код для події *OnClick BtnDraw* (у закладці Events):

```
const
    PI = 3.14;
var
  G0, Gmin: double;
  D : double;
  TrelMin, TrelMax: double;
  TrelStep: double;
  TrelCurrent : double;
  StepCount : integer;
```

```
  Lc : double;
  LineSeries : TLineSeries;
  i : integer;
begin
  G0 := 1.4;
  Gmin := 1;
  D := StrToFloat(EditG0.Text);
  TrelMin := 0.0001;
  TrelMax := 0.1;
  StepCount := 100;
  LineSeries := TLineSeries.Create(Chart);
  TrelStep := (TrelMax - TrelMin) / StepCount;
  for i := 0 to StepCount do begin
        TrelCurrent := TrelMin + TrelStep * i;
    Lc := (2 * PI)
                  /
        Sqrt(
          (G0 - Gmin)
          /
          (D * TrelCurrent * Gmin)
        );
    LineSeries.AddXY(TrelCurrent, Lc);
  end;
  Chart.AddSeries(LineSeries);
end;
```

Тут спочатку оголошується константа PI. Потім необхідні змінні (які є в формулі): *G0*, *Gmin*, *D*. Так як час релаксації τ_{rel} змінюватиметься, оголошуємо мінімальну *TrelMin*, максимальну *TrelMax*, крок TrelStep та поточну *TrelCurrent* змінні. Також кількість точок StepCount, сам період Lc, контейнер LineSeries для зберігання обрахованих даних та і для використання у циклі.

На початку коду ініціалізуються змінні (*D* береться із поля вводу), створюється LineSeries, обраховується крок TrelStep. Потім в циклі при кожній ітерації обраховується поточний час релаксації TrelCurrent, період дисипативної Lc структури і вони додаються у LineSeries як точка із координатами *(x, y)*. Після закінчення циклу заповнена серія додається до графіка Chart.

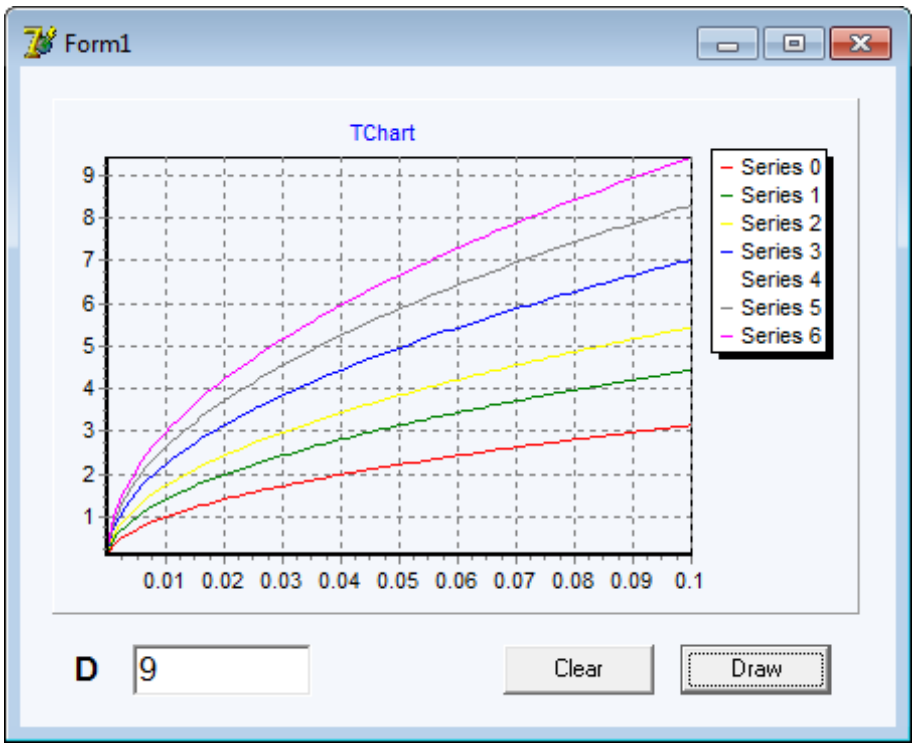


Рис. А.1. Вигляд форми програми для залежності $L_c(D)$

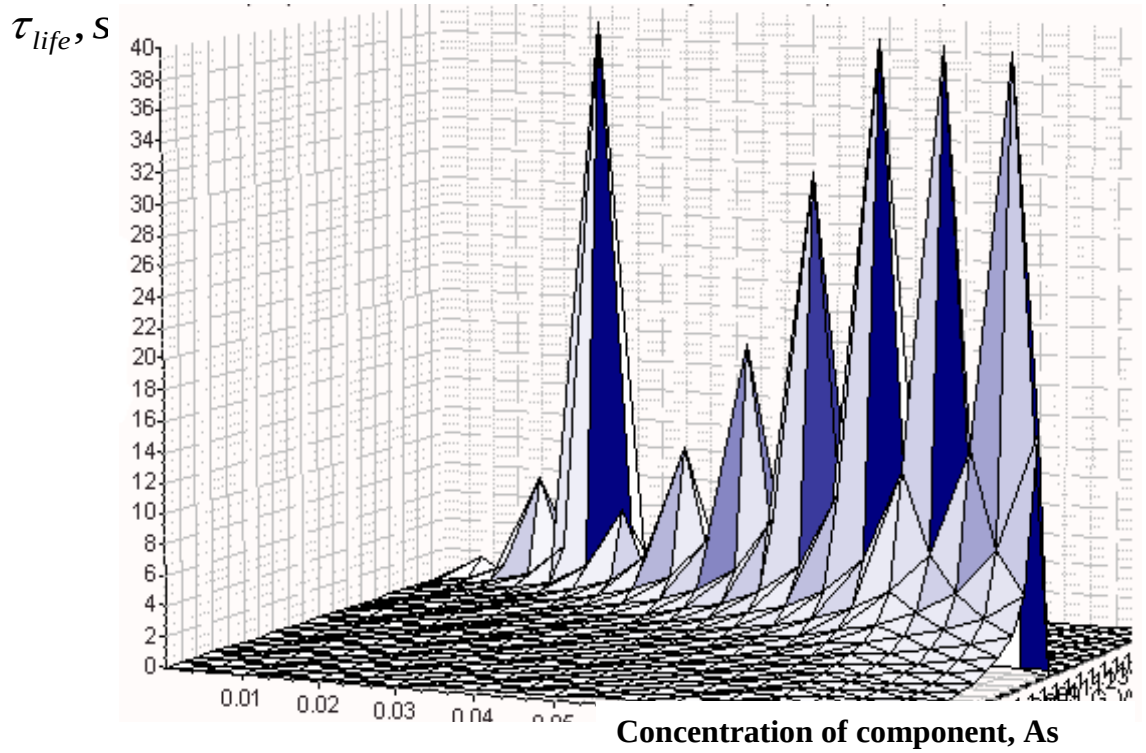


Рис. А.2. Залежність часу життя самоорганізованих структур від складу системи As-Se

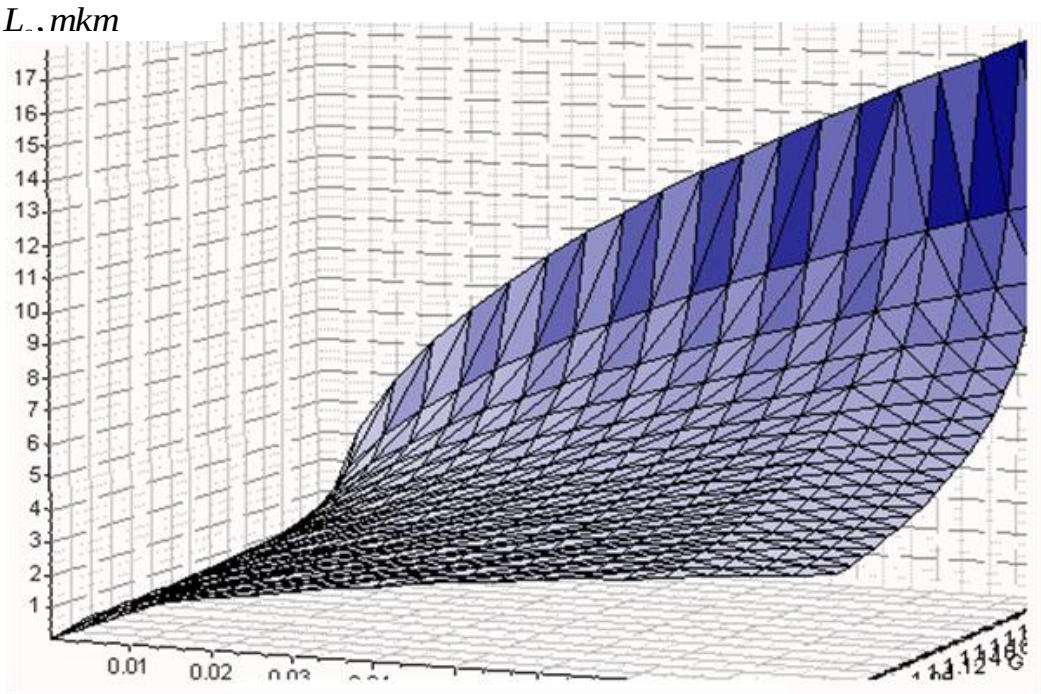
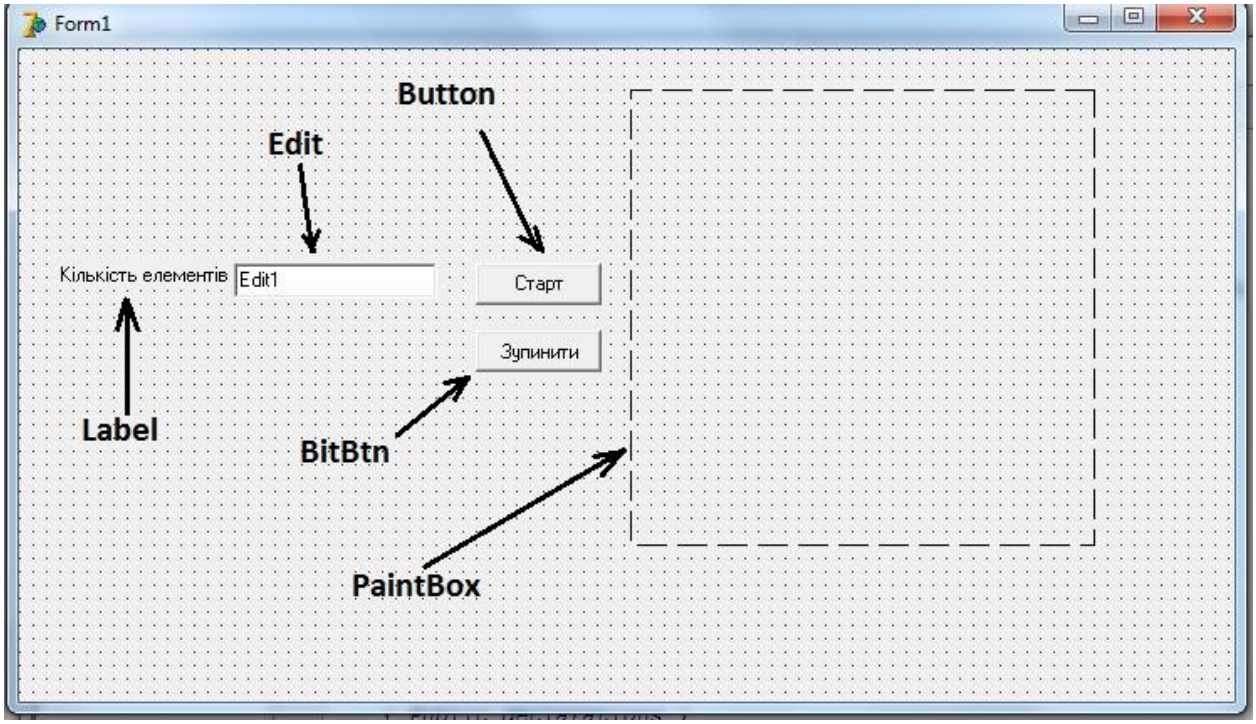


Рис. А.3. Залежність періоду самоорганізованих структур від складу системи As-Se

Додаток Б. Літінг коду програми та візуальний інтерфейс для моделі неперервної перколяції

Поставити на форму компоненти PaintBox, Button, BitBtn, Edit, Label, за допомогою Object Inspector внести потрібні зміни (форма, назва і т. д.).

Візуалізація форми для моделі неперервної перколяції:



Компоненти програми :

Компонента **Button** – це проста командна кнопка. Основна подія – OnClick виникає при клацанні на неї. Саме в обробці цієї події записуються оператори, які повинні виконуватися при клацанні користувача на кнопку.

Розміщена вона у вкладці Standart.

Кнопка **BitBtn** відрізняється від попередньої компоненти, насамперед, можливістю відображення на її поверхні зображення. Більшість властивостей, методів і подій у цих видів кнопок однакові.

Розміщена у вкладці Additional.

Компонента **Edit** являє собою однорядкове текстове поле, яке слугує для ввода тексту користувачем. Основною властивістю компонента **Edit**, яка передає введену інформацію є властивість Edit1.Text типу String.

Розміщена дана компонента у вкладці Standart.

Компонента **Label –** мітка, спеціально призначена для відображення тексту на формі. Розміщена вона у вкладці Standart.

Компонента **PaintBox** (вкладка System) класу TPaintBox використовується в тих випадках, коли необхідно мати прямокутну область для виконання графічних операцій, використовуючи властивість Canvas. При цьому можна

використовувати всі можливості Canvas - перо, пензлик , шрифт і методи побудови геометричних фігур. Але TPaintBox не дозволяє завантажувати готові зображення.

Два раза клікнути по компоненті **Button,** ввести код програми:

Unit1 | Project2

```
        procedure TForm1.Button1Click(Sender: TObject);
Var x,y,n,i: integer;
Const L=500; {задає розміри середовища, бажано більше 500}
begin
 n:=StrToInt (Edit1.Text);
 Randomize;
 With Form1.PaintBox1.Canvas do
 begin
 brush.color:= clBlack{колір фона ( в даному випадку чорний)};
 Rectangle (1,1,L,L); {задає форму середовища.(Rectangle з англ. прямокутник)}
 Pen.Color:=clWhite; {задає колір контуру елементів перколяції}
 brush.color:=clBlack; {задає колір елементів перколяції}
 for i:=1 to n do
 begin
 x:=1+round(random*L);
 y:=1+round(random*L);
 Ellipse ( x,y,x+10,y+10);{задає форму елементів перколяції}
 end;
end;

end;

procedure TForm1.BitBtn1Click(Sender: TObject);
begin
Close;
end;

End.
```

Додаток В. Лістінг коду програми нейронної мережі з алгоритмом Хебба в середовищі Java

Клас RecognizeNeuron

```
const int PIXELS = 64*64;
class RecognizeNeuron{
private:
  float weights[PIXELS];
```

```
  float totalWeights;
  float step;
public:
  RecognizeNeuron();
  void changeWeights(int[]);
  float activate(int[]);
};
RecognizeNeuron::RecognizeNeuron(){
  RecognizeNeuron::totalWeights = 0;
  for (int i=0; i<PIXELS; i++) {
        RecognizeNeuron::weights[i] = 1;
        RecognizeNeuron::totalWeights += RecognizeNeuron::weights[i];
  }
  RecognizeNeuron::step = 0.5;
}
void RecognizeNeuron::changeWeights(int inputVector[]){
  int changedWeightsNumber = 0;
  for (int i=0; i<PIXELS; i++) {
    if (inputVector[i] != 0){
          RecognizeNeuron::weights[i] += RecognizeNeuron::step;
      changedWeightsNumber++;
    }
  }
  RecognizeNeuron::totalWeights -= changedWeightsNumber *
RecognizeNeuron::step;
  float resultWeights = RecognizeNeuron::totalWeights / (PIXELS -
changedWeightsNumber);
  RecognizeNeuron::totalWeights = 0;
  for (int i=0; i<PIXELS; i++) {
        if (inputVector[i] == 0){
```

```
            RecognizeNeuron::weights[i] = resultWeights;
    }
        RecognizeNeuron::totalWeights += RecognizeNeuron::weights[i];
  }
}
float RecognizeNeuron::activate(int inputVector[]){
  float result = 0;
  for (int i = 0; i < PIXELS; i++) {
        result += RecognizeNeuron::weights[i]*inputVector[i];
  }
  return result;
}
```

Клас RecognizePerceptron

```
const int RECOGNIZE_PERCEPTRON_SIZE = 36;
class RecognizePerceptron{
private:
  RecognizeNeuron *neurons[RECOGNIZE_PERCEPTRON_SIZE];
public:
  RecognizePerceptron();
  ~RecognizePerceptron();
  int recognize(int[]);
  void teach(int[],int);
};
RecognizePerceptron::RecognizePerceptron(){
  for (int i=0; i<RECOGNIZE_PERCEPTRON_SIZE; i++) {
        RecognizePerceptron::neurons[i] = new RecognizeNeuron;
  }
}
RecognizePerceptron::~RecognizePerceptron(){
  for (int i=0; i<RECOGNIZE_PERCEPTRON_SIZE; i++) {
```

```
        delete RecognizePerceptron::neurons[i];
  }
}
int RecognizePerceptron::recognize(int inputVector[]){
  float result[RECOGNIZE_PERCEPTRON_SIZE] = {0};
  for (int i=0; i<RECOGNIZE_PERCEPTRON_SIZE; i++) {
    result[i] = RecognizePerceptron::neurons[i]->activate(inputVector);
  }
  float maxResult = -1;
  int resultSymbol = -1;
  for (int i=0; i<RECOGNIZE_PERCEPTRON_SIZE; i++) {
    if (maxResult < result[i]){
      maxResult = result[i];
          resultSymbol = i;
    }
  }
  return resultSymbol;
}
void RecognizePerceptron::teach(int inputVector[], int teachSymbol){
    RecognizePerceptron::neurons[teachSymbol]->changeWeights(inputVector);
}
```

Клас CodeNeuron

```
class CodeNeuron{
private:
  int value;
  int weights;
public:
  CodeNeuron();
  void setValue(int);
  void changeWeights(int);
```

```
  float activate(int);
  int getWeights();
};
CodeNeuron::CodeNeuron(){
  CodeNeuron::weights = 1;
  CodeNeuron::value = -1;
}
void CodeNeuron::setValue(int inputData){
  CodeNeuron::value = inputData;
}
void CodeNeuron::changeWeights(int inputData){
  CodeNeuron::weights = inputData;
}
float CodeNeuron::activate(int inputData){
  int result = -1;
  if (inputData == value) {
        result = CodeNeuron::weights*inputData;
  }
  return result;
}
int CodeNeuron::getWeights(){
  return CodeNeuron::weights;
}
```

Клас CodePerceptron

```
const int CODE_PERCEPTRON_SIZE = 5;
class CodePerceptron{
private:
  CodeNeuron *neurons[CODE_PERCEPTRON_SIZE];
public:
  CodePerceptron();
```

```
  ~CodePerceptron();
  int compare(int);
  bool blocked(int);
  void teach(int,int);
  void changeWeights(int,int);
};
CodePerceptron::CodePerceptron(){
  for (int i=0; i<CODE_PERCEPTRON_SIZE; i++) {
        CodePerceptron::neurons[i] = new CodeNeuron;
  }
}
CodePerceptron::~CodePerceptron(){
  for (int i=0; i<CODE_PERCEPTRON_SIZE; i++) {
        delete CodePerceptron::neurons[i];
  }
}
int CodePerceptron::compare(int inputData){
  int result[CODE_PERCEPTRON_SIZE] = {0};
  for (int i=0; i<CODE_PERCEPTRON_SIZE; i++) {
        result[i] = CodePerceptron::neurons[i]->activate(inputData);
  }
  int symbolIndex = -1;
  for (int i=0; i<CODE_PERCEPTRON_SIZE; i++) {
        if (result[i] > -1){
          symbolIndex = i;
          CodePerceptron::changeWeights(i, -1);
          break;
        }
  }
  return symbolIndex;
```

```
}
bool CodePerceptron::blocked(int limit){
  int index = 0;
  for (int i=0; i<CODE_PERCEPTRON_SIZE; i++) {
        if (CodePerceptron::neurons[i]->getWeights() == -1){
          index++;
        }
  }
  index  = (index*100)/CODE_PERCEPTRON_SIZE;
  if (index >= limit) {
        return true;
  } else {
        return false;
  }
}
void CodePerceptron::teach(int index, int value){
  CodePerceptron::neurons[index]->setValue(value);
}
void CodePerceptron::changeWeights(int index, int value){
  CodePerceptron::neurons[index]->changeWeights(value);
}
```

Додаток Г. Лістінг коду програми нейронної мережі Кохонена в середовищі Java

Клас Neuron.

Глобальні змінні:

```
private int[] w;              // вага синапсів
```

Конструктор. Параметр m - число входів:

```
Neuron(int m)
```

```
    {
        w = new int[m];
    }
```

Передаточна функція. Параметр x - вхідний вектор, повертається вихідне значення нейрона.

```
public int transfer(int[] x)
    {
        return adder(x);
    }
```

Суматор. Параметр x - вхідний вектор, повертається вага нейрона:

```
private int adder(int[] x)
    {
        int nec = 0;
        for (int i = 0; i < x.length; i++)
        {
            nec += x[i] * w[i];
        }
        return nec;
    }
```

Ініціалізація початкових ваг синапсів з використанням генератора випадкових чисел. Параметр n - максимальне випадкове значення:

```
public void initWeights(int n)
    {
        Random rand = new Random();
        for (int i = 0; i < w.length; i++) {
            w[i] = rand.nextInt(n);
        }
    }
```

Модифікація ваг синапсів для навчання. Параметр v - швидкість навчання, d - різниця між виходом нейрона і потрібним виходом, x - вхідний вектор.

```
public void changeWeights(int v, int d, int[] x)
   {
      for (int i = 0; i < w.length; i++)
      {
         w[i] += v*d*x[i];
      }
   }
```

Клас Perceptron.

Одношаровий n - нейронний перцептрон.

Глобальні змінні:

```
Neuron[] neurons;       // шар нейронів
int n, m;
```

Конструктор. Параметр n - число нейронів, m - число входів нейронів кожного прихованого шару.

```
public Perceptron(int n, int m)
   {
      this.n = n;
      this.m = m;
      neurons = new Neuron[n];
      for (int j = 0; j < n; j++)
         neurons[j] = new Neuron(m);
   }
```

Розпізнавання образу. Параметр x - вхідний вектор, повертається вихідний образ.

```
public int[] recognize(int[] x)
   {
      int[] y = new int[neurons.length];
```

```
        int max = 0;
        for (int j = 0; j < neurons.length; j++)
        {
            y[j] = neurons[j].transfer(x);
            if(max < y[j])
            {
                max = j;
            }
            y[j] = 0;
        }
        y[max] = 1;
        return y;
    }
```

Ініціалізація початкових ваг малим випадковим значенням псевдовипадкової функції.

```
public void initWeights()
    {
        for (int j = 0; j < neurons.length; j++)
        {
            neurons[j].initWeights(10);
        }
    }
```

Навчання перцептрона. Параметр x - вхідний вектор, y - правильний вихідний вектор.

```
public void teach(int[] x, int[] y)
    {
        int d;
        int v = 1; // швидкість навчання
        int[] t = recognize(x);
        while (!equal(t, y))
```

```
        {
            // налаштування ваг кожного нейрона
            for (int j = 0; j < neurons.length; j++)
            {
                d = y[j] - t[j];
                neurons[j].changeWeights(v, d, x);
            }
            t = recognize(x);
        }
    }
```

Клас Teacher.

Вчитель. Вчить перцептрон розпізнавати цифри.

Глобальні змінні:

```
private Perceptron perceptron;
```

Конструктор.

```
public Teacher(Perceptron perceptron)
    {
        this.perceptron = perceptron;
    }
```

Навчання перцептрона. Параметр path - шлях до зображень, на базі яких навчається мережа, n - кількість циклів навчання:

```
public void teach(String path, int n)
    {    ...
        // завантаження всіх тестових зображень в масив img[]
        String[] list = new File(path + "/").list(new JPGFilter());
        Image[] img = new Image[list.length];
        String fileName;
        for (int i = 0; i < list.length; i++)
        {
            try
```

```
    {
        fileName = path + "/" + list[i];
        img[i] = read(new File(fileName));
        System.out.println(fileName);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

// ініціалізація початкових ваг
perceptron.initWeights();
// отримання піксельного масиву кожного зображення
// і навчання n разів в кожній вибірці
int[][] pixel;
int[] x = new int[64*64], y;
int p = n;
while (n-- > 0)
{
    for (int j = 0; j < img.length; j++)
    {
        pixel = getInVector(img[j]);
        for (int t = 0, i = 0; t < 64; t++)
        {
            for (int k = 0; k < 64; k++)
            {
                x[i++] = pixel[k][t];
            }
        }
        y = getOutVector(Integer.parseInt(String.valueOf(list[j].charAt(0))));
        perceptron.teach(x, y);
```

```
        }
        System.out.println((((double)n)/((double)p)) * 100 + " %");
    }
}
```

Трансформація зображення в вектор з нулів та одиниць: 1 на тому місці, де є колір, 0 - там де колір білий. Параметр p - пікселі зображення, повертає вектор для входу перцептрона:

```
protected int[][] getInVector(final Image image)
{
    int[][] points = new int[64][64];
    BufferedImage bi = (BufferedImage) image;
    Raster raster = bi.getRaster();
    ColorModel model = bi.getColorModel();
    Object data;
    Color color;
    int argb;
    for(int i = 0; i<64; i++)
    {
        for(int j = 0; j<64; j++)
        {
            data = raster.getDataElements(i, j, null);
            argb = model.getRGB(data);
            color = new Color(argb, true);
            if(color.getRed() < 125 || color.getBlue() < 125 || color.getGreen() < 125)
                points[i][j] = 1;
            else
                points[i][j] = 0;
        }
    }
```

```
    return points;
  }
```

Генерація правильного вихідного вектора. Параметр n - цифра, у відповідності з якою потрібно побудувати вектор з міткою одиниця (1), а на інших (0), та повертає значення вихідного вектора перцептрона. private int[] getOutVector(int n)

```
  {
    int[] y = new int[perceptron.getN()];
    for (int i = 0; i < y.length; i++)
    {
      if (i == n) y[i] = 1; else y[i] = 0;
    }
    return y;
  }
```

Клас GBC.

Клас, який наслідує менеджер компонування GridBagConstraints і допомагає компонувати об'єкти інтерфейсу.

Клас PerceptronFrame.

Клас, який наслідує клас JFrame і являє собою головний фрейм програми.

Ініціалізація всіх елементів інтерфейсу та розміщення їх на фреймі.

private void init() {...}

Знаходження єдиного одиничного елемента вектора. Параметр y - вектор з єдиним одиничним елементом, повертає порядковий номер одиничного елемента:

```
private int getDigit(int[] y)
  {
    int x = 0;
    for (int i = 0; i < y.length; i++)
    {
      if (y[i] == 1)
```

```
        {
            x = i;
        }
    }
    return x;
}
```

Викликає метод recognize() класу Perceptron, щоб розпізнати образ. Повертає число нейрона з найбільшою масою вагів:

```
private int recognize()
{
    int[] x = new int[points.length * points.length];
    for (int i = 0, k = 0; i < points.length; i++)
    {
        for (int j = 0; j < points.length; j++)
        {
            x[k++] = points[j][i];
        }
    }
    return getDigit(perceptron.recognize(x));
}
```

Слухачі кнопок.

Слухач кнопки "Навчатися":

```
public void actionPerformed(ActionEvent e)
        {
            int n = Integer.parseInt(spin.getValue().toString().trim());
            Teacher t = new Teacher(perceptron);
            t.teach("data", n);
        }
```

Слухач кнопки "Розпізнати":

```
public void actionPerformed(ActionEvent e)
        {
            int d = recognize();
            result.setText(Integer.toString(d));
            for(int i = 0; i<64; i++)
            {
                for(int j = 0; j<64; j++)
                {
                    System.out.print(points[j][i]);
                }
                System.out.println();
            }
        }
```

Слухач кнопки "Очистити":

```
public void actionPerformed(ActionEvent e)
        {
            image.getGraphics().setPaintMode();
            image.getGraphics().setColor(Color.WHITE);
            image.getGraphics().fillRect(0, 0, ic.getWidth(), ic.getHeight());
            ic.repaint();
            for (int i = 0; i < points.length; i++)
            {
                for (int j = 0; j < points.length; j++)
                {
                    points[i][j] = 0;
                }
            }
        }
```

Слухач кнопки "Відкрити":

```
public void actionPerformed(ActionEvent e)
        {
           chooser.setCurrentDirectory(new File("."));
           int result = chooser.showOpenDialog(ic);
           if(result == JFileChooser.APPROVE_OPTION)
           {
              String filename = chooser.getSelectedFile().getPath();
              try
              {
                 System.out.println(filename);
                 image = read(new File(filename));
              } catch (IOException ex) {
                 System.err.println("не получилося =(");
                 System.err.println(ex);
              }
              ic.repaint();

              points = tch.getInVector(image);
           }
        }
```

ЛІТЕРАТУРА

Аксенов, С.В., Новосельцев, В.Б. (2006). *Организация и использование нейронных сетей (методы и технологии)* /Под общ. ред. В.Б. Новосельцева. – Томск: Изд-во НТЛ, 128 С.

Бобик, М.Ю., Іваницький, В.П., Ковтуненко, В.С., Сватюк, О.Я. (2012). Кількісне визначення контрасту електронномікроскопічних зображень аморфних наноматеріалів складного хімічного складу. *Журнал нано- та електронної фізики*, **4 (**2), 02041-1 – 02041-5.

Вороновский, Г. К., Махотило, К. В., Петрашев, С. Н., Сергеев С. А. (1997). *Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности*. — Харьков: Основа, 297 С.

Дебок, Г., Кохонен, Т. (2001). *Анализ финансовых данных с помощью самоорганизующихся карт*. - Альпина Паблишер, 224 С.

Ежов, А.А., Шумский, С.А. (1998). *Нейрокомпьютинг и его применения в экономике и бизнесе*. – М.: МИФИ, 1998, 348 С.

Іваницький, В.П. (2011). Теорія формування контрасту електронних мікросистем. *Наноматеріали, наносистеми, нанотехнології*, № 3, 793-802.

Каллан, Р. (2003). *Основные концепции нейронных сетей*. – М.: ООО «И.Д. Вильямс», 287с.

Каширина, И.Л. (2005). *Искусственные нейронные сети*. - Воронеж: Изд-во ВГУ, 51с.

Круглов, В.В., Борисов, В.В. (2002). *Искусственные нейронные сети. Теория и практика*. – М.: Телеком. – 382с.

Люгер, Ф. (2003). *Искусственный интелект: стратегии и методы решения сложных проблем.* – М.: Вильямс, 204 С.

Рутковская, Д., Пилиньский, М., Рутковский, Л. (2007). *Нейронные сети, генетические алгоритмы и нечеткие системы*. – М.: Горячая линия – Телеком. – 452 С.

Уоссермен, Ф. (1992). *Нейрокомпьютерная техника*. – М.: «Мир», 240с.

Хайкин, С. (2006). *Нейронные сети*.– М.: ООО «И.Д. Вильямс», 1104 С.

Ясницкий, Л.Н. (2008). *Введение в искусственный интеллект*. М.: Издательский центр «Академия». – 176 С.

Adam, J.L., & Zhang, X. (2014). *Chalcogenide Glasses: Preparation, Properties and Applications. - Elsevier Science, 234 P.*

Gould, H. & Tobochnik, J. (1988) *An Introduction to Computer Simulation Methods Applications to Physical* Systems. - NY: Addison-Wesley Publishing Company, 1988, 749 P.

Haken, H. (2006). *Information and Self-Organization: a Macroscopic Approach to Complex Systems*. - Berlin: Springer, 257 P.

Kohonen, T. (2001). *Self-Organizing Maps*. - New York: Springer, 2001, 202 P.

Mar'yan, M I., Kikineshy, A.A., & Misak, A.A. (1993). Photoinduced instabilities in chalcogenide glasses for optical recording media. *Philosophical Magazine B: Physics of Condensed Matter; Statistical Mechanics, Electronic, Optical and Magnetic Properties*, 68(5), 689-695.

Mar'yan, M.I., Szasz, A., Szendro, P., & Kikineshy, A.(2005). Synergetic model of the formation of non-crystalline structures. *Journal of Non-Crystalline Solids*, 351(2), 189-193.

Mar'yan, M., & Yurkovych, N. (2013).The processes of self-organization and the formation of fractals in non-crystalline solids. *Scientific Herald of Uzhhorod University. Series Physics 34, 40-47.*

Mar'yan, M., & Yurkovych, N. (2014). Influence on disorder structure of the non-crystalline materials and synergetic effects. *Scientific Herald of Uzhhorod University. Series Physis, 35, 17-24.*

Mar'yan M., Seben V. & Yurkovych N. (2018). *Synergetics and Fractality in Science Education*. - Presov: University of Presov in Presov Publishing, 168 P.

Mar'yan, M., Seben, V. & Yurkovych, N. (2018). *Synergetics of Computer Modeling, Research and Technology Education. Concepts&Methods.* - Saarbruecken: LAP Lambert Academic Publishing, 120 P.

Mar'yan, M. & Yurkovych, N. (2018). *Fractality of the Non-Crystalline Solids and other Systems. Synergetics&Self-Organization*. - Saarbruecken: LAP Lambert Academic Publishing, 116 P.

Mar'yan, M. & Yurkovych, N. (2019). *Self-Organized Structures in Non-Crystalline Solids and other Systems. Methods, Concepts and Applications to the Information Technology*. - Saarbruecken: LAP Lambert Academic Publishing, 132 P.

Mar'yan, M. & Yurkovych, N. (2019). *Synergetics of the Instability and Randomness. Non-Crystalline Solids, Self-Organized Structures and Information Technology*. - Saarbruecken: LAP Lambert Academic Publishing, 112 P.

Mar'yan, M., Yurkovych, N. & Seben, V. (2019). Nanosized levels of the self-organized structures in the non-crystalline semiconductors As-S(Se) system**.** *Semiconductor physics, quantum electronics and optoelectronics*, **22** (3), 299-309.

Mar'yan, M., Yurkovych, N. & Seben, V . (2019). Self-organized structures and fractality of the non-crystalline materials As(Ge)-S(Se) systems. - 24th Conference of Slovak Physicists. Zhilina, September, 2-5, 2019 (Slovakia).

Nicolis, G., & Prigogine, I. (1989). *Exploring Complexity. An introduction*. – New York: Freeman, 344 P.

Prigogine, I. (1980). *From Being to Becoming: Time and Complexity in Physical Sciences.* – San-Fransisco: Freeman. 380 P.

Yurkovych, N., Seben, V. & Mar'yan, M. (2017). *Computer Modeling and Innovative Approaches in Physics: Optics. - Presov: Prešovska univerzita v Prešove, 114 P.*

Yurkovych, N., Seben, V. & Mar'yan, M. (2017). Fractal approach to teaching physics and computer modeling. *Journal of Science Education*, 18(2), 117-120.

Yurkovych, N., Mar'yan, M. & Seben, V. (2018). Synergetics of the instability and randomness in the formation of gradient modified semiconductor structures. *Semiconductor physics, quantum electronics and optoelectronics*, 21 (4), 365-373.

Yurkovych, N., Mar'yan, M. & Seben, V. (2019). Synergetics of the science education and innovative teaching. - 24th Conference of Slovak Physicists. Zhilina, September, 2-5, 2019 (Slovakia).

Анотація

Розглядаються інноваційні технології інтелектуальних систем – нейронні мережі та об’єктно-орієнтоване моделювання, синергетика їх застосування при дослідженні фізичних та інформаційних процесів. Детально проаналізовано принципи функціонування нейронних мереж (прямого та зворотного поширення, нейронних мереж, побудованих на принципах самоорганізації), їх адекватність та доцільність при викладанні природничих наук. Розроблені та приведені об’єктно-орієнтовані застосунки для ряду фізичних моделей та інформаційних процесів на алгоритмічних мовах програмування Object Pascal, C++, Java.

Summary

Innovative technologies of intellectual systems - neural networks and object-oriented modeling, synergetics of their application in the study of physical and informational processes are considered. The principles of neural networks functioning (direct and back propagation, neural networks built on the principles of self-organization), their adequacy and expediency in the teaching of natural sciences are analyzed in detail. Object-oriented applications for series of physical models and information processes in algorithmic programming languages Object Pascal, C ++, Java are developed and presented.

Vysvetlivka

Inovatívne technológie intelektuálnych systémov - neurónové siete a objektovo orientované modelovanie, synergie ich aplikácie v štúdiu fyzikálnych a informačných procesov. Podrobne sa analyzujú princípy fungovania neurónových sietí (priame a spätné šírenie, neurónové siete postavené na princípoch samoorganizácie), ich primeranosť a účelnosť vo vyučovaní prírodných vied. Objektovo orientované aplikácie pre množstvo fyzikálnych modelov a informačných procesov v algoritmických programovacích jazykoch Object Pascal, C ++, Java sú vyvíjané a prezentované.

ПРЕДМЕТНИЙ ПОКАЖЧИК

Р

С

Т

У

Ф

Х

Ч

Ш

АБРЕВІАТУРА

ЕМ – Електронна Мікроскопія

ІСП – Інтегроване Середовище Програмування

ІТ - Інформаційні Технології (Information Technology - IT)

ІКТ - Інформаційно-Комунікаційні Технології та Системи (Information and Communication Technologies - ICT)

НМ – Нейронна Мережа

НКТ – Некристалічні Тіла

ООМ – Об’єктно-Орієнтоване Моделювання

ООП – Об’єктно-Орієнтоване Програмування

СВВ - Система Виявлення Вразливостей

СОА - Система Оповіщення Атаки

СОІС – Самоорганізовані Інформаційні Структури

ФРРА – Функція Радіального Розподілу Атомів

ШНМ - Штучні Нейронні Мережі

Authors and CV

Mykhaylo Mar'yan, Doctor of Sciences (Physics & Mathematics), Professor of the Department Solid-State Electronics & Information Security Faculty of Physics Uzhhorod National University, Uzhhorod, Ukraine. **Scientific focus on the researches of self-organization processes in the systems of different nature (non-crystalline state, fractal structures, information and stochastic systems) and innovative technologies for addressing informational, environmental problems are placed.**

E-mail: ***mykhaylo.maryan@uzhnu.edu.ua***
https://orcid.org/0000-0002-1667-4945

Nataliya Yurkovych, PhD (Physics & Mathematics), Ass. Prof. of the Department of Solid-State Electronics & Information Security Faculty of Physics Uzhhorod National University, Uzhhorod, Ukraine. **The area of scientific interests covers computer modeling in the diagnostics, formation, and modifications of different kinds of the thin film gradient modified semiconductor structures.**

E-mail: nataliya.yurkovych@uzhnu.edu.ua
https://orcid.org/0000-0001-7686-9667

Vladimir Seben, PhD, Head of the Department of Physics, Mathematics & Technics Faculty of Humanities and Natural Science University of Presov, Presov, Slovakia. **A sphere of the scientific interests is associated with study of didactics and to the development of innovative information products of teaching physics.**

E-mail: ***vladimir.seben@unipo.sk***

Mykhaylo Mar'yan, Vladimir Seben, Nataliya Yurkovych. ***Innovative Technologies of Computer Modeling for Physical and Information Processes. Synergetics of Information and Communication Systems.*** **- Presov: University of Presov Publishing, 2019. 120 P.**

Authors / Autori: **Prof. Mykhaylo Mar'yan, DrSc**./
Prof. Michajlo Marian, DrSc.
Dr.h.c. doc. PaedDr. Vladimir Seben, PhD./
Dr.h.c.doc. PaedDr. Vladimír Šebeň, PhD.
Ass Prof. Nataliya Yurkovych, PhD./
Doc. Natalia Jurkovič, PhD.

Reviewers / Recenzenti: Prof. Vasyl Rubish, DrSc.
Prof. Oleksandr Grabar, DrSc.

Vydala: © Prešovská univerzita v Prešove, 2019
Vydanie: Prvé

ISBN: 978-80-555-2278-4

TheStartPointAhead

The Synergetics of the Non-Crystalline Solids, Intellectual Materials and Information & Comunication systems