Міністерство освіти й науки України
Донбаська державна машинобудівна академія (ДДМА)

Методичні вказівки

до ком‘ютерного практикуму по дисципліні "Сучасні методи дослідження систем"

для студентів спеціальності 151 «Автоматизація та комп‘ютерно-інтегровані системи» усіх форм навчання

Затверджено
на засіданні методичної ради
Протокол № 2 від 03,12,2020 р.

Краматорськ
ДДМА
2020

УДК 001.891.5

Методичні вказівки до комп‘ютерного практикуму по дисципліні «Сучасні методи дослідження систем» для студентів спеціальності 151 «Автоматизація та комп‘ютерно-інтегровані системи» / уклад. О. О. Сердюк. – Краматорськ : ДДМА, 2020. – 40 с.

Викладені рекомендації з моделювання об'єктів фрактальної структури й процесів хаотичної динаміки, а також методика застосування пакета прикладних програм Neural Network Toolbox у середовищі MATLAB для моделювання поведінки динамічних систем.

Укладач О. О. Сердюк, доц.

Відп. за випуск Г. П. Кліменко, зав. кафедри

ЗМІСТ

ВСТУП

Особливістю наукових проблемних ситуацій, що зустрічаються в темах магістерських робіт, є складність формалізації досліджуваних процесів, тобто їх математичного опису. В багатьох технологічних системах ця складність пов‘язана із нелінійністю динамічних систем.

Існуюча теорія нелінійних систем дозволяє одержати тільки часткові розв'язки у вигляді фазових траєкторій на площині. Але в останні десятиріччя спостерігається активний розвиток нових теорій дослідження нелінійних динамічних систем, які дозволяють створювати більш точний опис поведінки нелінійних об’єктів.

Дані методичні вказівки спрямовані на формування навичок у студентів щодо використання новітніх методів аналізу поведінки нелінійних динамічних систем з використанням програмних засобів.

Методичиі вказівки призначені для студентів денної та заочної форм навчання зі спеціальності 151 «Автоматизація та комп‘ютерно-інтегровані системи».

Методичні вказівки підготовлені на основі нормативно-правових актів України у сфері освіти, державних стандартів, освітнього стандарту вищої професійної освіти та відповідно до Положення про організацію навчального процесу у Донбаській державній машинобудівній академії.

1 МОДЕЛЮВАННЯ ГЕОМЕТРИЧНИХ ФРАКТАЛІВ

1.1 Принципи побудови зображень фрактальних об‘єктів

Ціль роботи: освоїти методику моделювання фрактальних об'єктів, описуваних алгебраїчними вираженнями.

Для побудови алгебраїчних фракталів використовуються ітерації нелінійних відображень, що задаються простими алгебраїчними формулами.

Найбільш вивченим являється двомірний випадок. Нелінійні динамічні системи можуть мати декілька стійких станів. Кожний стійкий стан (атрактор) має деяку область початкових станів, при яких система обов'язково в них перейде. Таким чином, фазовий простір розбивається на області притягання атракторів.

Якщо фазовий простір є двомірним, то, забарвлюючи області притягання різними кольорами, можна одержати кольоровий фазовий портрет цієї системи (ітераційного процесу). Міняючи алгоритм вибору кольору, можна одержати складні фрактальні картини з вигадливими багатобарвними візерунками. При цьому існує можливість за допомогою примітивних алгоритмів породжувати дуже складні нетривіальні структури.

Так, наприклад, на комплексній площині можна побудувати фрактал Мандельбротта із застосуванням простого вираження:

$$z_{n+1} = f(z_n) = z_n^2 + c. \qquad (1.1)$$

де $c = x + iy$.

Цей фрактал наведено на рисунку 1.1.

Для всіх точок прямокутної області на комплексній площині обчислюємо досить велику кількість раз *zn+1,* При цьому значення функції для різних точок комплексної площини можуть мати різну поведінку.

Із часом можливі наступні варіанти зміни *zn+1*

- | *zn+1*| прямує до нескінченності;
- | *zn+1*| прагне до 0;
- | *zn+1*| ухвалює кілька фіксованих значень і не виходить за їхні межі;

У загальному випадку поведінка |zn+*1*| хаотична, без яких-небудь тенденцій.

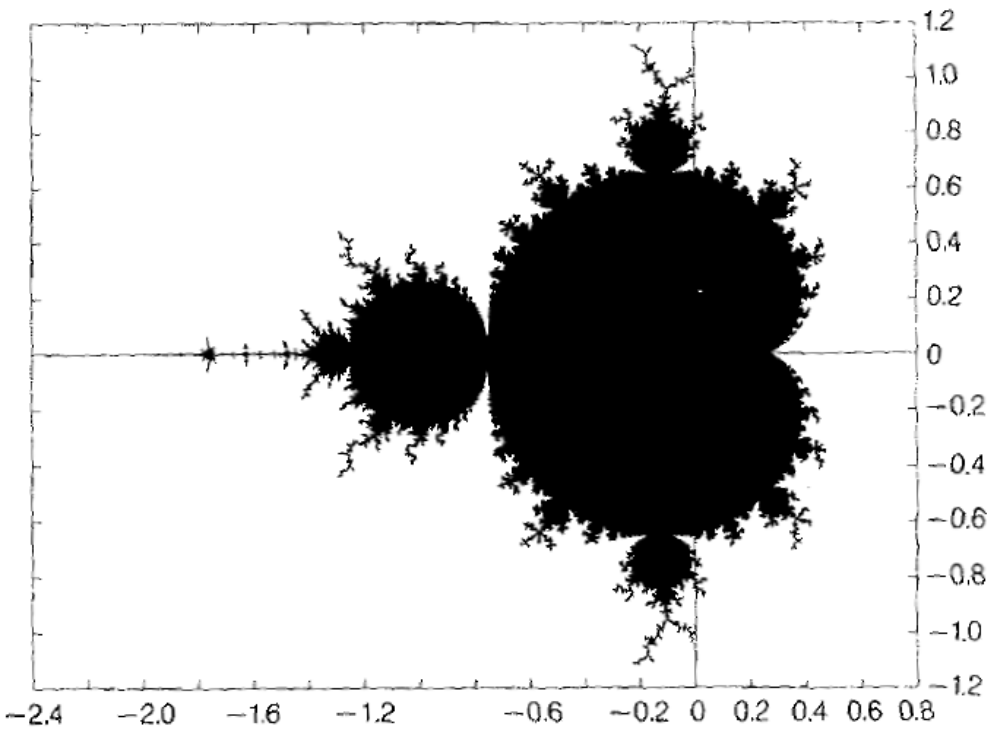


Рисунок 1.1 – Фрактал Мандельброта

1.2 Використання існуючих програм для створення фрактальних об'єктів

Галерея фракталів представлена на сайті за адресою http://aquale.narod.ru/photo/. На цьому сайті зібрані також посилання на завантаження самих популярних фрактальних програм та існуючі найбільш корисні матеріали по них, включаючи керівництва.

Отже, для створення фрактальних об‘єктів пропонується програма Apophysis версія 7X16**,** яка доступна для завантаження отут: http://sourceforge.net/projects/apophysis7x.

Існує також стара версія програми 2.02 знайти її можна по цій адресі: http://www.apophysis.org, а також версія 2.09: http://sourceforge.net/projects/apophysis/.

Плагіни для Apophysis доступні отут:

http://fractal-resources.deviantart.com/favourites/1009556.

Ще один вільно розповсюджуваний генератор 3D фракталів – програма Incendia (http://www.incendia.net/).

Приклад створення найпростішої анімації в цій програмі (англійською мовою) можна вивчити на сторінці http://www.incendia.net/wiki/index.php/Very_Simple_Animation_Tutorial

Деякі особливості цієї програми: в програмі доступні дозволи зображення 1024, 2048, 2560 і 3072 px. Підтримується 2X згладжування, пред-

установлено 45 типів фракталів 3D, Підтримуються скрипти, є бібліотека текстур, також можливий імпорт зовнішніх зображень, створених у форматі bmp, зручна робота із градієнтом.

На додаток до цієї програми варто згадати пов'язану з нею в одному архіві програму Geometrica.

Обидві програми працюють у зв'язуванні, оскільки по суті Geometrica — це генератор сітки, що дозволяє експортувати фрактали з Incendia.

Фрактальний об'єкт імпортується у своєму індивідуальному форматі, який перетворюється в obj для використання в інших програмах, Прикладом можуть послужити Blender або Bryce. У програмі доступні до зміни дозвіл сітки й кількість ітерацій. Крім того Geometrica підтримує створення анімірованих сіток від Incendia анімації.

Заслуговує уваги також програма Fractal Explorer**.** Сторінка завантаження версії 2.02 знаходиться за адресою http://www.eclectasy.com/Fractal-Explorer/download.htm.

Даний дослідник фракталів є безкоштовним фрактальним генератором. Він може створювати класичні поліноміальні фрактальні множини (наприклад, набори Mandelbrot-set, the Julia-set, the Newton-set і їх варіації), набір з 22 квартеронів (4д комплексні фрактали), 3D attractors і IFS. Крім того, FE має множину функцій для створення спецефектів і поліпшення зображень у постобробці отриманих фракталів. Має також можливість створення Landscapes (земних поверхонь).

Остання версія програми 2.02 була випущена в 2005 році. Незважаючи на розвиток і виникнення усе більш досконалих, нових фрактальних програм, вона не втратила свого значення.

Посібник з роботи із цією програмою перебуває за адресою - http://vasnyov.narod.ru/fe.htm

Для генерації атарактoрів можна також застосовувати програму Chaoscope. Це вільно розповсюджувана програма, сторінка завантаження http://www.chaoscope.org/download.htm. Автор програми Nicolas Desprez.

Ця програма є генератором дивних атракторів, тому атрактори мають фрактальну розмірність. Програма працює в середовищі Windows.

Дуже добре написане керівництво для початківців можна вивчити на деміарті http://demiart.ru/forum/index.php?showtopic=149337.

Програма Xaos 3.5 доступна для скачування за адресою http://sourceforge.net/projects/xaos/files/, а мануал по програмі на англійській мові можна подивитися тут: http://xaos.sourceforge.net/doc-trunk.

Програма поширюється безкоштовно під ліцензією GPL. Спочатку вона була написана Thomas Marsh і Jan Hubicka, зараз підтримується Zoltan Kovacs і J.B. Langston.

Генератор дозволяє зумірувати фрактал у реальному часі. Така можливість дозволяє детально досліджувати фрактал, поринаючи в самі глибини. Можна створювати 2-х і 3-х мірні фрактали, а також анімацію. У комплект, що завантажується зі сторінки сайту, входять анімірovані підручни-

ки, з їхньою допомогою досить легко вчитися працювати із програмою.

Xaos може відображати фрактали різних типів, включаючи Mandelbrot, Barnsley, Newton, Phoenix а так само інші. Xaos також підтримує перемикання між Джулією і Мандельбротом, ця опція встановлюється окремо для кожної формули.

На рисунку 1.2 показаний приклад відображення фрактала у вікні програми.

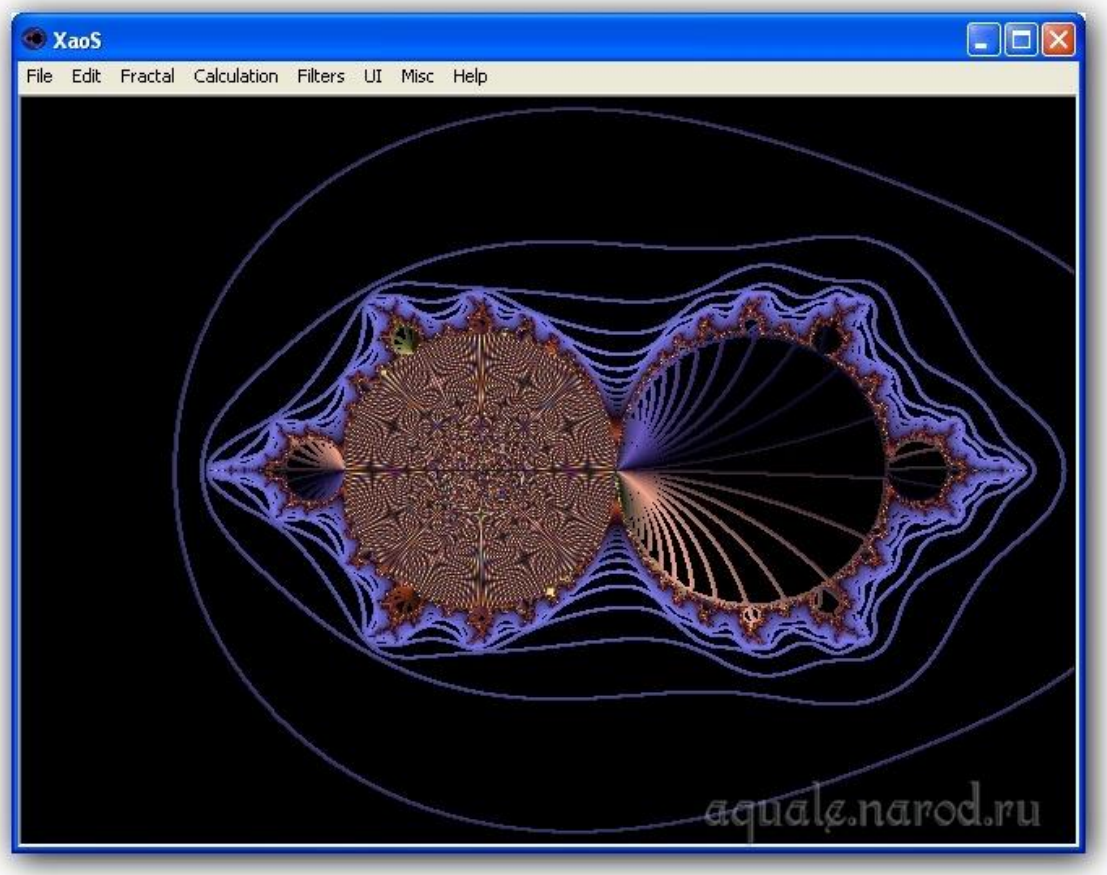


Рисунок 1.2 – Відображення фрактала у вікні програми Xaos

У програмі є можливість завантажити готові приклади, для цього в меню "File" треба вибрати "Load random example".

1.3 Порядок виконання й зміст звіту по роботі

Студент може самостійно вибрати програму та фрак тал, що сподобався.

Звіт повинен містити:

1. Вихідну інформацію й математичний опис фрактала.
2. Скріншот настроювань програми.
3. Зображення фракталів при зміні основних параметрів.
4. Висновки по роботі.

При захисті звіту студент повинен продемонструвати роботу програми й відповісти на запитання, що стосуються теорії фракталів і методики їх моделювання.

2 СТВОРЕННЯ ФРАКТАЛЬНИХ ОБ'ЄКТІВ ІЗ ВИКОРИСТАННЯМ TURTLE-ТЕХНОЛОГІЇ

2.1 Сутність turtle-технології

Класичні фрактали мають єдиний принцип побудови – додавання або викидання окремих ліній або областей. Цей процес повторюється багаторазово *(ітераційно)*.

Досить велику групу самоподібних фракталів (фрактальні дерева, рослини, русла рік і т буд.) можна створювати за допомогою так званих L-*систем*, у яких використовується підсистема графічного висновку за назвою тертл-*графіка* (від англійського turtle – черепаха).

Сутність тертл-*графіки* полягає в тому, що зображуючи точка (черепашка) рухається по екрану монітора прямолінійно, дискретними кроками, залишаючи або не залишаючи свій слід. Після кожного переміщення вона може повернутися на деякий кут у ту, або іншу сторону, або продовжити рух знову по прямій. Так утворюється безперервна або розривна дискретна лінія на екрані. Зображуюча точка може повернутися на кілька кроків назад, не перериваючи свій слід, і почати рух у новому напрямку. У цьому випадку відбувається розгалуження траєкторії руху.

Зображуюча точка рухається по командах, що задаються кодовими словами. У кожній точці екрана положення точки задається трьома *параметрами x, y,* α, *(x, y* – координати точки α – кут, що визначає напрямок руху). Кодове слово складається із вказівок переміщення на один крок із залишенням або незалишенням сліду, збільшенням або зменшенням напрямку руху на деякий кут Θ, відкриттям гілок, закриттям гілок.

L-систему утворюють алфавіт, ініціатор (слово ініціалізації, аксіома) і набір породжуючи правил, які визначають перетворення аксіоми для організації ітераційного процесу. Алфавіт складається з набору окремих символів. Кожний символ являє собою мікрокоманду, що пропонує певну дію, виконувану зображуючою точкою.

Наприклад:

F – переміститися вперед на один крок, прорисовуючи слід;

b - переміститися вперед на один крок, не прорисовуючи слід;

[– відкрити гілку;

] – закрити гілку;

+ – збільшити кут α на величину Θ;

– – зменшити кут α на величину Θ.

З елементів алфавіту можна створювати слова ініціалізації (аксіоми). Наприклад, L -система, що дозволяє намалювати на екрані рівносторонній трикутник, має такі елементи:

$$\Theta = \frac{\pi}{3},$$

аксіома: $F + +F + +F$.

Зображуюча точка має первісний напрямок руху під кутом $\frac{\pi}{3}$. Згідно з командою F виконується рух на один крок. По команді + і + здійснюється поворот на кут $2\frac{\pi}{3}$. Наступна команда F пропонує рух ще на один крок. Остаточна команда F замикає трикутник.

Породжуюче правило призначене для заміни мікрокоманди в аксіомі групою мікрокоманд. Наприклад, якщо в наведеній вище аксіомі команду F замінити правилом $newf = F - F + +F - F$, то зображуючи точка при русі по екрану намалює сніжинку Коха.

Породжуюче правило – це різновид рекурсивної процедури. Глибина рекурсії показує, яку кількість ітерацій по заміні мікрокоманд групою мікрокоманд необхідно виконати.

За допомогою мікрокоманди розгалуження здійснюється побудова дерев і рослин. Породжуючі правила дозволяють виконувати розгалуження багаторазово не тільки від лінії основного напрямку руху зображуючої точки, але й від побудованих раніше гілок.

2.2 Методика виконання роботи

На рисунку 2.1 показаний інтерфейс програми для on-line побудови фракталів методом turtle-графіки, яка представлена на сайті http://www.kevs3d.co.uk/dev/lsystems/.

Нехай, наприклад, необхідно побудувати фрактал Дерево Піфагора. Для даного прикладу черепасі можуть бути дані наступні команди:

- 0: малювати відрізок, що закінчується листом;
- 1: малювати відрізок без листа;
- [: покласти в стек положення й кут малювання та повернути вліво на 45 градусів;

-]: вибрати зі стека положення й кут, повернути вправо на 45 градусів.

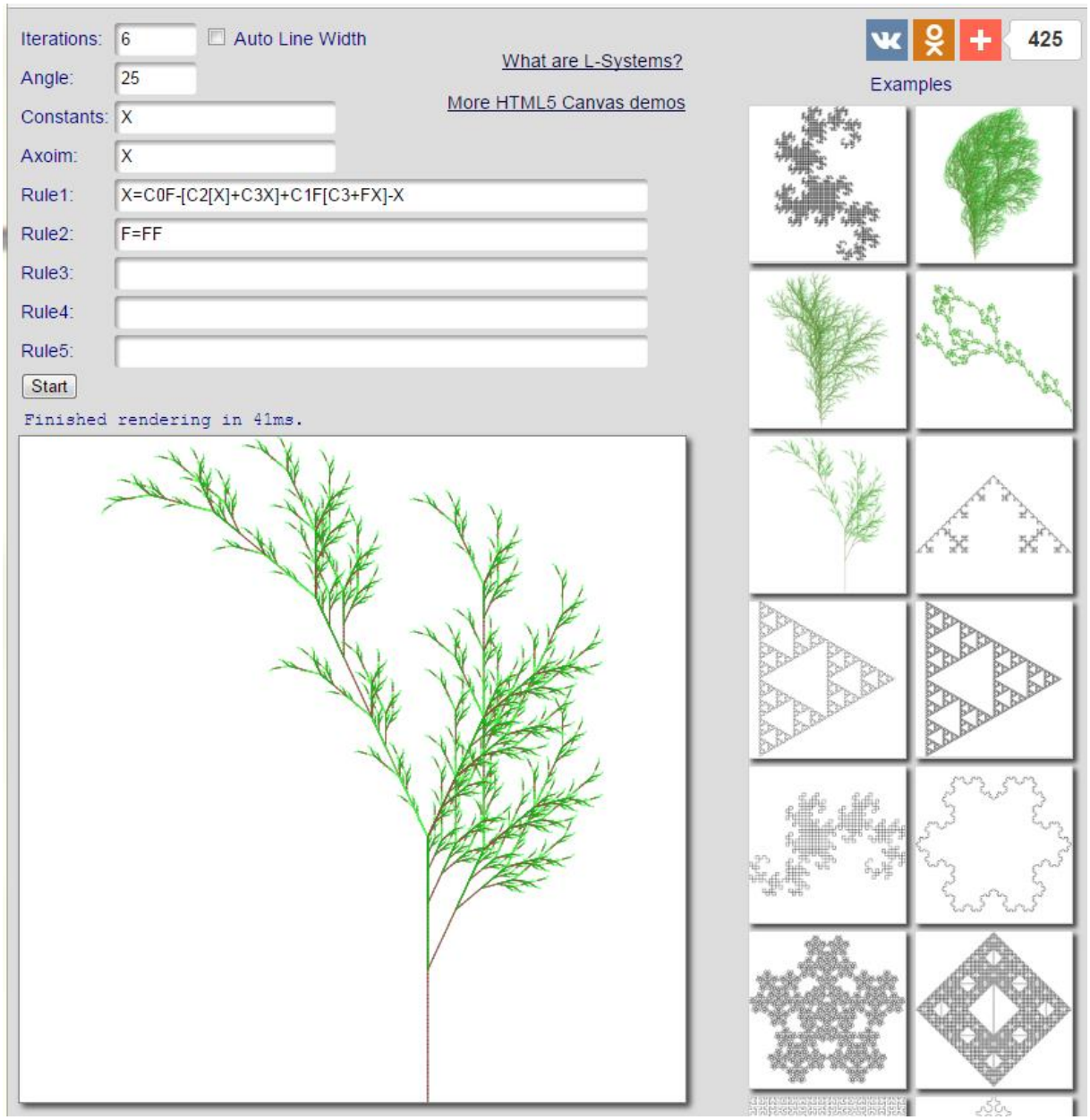


Рисунок 2.1 – Інтерфейс програми для побудови фракталів методом turtle-графіки

Тут висловлення «покласти в стек» і «вибрати зі стека» ставляться до LIFO-стеку (більш докладна граматика зажадала б розділити на «покласти в стек» і «повернути»). Коли інтерпретатор зустрічає «[», поточне положення черепахи й кут руху зберігаються в стеці, коли ж зустрічається «]», положення й кут відновлюються. Якщо кілька значень заносяться в стек, відновлюється останнє занесене значення. У літературі L-системи, що використовують такий підхід до розгалуження, називають «bracketed L-systems» (скобкові L-системи).

Застосовуючи ці графічні правила до отриманого раніше рядка, одержимо від кожного циклу реалізації правила рекурсії, показані на рисунку 2.2.

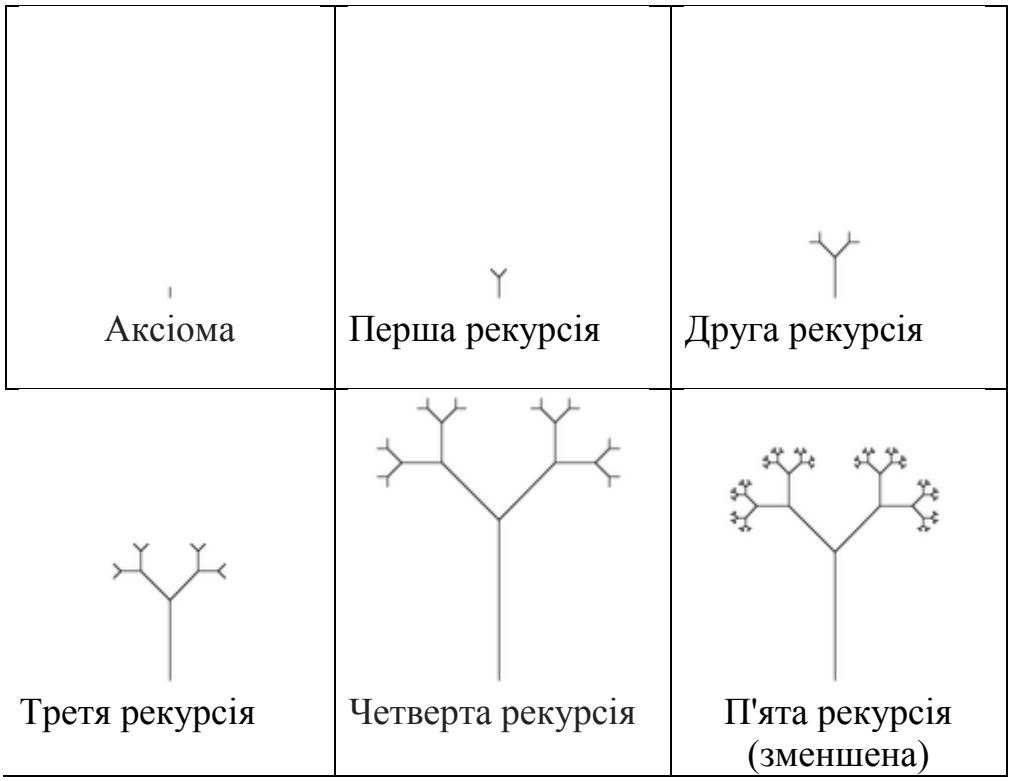


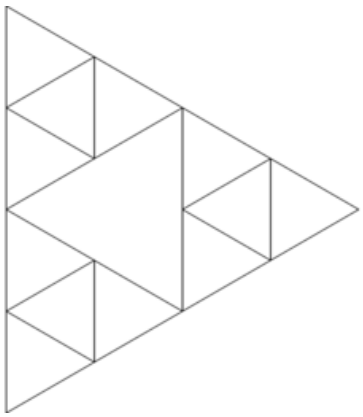
Рисунок 2.2 – Ілюстрація рекурсивного процесу побудови фрактала Дерево Піфагора

Розглянемо ще один приклад – побудувати за допомогою L-системи фрактал Трикутник Серпинского. Для цього прикладу застосовуються:

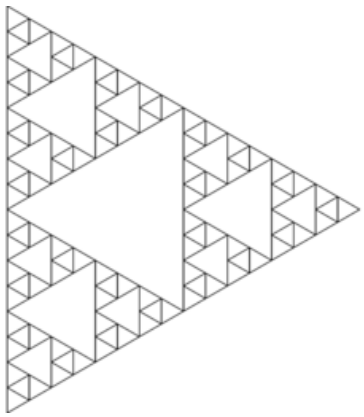
- змінні: F і G;
- константи: + і −;
- команда старт: F-G-G
- правило: (F → F-G+F+G-F), (G → GG)
- кут: 120°

Тут F і G означають «малюємо відрізок», «+» означає «повернути вправо на кут», « − » означає «повернути вліво на кут».

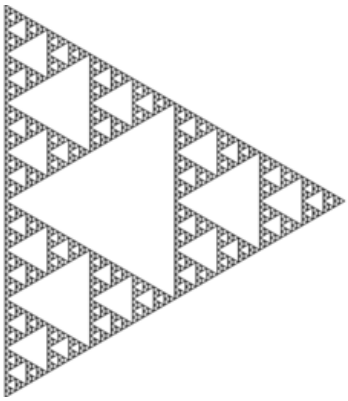
Рекурсивні перетворення Трикутника показано на рисунку 2.3 для кількості рекурсій 2, 4 і 6.



n = 2



n = 4



n = 6

Рисунок 2.3 – Ілюстрація рекурсивного процесу побудови Трикутника Серпинського

2.3 З**міст звіту по роботі**

Студент може самостійно вибрати програму та фрактал, що сподобався.

Звіт повинен містити:

1. Вихідну інформацію й опис команд.
2. Скриншот настроювань програми.
3. Зображення фракталів при зміні параметрів (ітерації).
4. Висновки по роботі.

При захисті звіту студент повинен продемонструвати роботу програми й відповісти на запитання, що стосуються теорії фракталів і методики їх моделювання.

ЛІТЕРАТУРА

1. Кроновер Р. М. Фракталы и хаос в динамических системах. Основы теории / Р. М. Кроновер.– Москва : Постмаркет, 2000. –с. 23 – 37
2. L-системы. [Электронный ресурс]. Метод доступа: http://stu.sernam.ru/book_fah.php?id=9

3 ДОСЛІДЖЕННЯ ДИВНИХ АТТРАКТОРІВ

3.1 Принципи утвору дивних атракторів

Математичним образом режиму функціонування динамічної системи служить *атрактор* – гранична траєкторія зображуючої точки у фазовому просторі, до якої прагнуть усі вихідні режими.

Якщо система прагне до ***стійкого стану рівноваги***, атрактор системи буде просто нерухливою точкою, якщо це стійкий ***періодичний*** рух – атрактором буде замкнена крива, називана граничним циклом.

Раніше вважалося, що атрактор є образом винятково стійкого режиму функціонування системи. Зараз же, у зв'язку з виявленням режиму детермінованого хаосу, стає зрозуміло, що граничною траєкторією системи, що перебуває в такому режимі, теж повинен бути атрактор. Однак такий атрактор буде мати дві істотні відмінності: його ***траєкторія неперіодична*** (вона не замикається) і ***режим функціонування нестійкий*** (малі відхилення від режиму наростають). Саме ці відмінності й привели до необхідності ввести новий термін – з легкої руки французького дослідника Ф. Такенса такі атрактори стали називати *дивними.*

Як установлено теоретиками, основним критерієм дивності атрактора є нестійкість траєкторії. Режими функціонування детермінованих нелінійних систем з дивними атракторами мають специфічні властивості, сукупність яких включається в поняття детермінованого хаосу.

Варто відразу відзначити, що слова «хаос» і «хаотична динаміка» зовсім не означають «повну плутанину». Динамічний хаос означає лише неможливість *акуратно передбачити віддалене майбутнє* системи, навіть якщо її «найближче майбутнє» описується досить точно. (Саме такими системами є, наприклад, земний клімат і навіть сама сонячна система.)

Стосовно до просторово-тимчасової системи хаос означає, що в однорідному середовищі спочатку спонтанно утворюються неперіодичні візерунки, які безладно мутують із часом. Конкретна фізична реалізація такого хаосу може бути всілякою. Наприклад, це можуть бути хвилі концентрації жертви в біологічній моделі «хижак-жертва» або конвекційні плини в широкій і дрібній кюветі, що підігрівається знизу.

Незважаючи на невизначеність «у довгостроковій перспективі», чимало інформації можна все-таки почерпнути з *ляпуновських експонент* — величин, що характеризують те, *як саме* в системі розбудовується хаос. Їхній аналіз показує, що хаоси бувають дуже різними, і навіть в одній і тій же системі при зміні параметрів можливі переходи від одного типу хаосу до іншого.

Усе це свідчить про те, що хаос теж можна розглядати як якусь фізичну систему, як *умовний газ «елементиків хаосу»*, у якому можуть бути свої

фізичні явища, наприклад, фазові переходи начебто плавлення або випари. Правда, для цієї картини треба прояснити суть цих «елементарних цеглинок хаосу» або хоча б навчитися підраховувати їхнє число (тобто число ступенів свободи хаосу) у кожному конкретному випадку.

3.2 Методика виконання роботи

Для виконання роботи можна застосувати існуючі в інтернеті програми:

1. Chaoscope 0.3.1
2. 2 Xaos 3.5
3. S-Attractor

Програма Chaoscope – це програма для генерування дивних атракторів. Інтерфейс програми представлено на рисунку 3.1.

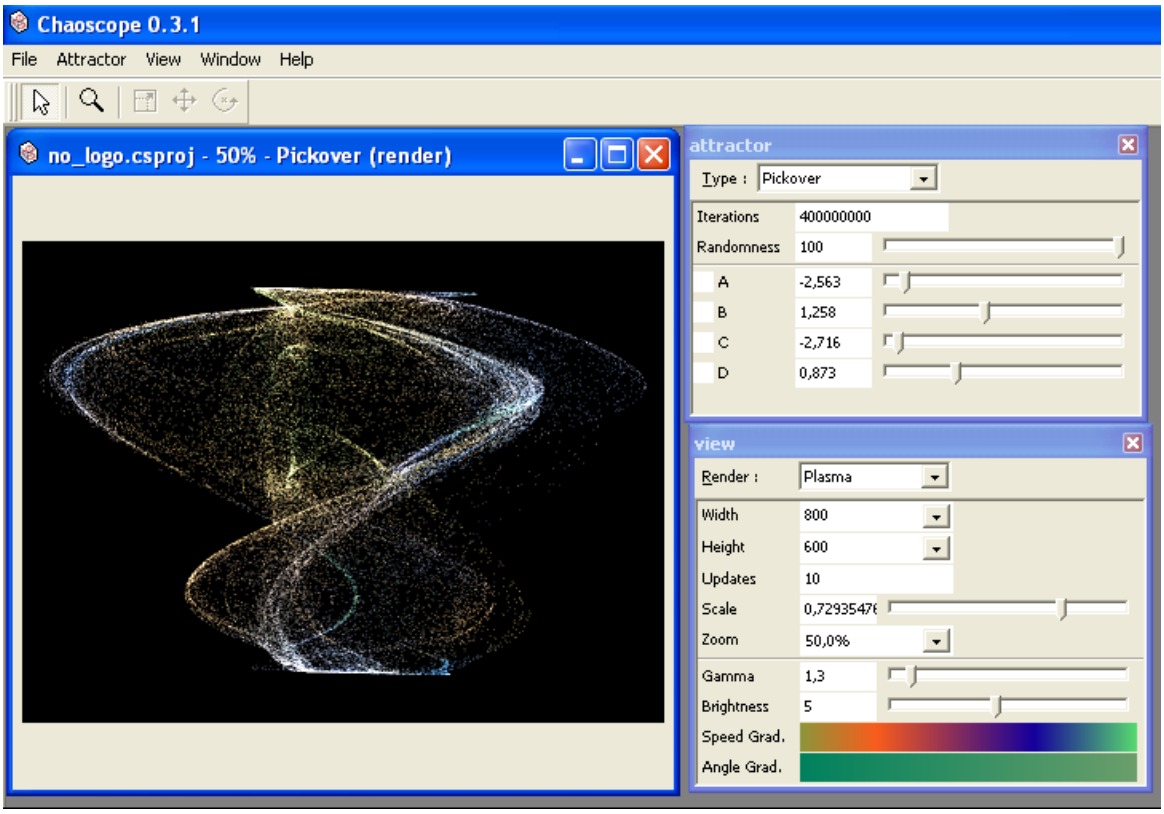


Рисунок 3.1 – Інтерфейс програми для дослідження дивних атракторів

Принцип генерації дивних атракторів полягає в тому, що спочатку створюється фрактальна множина, яка потім візуалізується в атрактор. При генерації фрактала можна піти двома шляхами: відкрити один із вбудованих бібліотечних проектів фрактальних зображень і доробити його задуманим способом або створити нову фрактальну множину (*File —> New*), а після цього запустити процес пошуку параметрів для можливого фрактального

зображення (*Attractor —> Search*), після чого вручну настроїти його параметри, підібрати колірну палітру й вибрати метод рендерінга.

Перелік настроювань досить скромний, але, проте, після серії експериментів програма дозволяє одержати цікаві результати. По закінченню можна вручну (тобто мишею) повернути фрактал бажаним чином і через командне меню розташувати його в центрі зображення.

Готове фрактальне зображення шляхом візуалізації (*Attractor —> Render*) перетворюється в атрактор, який можна зберегти у вигляді проекту у власному форматі програми або експортувати в bmp-файл. Процес рендерінга найчастіше виявляється досить тривалим.

Програма Xaos 3.5 – це багатоплатформений генератор фракталів, що дозволяє генерувати фрактальні зображення для базових типів фрактальних множин. Як і в інших розв'язках, у ньому можна одержати цікаві варіанти зображень, однак можливості настроювання в генераторі мінімальні. Зате освоїти програму нескладно, тому вона цілком підходить для вивчення фрактальної графіки в навчальних закладах.

Сайт програми: *http://wmi.math.u-szeged.hu/xaos/doku.php*

Інтерфейс програми показано на рисунку 3.2.

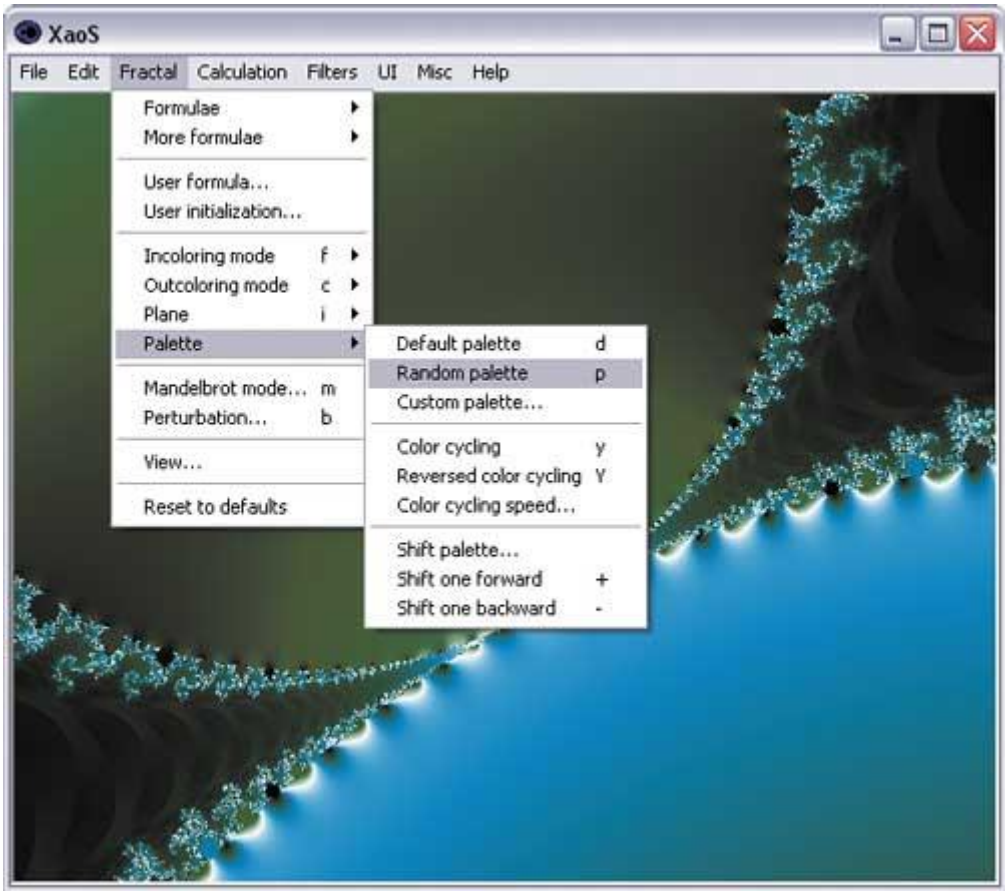


Рисунок 3.2 –. Вибір випадкової колірної палітри в Xaos

Варіантів генерації фрактальних зображень в Xaos два: їх можна створювати на базі вбудованих прикладів (*File —> Load*) або базових фрак-

тальних множин (*Fractal —> Formula*). Крім того, допускається введення користувацьких формул у простенькому редакторі. Отримані зображення масштабуються, переміщуються й обертаються мишею, а колірні переходи в них настроюються через серію вбудованих палітр (останнє реалізоване незручно, але рано або пізно дозволяє добитися ефектного результату). Крім цього передбачені можливості зміни режиму позиціонування фрактала, регулювання числа ітерацій і накладення ряду фільтрів.

Згенеровані фрактальні зображення зберігаються у вигляді проектів у власному форматі або експортуються у формат PNG.

Програма S-Attractor 1.00, інтерфейс якої показано на рисунку 3.3, демонструє дивний атрактор, створюваний ітераційним процесом відповідно до виражень:

$$x[i+1] = Sin(ax[i]) - Cos(by[i])$$
$$y[i+1] = Sin(cx[i]) - Cos(dy[i])$$

При зміні параметрів ***a, b, c, d*** форма атрактора змінюється.

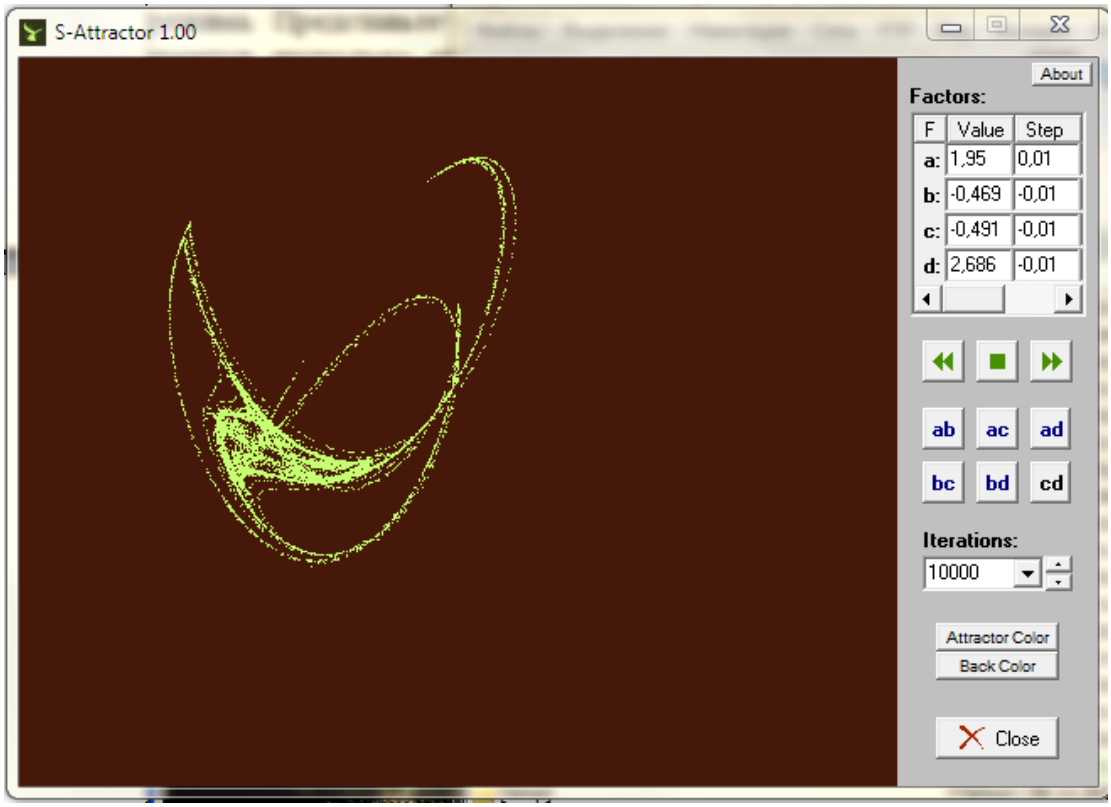


Рисунок 3.3 – Інтерфейс програми S-Attractor 1.00

Таблиця параметрів дозволяє вибирати значення ***a, b, c, d***, кроки й діапазони їх зміни. Також параметри можна змінювати на ходу. Слід помітити, що застосування зміненого параметра може відбуватися при натисканні ENTER, а якщо ні, то присвоєння нового значення не відбу-

ається й після перемикання фокуса на інший елемент знову відображається старе значення.

Керування процесом генерації атрактора може здійснюватися кнопками й інструментом «Рука».

Кнопки виконують функції:

◄ – зменшення параметра.

■ – стоп.

► – збільшення параметра.

Після досягнення параметром максимального або мінімального значення (установлюється в таблиці параметрів), при автоматичній зміні, крок міняє знак.

Інструмент "Рука" міняє параметри при переміщенні по зображенню з натиснутою клавішею миші.

Якщо натиснути, наприклад, кнопку ***ab***, то при натиснутій лівій кнопці миші переміщення "Руки" змінює параметри ***a*** і ***b***, а при натиснутій правій кнопці миші – параметри ***b*** і ***c***. Якщо натиснути кнопку ***ac***, то рух "Руки" з лівою кнопкою миші змінює значення параметрів ***a*** і ***c***, а правою кнопкою – параметри ***b*** і ***d***.

Кнопкою Iteration можна задати число ітерацій.

Вибір кольору атрактора й кольору фону здійснюється кнопками Attractor Color і Back Color.

Програма не прописується до реєстру й не торкається інших каталогів, тому для видалення досить вилучити каталог S-Attractor.

3.3 Зміст звіту по роботі

Студент може самостійно вибрати програму й побудувати декілька (3-4) атракторів.

Звіт повинен містити:

1. Вихідну інформацію й опис програми.
2. Скриншот настроювань програми.
3. Зображення атракторів при зміні настроювань параметрів (ітерації).
4. Висновки по роботі.

При захисті звіту студент повинен продемонструвати роботу програми й відповісти на запитання, що стосуються теорії динамічного хаосу й методики моделювання хаотичних траєкторій.

ЛІТЕРАТУРА

1 Мун Ф. Хаотические колебания: Вводный курс для научных работников и инженеров / Ф. Мун - Пер. с англ. – М.: Мир, 1990. - 312 с.

4 СТВОРЕННЯ, АДАПТАЦІЯ Й НАВЧАННЯ ЛІНІЙНОЇ НЕЙРОННОЇ МЕРЕЖІ В КОМАНДНОМУ ВІКНІ MATLAB

Ціль роботи: освоїти методику й придбати навички створення, адаптації й навчання лінійної нейронної мережі із застосування пакета прикладних програм (ППП) Neural Network Toolbox системи програмування MATLAB.

4.1 Методика створення і адаптації мережі у програмному середовищі *NEURAL NETWORK TOOLBOX*

Формування й ініціалізація мережі. Нейронна мережа створюється оператором **net**, а лінійний шар – оператором **newlin**. Наприклад, лінійна мережа із двоелементним вектором входу, значення якого перебувають у діапазоні **[-1 1]**, одним шаром, без лінії затримки на вході (тип 0) і параметром швидкості настроювання 0.2 формується наступною командою:

```
net=newlin([-1 1;-1 1],1,0,0.2);
```

Така мережа функціонує відповідно до лінійної залежності виду:

$$a = purelin(\mathbf{W}\mathbf{p} + b),$$

де a – вихід мережі, $purelin$ – лінійна функція активації нейрона, $\mathbf{W}$ – вагова матриця входів, $\mathbf{P}$ – вектор входів, b – зсув нейрона.

Модель цієї нейронної мережі показано на рисунку 4.1.

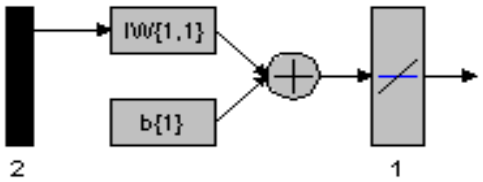


Рисунок 4.1 – Модель лінійної нейронної мережі із двоелементним вектором входу

Нехай, наприклад, вагова матриця повинна бути задана вектором $\mathbf{W} = [\mathbf{1}\ \mathbf{2}]$, а зсув $b = 0$. Тоді ці параметри мережі в описі її структури будуть представлені в такий спосіб:

```
net.IW{1,1}=[1,2];
net.b{1}=0;
```

При формуванні мережі оператором newlin ініціалізація мережі проводиться за замовчуванням, тому застосовувати метод net=init(net) не потрібно.

Адаптація (настроювання) нейронної мережі. Процедура адаптації здійснюється при введенні навчальної вибірки послідовним або груповим способом. Для адаптації мережі необхідно задати вектори входу **P** й мети **T**, а також установити початкові значення ваги **IW** і зсуву b.

Процес адаптації розглянемо на прикладі формування лінійної залежності виду:

$$t = 2p_1 + p_2.$$

При ***послідовному способі*** завдання навчальної послідовності вектори задаються у вигляді масивів гнізд формату cell**:**

```
>> net=newlin([-1 1;-1 1],1,0,0);
>> P={[-1;1] [-1/3;1/4] [1/2;0] [1/6;2/3]};
>> T={-1 -5/12 1 1};
>> P1=[P{:}], T1=[T{:}]
P1 =
  -1.0000   -0.3333    0.5000    0.1667
   1.0000    0.2500         0    0.6667
T1 =
  -1.0000   -0.4167    1.0000    1.0000
```

Спочатку задамо мережу з нульовими значеннями початкових значень ваги і зсувів:

```
>> net.IW{1}=[0 0];
>> net.b{1}=0;
```

Щоб забезпечити можливість вибирати довільні функції для настроювання ваги і зсувів, скористаємося М-функцією adapt. Для лінійних мереж, створюваних за допомогою методу newlin, за замовчуванням установлюється функція adaptwb настроювання режиму й функція learnwh настроювання параметрів мережі по алгоритму WH (правило Уідроу-Хоффа).

Learnwh обчислює величину зміни ваги входу даного нейрона **dw** від внеску нейрона *pn',* помилки e і швидкості настроювання *lr,* згідно Widrow-Hoff правилу:

$$\mathbf{dw} = lr * e * pn'.$$

Функції настроювання ваги і зсувів задаються у форматі net.inputweighst{i,j}.learnfcn і net.biases{i,j}.learnfcn, відповідно.

Виконаємо перший цикл адаптації з нульовим параметром швидкості настроювання й визначимо значення виходів (a) і помилки (e):

```
>> [net1,a,e]=adapt(net,P,T);
>> net1.IW{1,1},a,e
ans =
    0     0
a =
   [0]    [0]    [0]    [0]
e =
   [-1]    [-0.4167]    [1]    [1]
```

Як видно з наведеного фрагмента, ваги не модифікуються, виходи мережі залишаються рівними нулю й адаптація не відбувається, тому що параметр швидкості настроювання не заданий.

Задамо початкові значення ваг входів і зсуву, а також установимо функції й параметр швидкості настроювання. При цьому вважаємо, що в шуканій функції виходу не повинно бути постійної складової, а тому для параметра швидкості настроювання зсуву встановлюємо нульове значення:

```
>> net.IW{1}=[0 0];
>> net.b{1}=0;
>> net.inputweights{1,1}.learnparam.lr=0.2;
>> net.biases{1,1}.learnparam.lr=0;
>> [net1,a,e]=adapt(net,P,T);
>> net1.IW{1,1},a,e
ans =
   0.3454   -0.0694
a =
   [0]    [-0.1167]    [0.1100]    [-0.0918]
e =
   [-1]    [-0.3000]    [0.8900]    [1.0918]
```

Звідси видно, що процес адаптації почався, але для досягнення необхідної точності, наприклад 0,015, одного циклу недостатньо.

Виконаємо послідовну адаптацію мережі протягом 30 циклів. Для обчислення середньоквадратичної помилки використовуємо оператор mse**,** а для конвертування масиву гнізд cell у масив чисел подвоєної точності double застосуємо функцію cell2mat:

```
>> net=newlin([-1 1;-1 1],1, 0, 0);
>> net.IW{1}=[0 0];
>> net.b{1}=0;
>> net.inputweights{1,1}.learnparam.lr=0.2;
>> net.biases{1,1}.learnparam.lr=0;
>> P={[-1;1] [-1/3; 1/4] [1/2; 0] [1/6; 2/3]};
>> T={-1 -5/12 1 1};
```

```
>> for i=1:30,
[net, a{i}, e{i}]=adapt(net, P, T);
W(i,:)=net.IW{1,1};
end
>> mse(cell2mat(e{30}))
ans =
   0.0017
>> W(30,:)
ans =
   1.9199    0.9250
>> cell2mat(e{30})
ans =
  -0.0056   -0.0081    0.0434    0.0699
```

Після 30 циклів адаптації мережі по чотирьом навчальним векторам входу й мети відбулася зміна значень ваг входів і помилок.

Побудуємо графіки цих змін у функції числа циклів (ітерацій) процесу адаптації. Розташуємо їх в одній фігурі друг над другом, застосовуючи для цього функцію subplot:

```
>> subplot(2,1,1)
>> for i=1:30, E(i)=mse(e{i}); end
>> semilogy(1:30, E, '+k')
>> xlabel('Цикли'), ylabel('Помилка'), grid
>> subplot(2,1,2)
>> plot(0:30,[[0 0]; W], 'k');
>> xlabel('Цикли'), ylabel('Ваги входів w(i)'),grid
```

Графіки функцій представлено на рисунку 4.2.

Не важко переконатися, що процес адаптації зажадав 12 кроків – помилка навчання досягла на цьому етапі значення 1,489e-3. Це свідчить про те, що навчальна вибірка була представницькою й мережа відповідає розв'язуваній задачі.

При ***груповому способі*** завдання навчальної послідовності вектори входу й мети задаються масивами у форматі double.

Основний цикл адаптації мережі зорганізується в такий спосіб:

```
net=newlin([-1 1;-1 1],1,0,0);
P=[-1 -1/3 1/2 1/6; 1 1/4 0 2/3];
T=[-1 -5/12 1 1];
net=newlin([-1 1;-1 1],1,0,0.2);
net.IW{1}=[0 0];
net.b{1}=0;
net.inputweights{1,1}.learnparam.lr=0.2;
```

```
P=[-1 -1/3 1/2 1/6; 1 1/4 0 2/3];
T=[-1 -5/12 1 1];
EE=10; i=1;
while EE>0.0017176
   [net,a{i}, e{i}, pf]=adapt(net,P,T);
   W(i,:)=net.IW{1,1};
   EE=mse(e{i});
   ee(i)=EE;
   i=i+1;
end
```

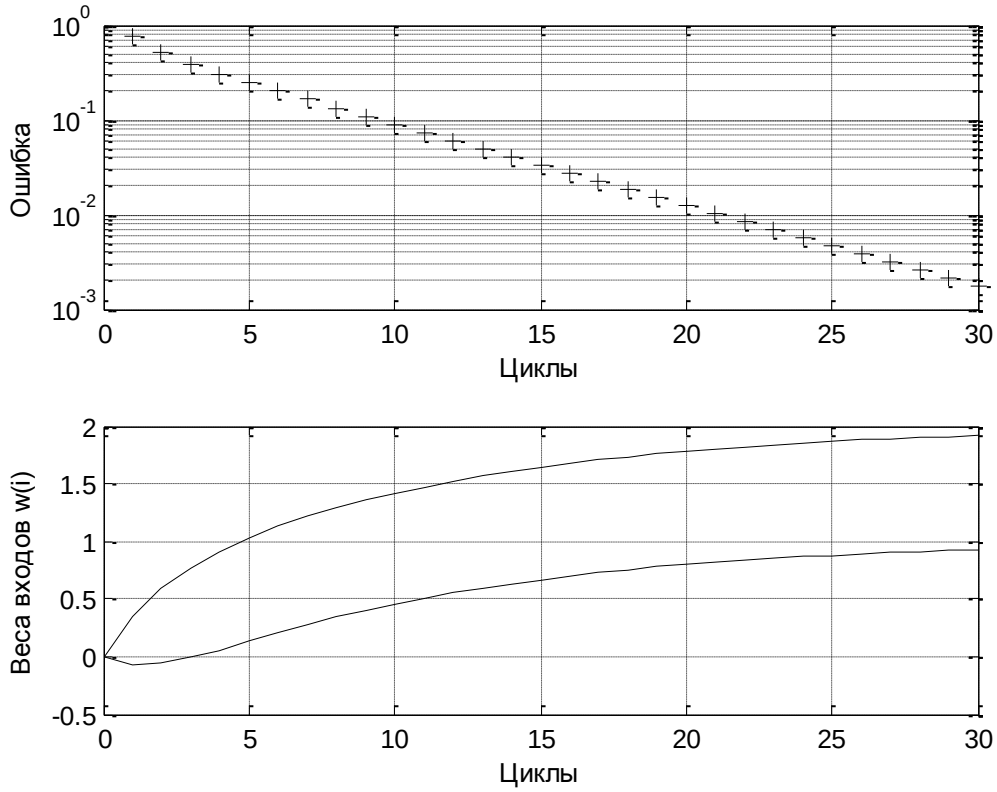


Рисунок 4.2 – Графіки зміни значень помилки й ваг входів у процесі адаптації

Результатом адаптації є наступні значення коефіцієнтів лінійної залежності (ваг входів), значень виходів мережі й середньоквадратичної погрішності адаптації:

```
W(63,:)
ans =
   1.9114    0.8477
cell2mat(a(63))
ans =
  -1.0030   -0.3624    1.0172    0.9426
EE=mse(e{63})
```

```
EE =
   0.0016
```

Представимо процедуру адаптації нейронної мережі в динаміці на графіках. Об'єднаємо в одну фігуру графіки функцій виходів, ваг входів і помилки:

```
subplot(3,1,1)
plot(0:63,[zeros(1,4); cell2mat(a')],'k')
xlabel(''),ylabel('Виходи a(i)'),grid
subplot(3,1,2)
plot(0:63,[[0 0];W],'k')
xlabel(''),ylabel('Ваги входів w(i)'),grid
subplot (3,1,3)
semilogy(1:63, ee,'+k')
xlabel('Цикли'),ylabel('Помилка'),grid
```

Графіки функцій показано на рисунку 4.3.

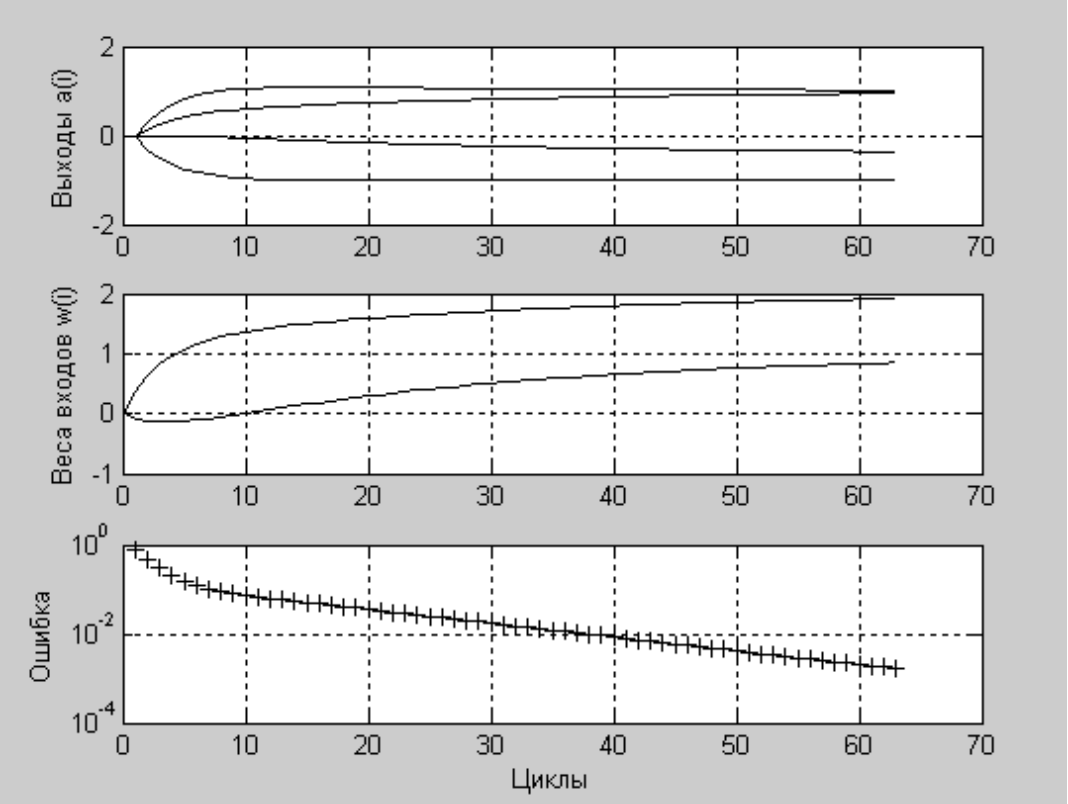


Рисунок 4.3 – Графіки вихідних сигналів мережі, ваг входів і помилки при адаптації із груповим завданням навчальної вибірки

Як випливає із аналізу графіків, для досягнення необхідної точності адаптації тут також треба було 12 кроків.

Слід урахувати, що в лінійних *динамічних* мережах природнім є тільки послідовний спосіб, тому що ці мережі мають лінії затримки сигналів.

4.2 Методика навчання мережі.

Для навчання й настроювання параметрів мережі використовуються функції **trainwb** і **learnwh**, відповідно.

Після формування мережі, завдання навчальної вибірки (послідовним способом), уведення параметрів швидкості настроювання кількості циклів навчання (епох) одержимо значення ваг входів і графік залежності величини помилки від числа циклів навчання, який представлено на рисунку 4.4.

Рисунок 4.4 – Графік залежності помилки навчання від числа циклів

Далі складаємо програму навчання.

```
>> net=newlin([-1 1;-1 1],1, 0, 0);
net.IW{1}=[0 0];
net.b{1}=0;
P={[-1;1] [-1/3; 1/4] [1/2; 0] [1/6; 2/3]};
T={-1 -5/12 1 1};
net.inputweights{1,1}.learnparam.lr=0.2;
net.biases{1,1}.learnparam.lr=0;
net.trainparam.epochs=30;
net1=train(net,P,T);
W=net1.IW{1}
TRAINB, Epoch 0/30, MSE 0.793403/0.
TRAINB, Epoch 25/30, MSE 0.00373997/0.
```

TRAINB, Epoch 30/30, MSE 0.00138167/0.
TRAINB, Maximum epoch reached.
W =
1.9214 0.9260

Аналіз графіку на рис. 4.4 свідчить про те, що при 30 циклах навчання помилка навчання практично відсутня.

4.3 Варіанти індивідуальних завдань та зміст звіту по роботі

Для виконання роботи студенти одержують індивідуальні завдання, варіанти яких наведено в таблиці 4.1.

Таблиця 4.1 – Варіанти індивідуальних завдань

Варіант	**Вхід**	Значення вхідних сигналів навчальної послідовності $P1/P2$ (один по одному)						Значення функції мети T					
		1	2	3	4	5	6	1	2	3	4	5	6
1	1	1/2	-1/2	1	-1/4	1/4	-1/2	0,9	-0,5	1,5	-1	1	-1
	2	0	-1/4	1/4	1/2	-3/4	1/6						
2	1	-1/2	-1	3/4	1/2	-1	-3/4	-0,5	-1,8	1	0,8	1	0
	2	1/5	3/4	-2/5	-1/4	-1	-1/4						
3	1	1/4	1	1/2	-1	-1/2	0	0,3	0,8	0	-0,7	0,2	0,5
	2	-1/4	-1/2	3/4	1/4	0	1						
4	1	1/2	-1/2	-1	1	3/4	-3/4	1,5	1,5	0	-3	-2	0,8
	2	1	1/2	-1/2	-1	-2/3	0						
5	1	-3/4	1/2	-1	0	1	3/4	1,5	0,7	1,5	-1	-2	-2,2
	2	0	1/4	-1/2	-1	0	-3/4						
6	1	3/4	-1	0	1/2	-1/4	-3/4	2	-1	1,5	1,2	-1	-2,3
	2	-3/4	0	-1	-1/2	1/2	1						
7	1	-3/4	1	1/4	1/2	-1	0	-1,6	1,8	-0,1	1	-1,5	-0,2
	2	1	-1/2	1	-1	0	1/2						
8	1	1	1/2	-2/3	3/4	-1	3/4	-0,3	2,4	0	0,2	-1,3	2,3
	2	1	-1	-1/2	1/2	0	-3/4						
9	1	3/4	-1	1/4	-1/4	1	-1/5	1,5	-2	-0,4	1	1	-1
	2	-3/4	1	1/2	-1	-1/4	3/4						
10	1	1/2	-1/2	-1	1	1/2	0	-0,3	0,7	-2,8	3,5	2	-1
	2	3/4	-1	1/2	-1	-2/3	2/3						

У процесі створення мережі необхідно виконати наступне.

1 Вибрати структуру мережі й зробити її ініціалізацію (задати початкові значення ваг і зсув).

2 Вибрати критерій якості й функцію настроювання мережі.

3 Задати кількість циклів адаптації, зробити адаптацію мережі.

4 Побудувати графіки функцій виходу мережі, ваг коефіцієнтів і критерію якості адаптації.

Звіт по роботі повинен містити завдання, структуру мережі, програму й результати адаптації мережі.

При захисті звіту студент повинен продемонструвати роботу програми й відповісти на запитання, що стосуються теорії динамічного хаосу й методики моделювання хаотичних траєкторій.

ЛІТЕРАТУРА

1. Підручник по нейронних мережах. [Електронний ресурс] Режим доступу: http://neuralnet.info

5 РОЗРОБКА РАДІАЛЬНОЇ БАЗИСНОЇ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ АПРОКСИМАЦІЇ ФУНКЦІЙ

5.1 Методика формування радіальної базисної мережі

Принцип функціонування мережі. Мережі радіальних базисних функцій (RBF) відрізняються від лінійних нейронних мереж тим, що містять шар радіально-симетричних схованих нейронів (шаблоновий шар). Для забезпечення радіально-симетричних властивостей необхідно:

- наявність центру, представленого вектором;
- наявність способу виміру відстані від центру до вхідного вектора;
- наявність спеціальної функції, що відображає функцію відстані.

У радіальних базисних мережах усі ці умови забезпечуються застосуванням радіального базисного нейрона з функцією активації виду:

$$a = radbas(n) = e^{-n^2},$$

де n – вхід функції активації, обумовлений як модуль різниці вектора ваг і вектора входу, помножений на зсув: $n = \|\mathbf{p} - \mathbf{w}\| b$.

Практично це означає, що вихідний сигнал шаблонового нейрона – це функція відстані між вхідним вектором і збереженим центром. Чим ближче вхідний вектор до центру, тим більше вихідний сигнал нейрона (рис. 5.1). При цьому радіальний базисний нейрон діє як індикатор, який формує значення 1, коли вхід $\mathbf{p}$ ідентичний вектору ваги $\mathbf{IW}$. Зсув b дозволяє коректувати чутливість радіального базисного нейрона.

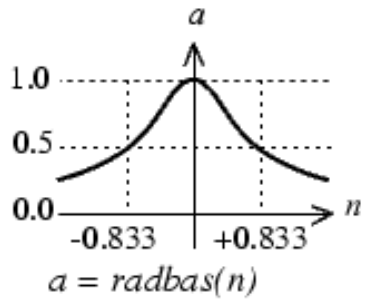
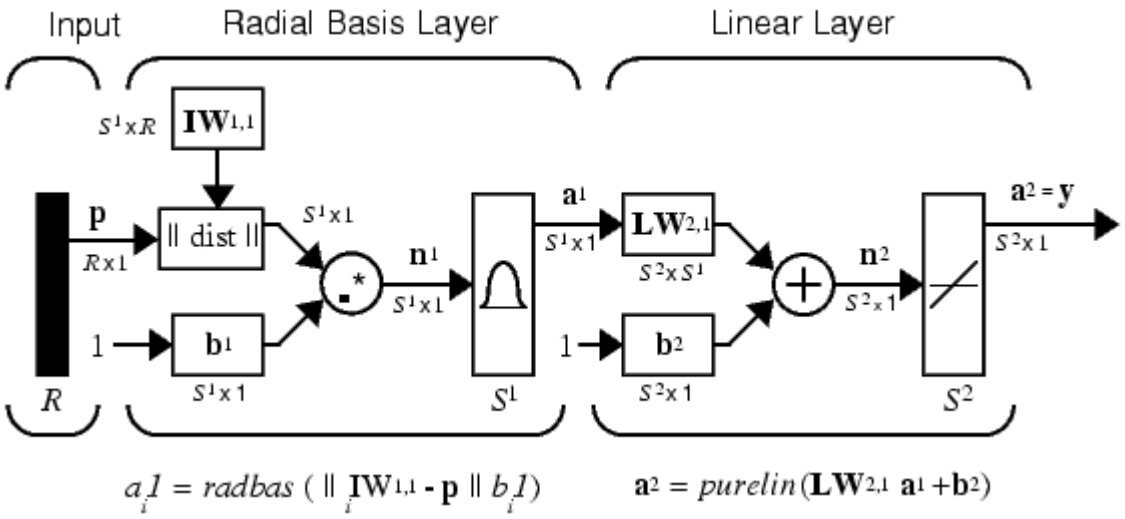
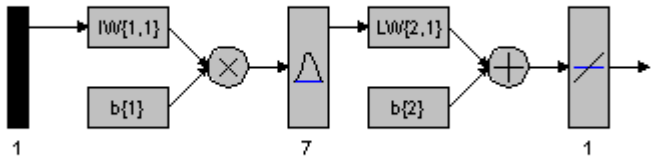


Рисунок 5.1 – Графік функції активації

Структуру мережі показано на рисунку 5.2. Мережа містить радіальний базисний шар і лінійний шар.



а



б

Рисунок 5.2 – Повна (а) і спрощена (б) структури моделі радіальної базисної мережі

Схований радіальний базисний шар має S^1 нейронів, а вихідний шар – S^2 нейронів. Радіальний базисний шар містить спеціальний блок dist, на вхід якого подаються вхідний вектор **p** і матриця ваг **IW**. Виходом блоку є вектор, що полягає з S^1 елементів, які визначаються відстанями між i-м вектором входу й i-м вектор-рядком **IW** матриці ваг. Вихід блоку dist множиться поелементно на вектор зсуву **b** й формує вхід функції активації. У якості функції активації використовується зазвичай функція Гауса.

Створення мережі. Для створення радіальної базисної мережі призначені М-функції **newrbe** і **newrb**. Перша дозволяє побудувати мережу з нульовою помилкою, друга дозволяє управляти кількістю нейронів схованого шару. Недолік функції newrb полягає в тому, що вона формує мережу із числом нейронів у схованому шарі, рівним числу елементів навчальної множини. Якщо таких елементів багато, що характерно для реальних випадків, то прийнятний розв'язок досягається із труднощами.

Створимо мережу з нульовою помилкою:

```
net=newrbe(P,T,SPREAD);
```

Вхідними аргументами функції newrbe є масиви вхідних векторів **P** і векторів мети **T**, а також параметр впливу SPREAD**.** Функція встановлює ваги першого шару рівними **P**. Зсуви встановлюються рівними 0,8326 / SPREAD. Це означає, що рівень перекриття радіальних базисних функцій рівний 0,5 і всі входи в діапазоні ±SPREAD вважаються значимими.

Ваги другого шару можуть бути знайдені шляхом моделювання виходів першого шару (A {1}) і наступного розв'язку системи лінійних алгебраїчних рівнянь:

[W{2,1} b{2}] * [A{1}; ones]=T

Враховуючи те, що входи другого шару A {1} і мети T відомі, а функція активації лінійна, то для обчислення ваг і зсувів другого шару досить розв'язати систему рівнянь:

Wb = T/[P; ones(1, size(P,2))]

5.2 Меодика навчання радіальної базисної мережі

Розглянемо процедуру навчання радіальної базисної мережі для довільної функції. Значення входів змінюються в діапазоні від -1 до 1. Навчальна вибірка складена для 21 точки функції через 0,1. Графік функції представлено на рисунку 5.3.

```
P = -1:.1:1;
T = [-.9602 -.5770 -.0729  .3771  .6405  .6600  .4609
  .1336 -.2013 -.4344 -.5000 -.3930 -.1647  .0988
  .3072  .3960  .3449  .1816 -.0312 -.2189 -.3201];
plot(P,T,'+');
title('Training Vectors');
xlabel('Input Vector P');
ylabel('Target Vector T');grid
```

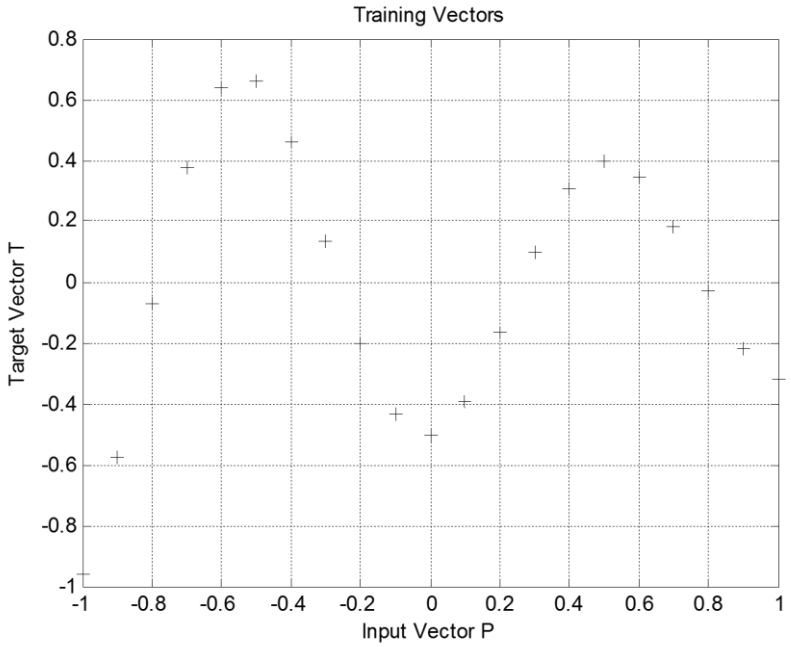


Рисунок 5.3 – Графік навчальної функції (вектор **T**)

Збережемо поточний графік і сформуємо мережу:

```
hold on
net = newrbe(P,T);
net.layers{1}.size
Warning: Rank deficient, rank = 13, tol = 2.2386e-014.
> In newrbe>designrbe at 120
  In newrbe at 103
ans =
   21        %  Кількість нейронів схованого шару рівна 21
```

Зробимо моделювання мережі:

```
V = sim(net,P);
plot (P,V,'ob','Markersize',5,'Linewidth',2)
p = [-0.75 -0.25 0.25 0.75];
v = sim(net,p);
plot(p,v,'+k','Markersize',10,'Linewidth',2)
```

Результати моделювання наведено на рисунку 5.4.

Як видно з рисунка 5.4, вектор **T** для нових значень входу визначений з високою точністю.

Створимо мережу з меншим числом нейронів, застосувавши функцію newrb.

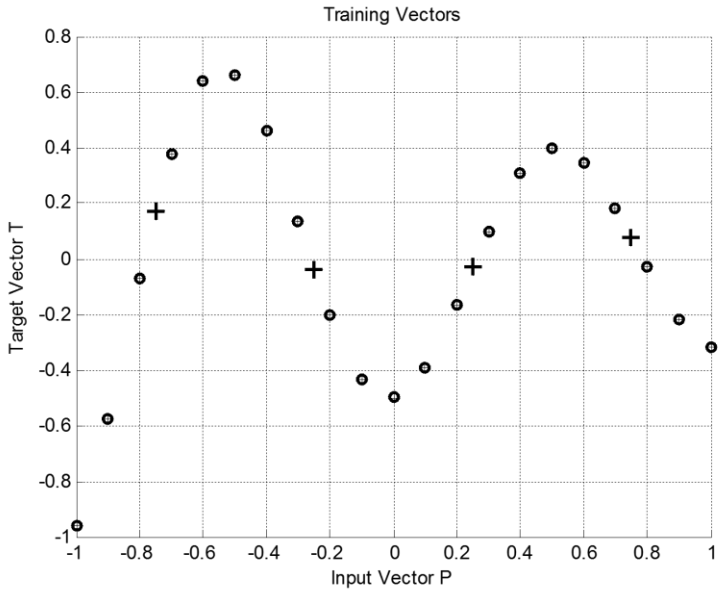


Рисунок 5.4 – Результати моделювання мережі при подачі на вхід чотирьох нових значень

Нову мережу навчимо колишньою послідовністю цілей:

```
P = -1:.1:1;
T = [-.9602 -.5770 -.0729  .3771  .6405  .6600  .4609
    .1336 -.2013 -.4344 -.5000 -.3930 -.1647  .0988
    .3072  .3960  .3449  .1816 -.0312 -.2189 -.3201];
plot(P,T,'*r','Markersize',4,'Linewidth',2)
hold on
GOAL=0.01; % Припустиме значення помилки
net=newrb(P,T,GOAL);
net.layers{1}.size
NEWRB, neurons = 0, SSE = 3.69051
ans =
    6    % Кількість нейронів схованого шару рівна 6
v=sim(net,p);
p=[-0.75 -0.25 0.25 0.75];
v=sim(net,p);
plot(p,v,'+k','Markersize',10,'Linewidth',2)
```

Як видно з М-файлу, число нейронів першого шару рівне 6. При цьому забезпечується середньоквадратична помилка менш 0,01.

Результати моделювання наведено на рисунку 5.5.

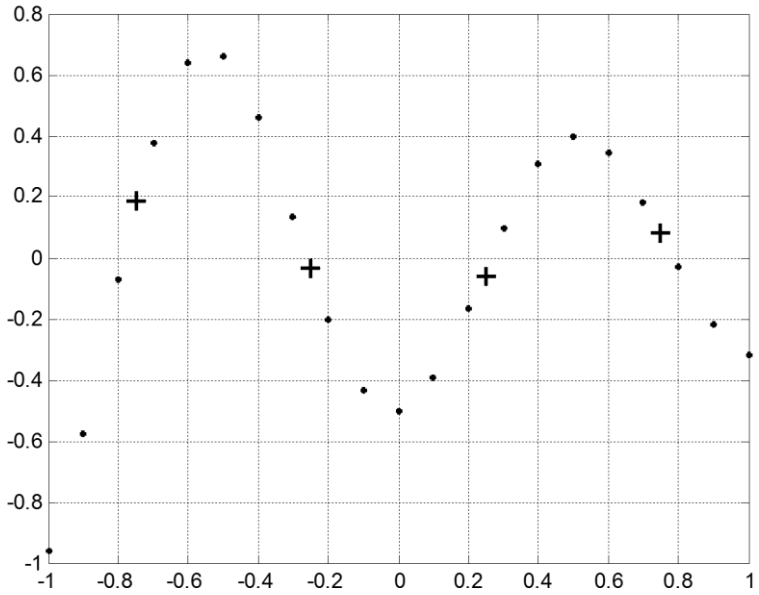


Рисунок 5.5 – Результати моделювання мережі із шістьома нейронами

Апроксимація функцій. При застосуванні радіальних базисних мереж для апроксимації функцій використовується ідея розкладання довільної функції на ряд радіальних базисних функцій у такий спосіб:

$$f(x) = \sum_{i=1}^{N} a_i \varphi_i(x),$$

де $\varphi_i(x)$ – радіальна базисна функція.

Розглянемо процес апроксимації трьох радіальних базисних функцій, заданих на інтервалі [-3 3]:

```
p = -3:.1:3;
a1 = radbas(p);
a2 = radbas(p-1.5);
a3 = radbas(p+2)*0.5;
a4 = a1 + a2 + a3;
plot(p,a1,'b-',p,a2,'b--',p,a3,'b--',p,a4,'m-')
title('Weighted Sum of Radial Basis Transfer Functions');
xlabel('Input p');
ylabel('Output a');
```

Результати апроксимації представлено на рисунку 5.6.

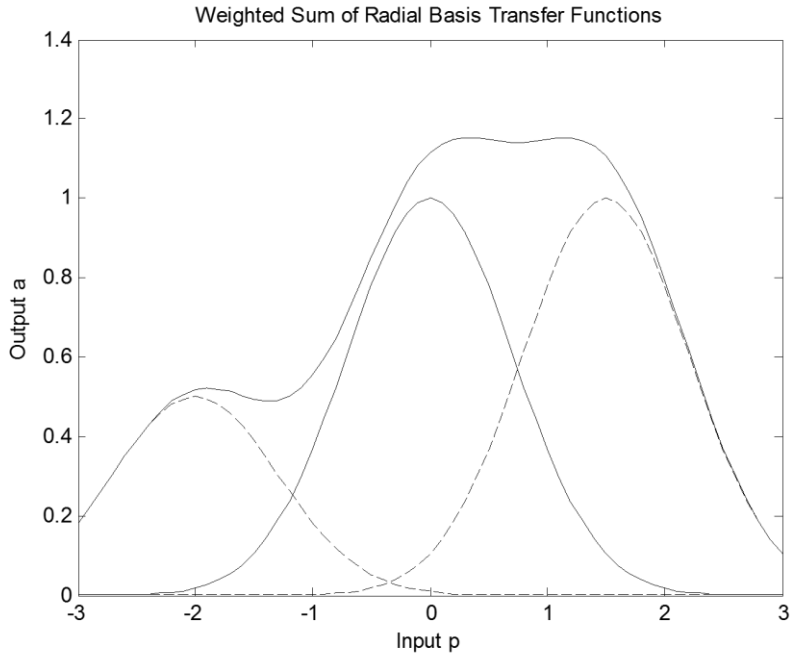


Рисунок 5.6 – Графіки базисних функцій і результат їх апроксимації

З рисунка 5.6 видно, що при розкладанні довільної функції радіальними базисними функціями забезпечується необхідна гладкість.

Ступінь гладкості й точність апроксимації пов'язані з параметром впливу SPREAD. Як відомо, цей параметр визначає діапазон значимих входів (див. вище).

Розглянемо, у чому проявляється цей вплив.

Використовуючи попередній приклад навчальної вибірки, будемо задавати значення SPREAD, рівними 0,01; 1 і 20.

Уведемо M-файл зі значенням SPREAD = 0,01.

```
P = -1:.1:1;
T = [-.9602 -.5770 -.0729 .3771 .6405 .6600 .4609
   .1336 -.2013 -.4344 -.5000 -.3930 -.1647 .0988
   .3072 .3960 .3449 .1816 -.0312 -.2189 -.3201];
  GOAL=0.01;
  SPREAD=0.01;
  net=newrb(P,T,GOAL,SPREAD);
  net.layers{1}.size
NEWRB, neurons = 0, SSE = 2.758
ans =
    19      % Кількість нейронів схованого шару рівно 19
X=-1:.01:1;
Y=sim(net,X);
plot(X,Y);
title('Training Vectors');
xlabel('Input Vector P');
ylabel('Target Vector T');
```

Після цього одержуємо результат, який показано на рисунку 5.7.

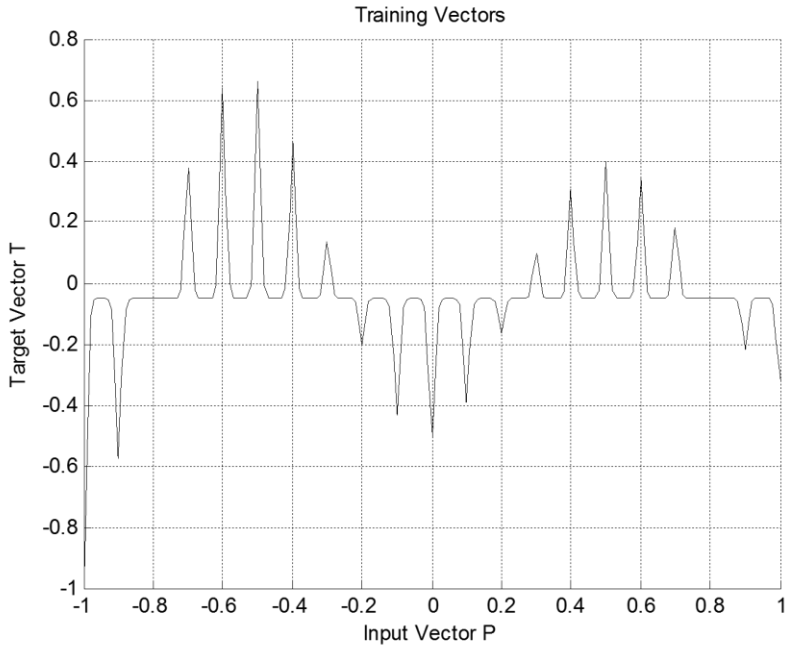


Рисунок 5.7 – Результат апроксимації при SPREAD=0,01

Як видно з результатів моделювання, мережа, що складається з 19 нейронів, не забезпечує гладкості функції апроксимації.

Розглянемо процес апроксимації при SPREAD = 1.

```
P = -1:.1:1;
T = [-.9602 -.5770 -.0729 .3771 .6405 .6600 .4609
   .1336 -.2013 -.4344 -.5000 -.3930 -.1647 .0988
   .3072 .3960 .3449 .1816 -.0312 -.2189 -.3201];
  GOAL=0.01;
>> SPREAD=1;
>> net=newrb(P,T,GOAL,SPREAD);
  net.layers{1}.size
NEWRB, neurons = 0, SSE = 3.69051
ans =
     6
>> X=-1:.01:1;
Y=sim(net,X);
hold on;
plot(X,Y);
title('Training Vectors');
xlabel('Input Vector P');
>> grid
```

Результати моделювання мережі представлено на рисунку 5.8.

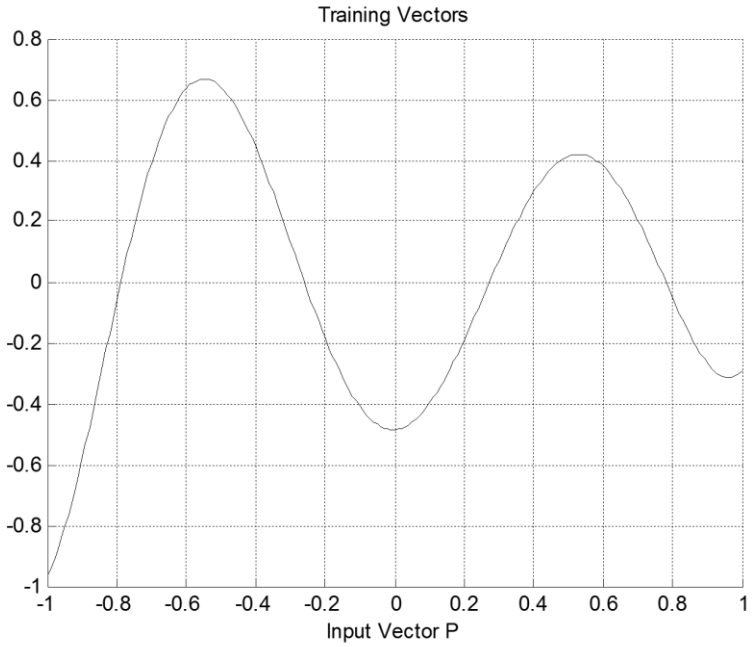


Рисунок 5.8 – Графік функції апроксимації при значенні SPREAD=1

Ця мережа складається з 6 нейронів схованого шару й достатньо добре апроксимує нелінійну залежність, задану множиною з 21 точки.

Нарешті установимо параметр впливу SPREAD=20.

```
P = -1:.1:1;
T = [-.9602 -.5770 -.0729 .3771 .6405 .6600 .4609
   .1336 -.2013 -.4344 -.5000 -.3930 -.1647 .0988
   .3072 .3960 .3449 .1816 -.0312 -.2189 -.3201];
  GOAL=0.01;
SPREAD=20;
net=newrb(P,T,GOAL,SPREAD);
  net.layers{1}.size
NEWRB, neurons = 0, SSE = 3.69972
ans =
    21          % Кількість нейронів схованого шару в мережі
X=-1:.01:1;
Y=sim(net,X);
hold on;
plot(X,Y);
title('Training Vectors');
xlabel('Input Vector P'); grid
```

Результат моделювання наведено на рисунку 5.9.

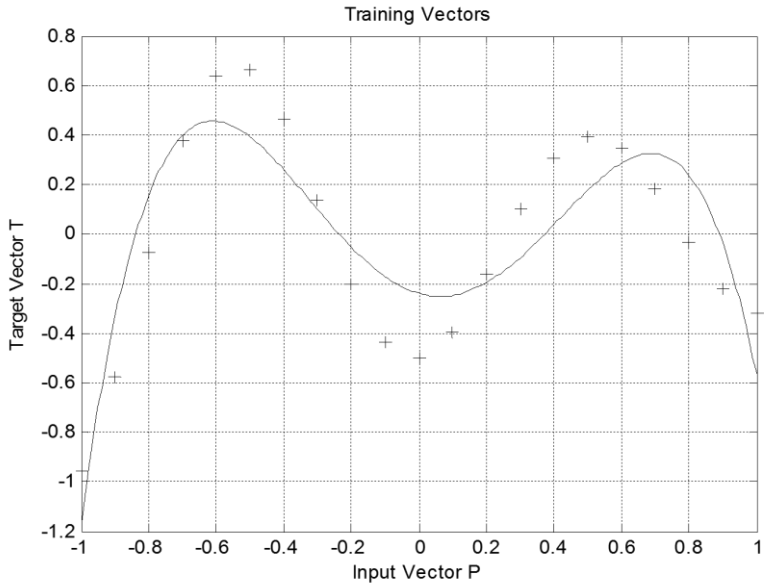


Рисунок 5.9 – Графік функції апроксимації при значенні SPREAD=20

З рисунка 5.9 видно, що при великому значенні параметра впливу SPREAD і великій кількості нейронів схованого шару (21 нейрон) функції активації перекриваються, і кожний базисний нейрон для всіх значень входу генерує вихідні значення, близькі до 1. Це приводить до того, що мережа не може забезпечити необхідну точність через обчислювальні проблеми.

Таким чином, за результатами виконаних досліджень можна зробити висновок про те, що параметр впливу SPREAD слід брати більшим, ніж крок розбивки функції навчальної послідовності, але меншим самого інтервалу її розбивки. Для розглянутого прикладу цей діапазон становить від 0,1 до 2.

5.3 Варіанти індивідуальних завдань та зміст і захист звіту

Для виконання роботи студенти одержують індивідуальні завдання, варіанти яких наведено в таблиці 5.1 і на рисунку 5.10.

Таблиця 5.1 – Варіанти завдань

Варіант	Діапазон вхідного вектора P		Діапазон мети T	
	від	до	від	до
1	-1	+1	0	+1
2	-2	+2	0	+2
3	-3	+3	-1	+1
4	-4	+4	-2	+2
5	-5	+5	-3	+3

На рисунку 5.10 наведено 4 виду графіків функцій, позначених буквами «а, б, в».

Варіант індивідуального завдання визначається буквою рисунка 5.10 і рядком таблиці. Наприклад, варіант "а-2" означає, що для індивідуального завдання графік функції наведений на рис. **5.10,а,** а вихідні дані наведені у рядку 2 таблиці 5.1.

У процесі виконання роботи необхідно виконати наступне.

1 Із застосуванням функції newrbe створити першу структуру мережі з нульовою помилкою й виконати її навчання.

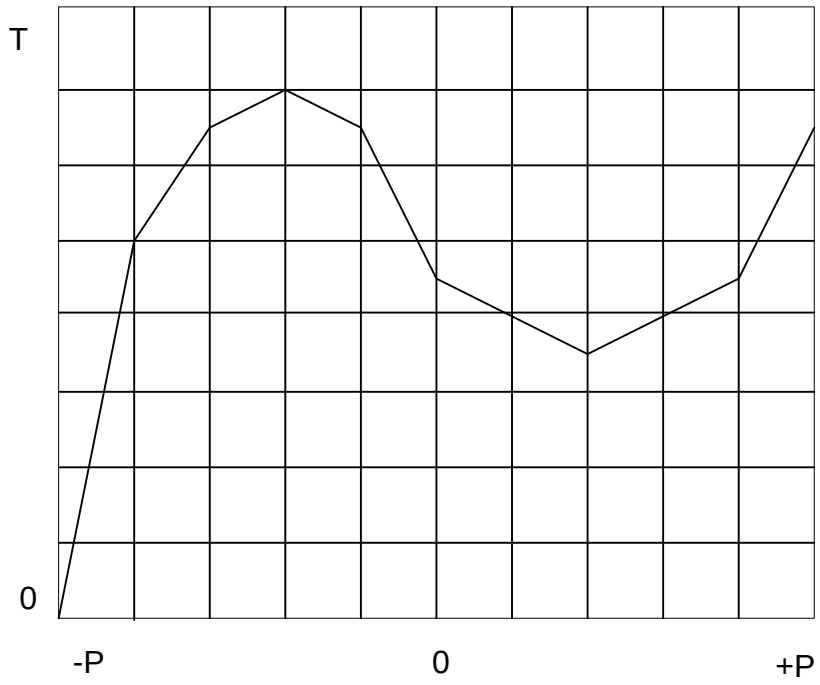
2 Із застосуванням функції newrb створити другу структуру мережі (із завданням кількості нейронів схованого шару) й зробити її навчання при різних значеннях параметра впливу SPREAD**.**

3 Побудувати графіки вихідних функцій мережі й зробити висновок про якість апроксимації.

Звіт по роботі повинен містити завдання, структуру мережі, програми й результати роботи мережі.

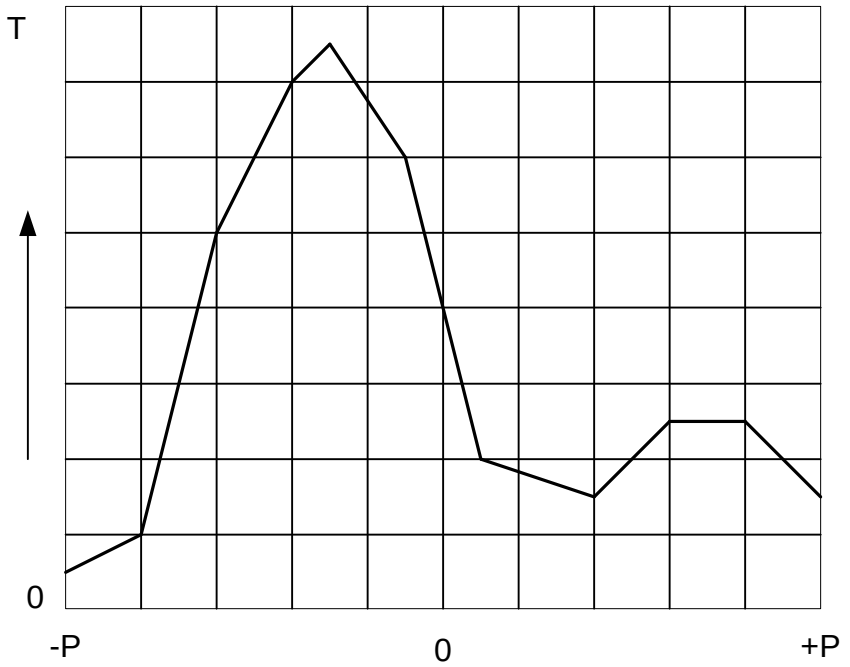
При захисті звіту студент повинен відповісти на запитання, що стосуються методики створення й застосування радіальної базисної нейронної мережі.

ВАРІАНТИ ФУНКЦІЙ ДО ЗАВДАНЬ

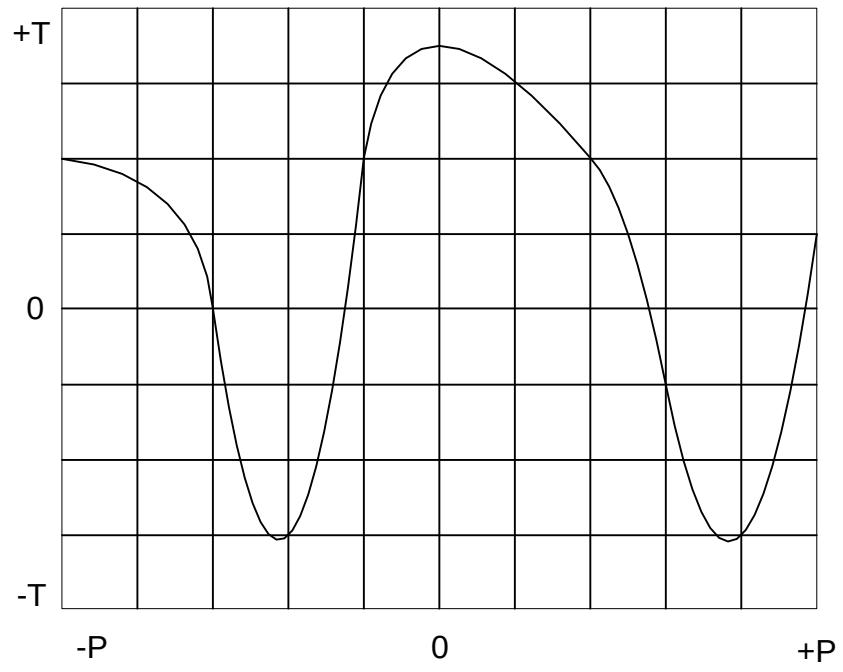


а

Рисунок 5.10, аркуш 1



б



в

Рисунок 3.10, аркуш 2

ЛІТЕРАТУРА

1. Морозов А.Д. Введение в теорию фракталов / А. Д. Морозов - Москва-Ижевск: Институт компьютерных исследований, 2002, 160 с.

2. Кроновер Р.М. Фракталы и хаос в динамических системах. Основы теории / Р. М. Кроновер - Москва: Постмаркет, 2000. - 352 с.

Навчальне видання

Методичні вказівки

до ком‘ютерного практикуму по дисципліні "Сучасні методи дослідження систем"

для студентів спеціальності 151 «Автоматизація та комп‘ютерно-інтегровані системи» усіх форм навчання

Укладач СЕРДЮК Олександр Олександрович

За авторською редакцією
Комп’ютерне верстання І. І. Дьякова

140/2019. Формат 60 х 84/16. Ум. друк. арк. 1,4.
Обл.-вид. арк. 0,94. Тираж 25 пр. Зам. №…..

Видавець і виготівник
Донбаська державна машинобудівна академія
84313, м. Краматорськ, вул. Академічна, 72.
Свідоцтво суб’єкта видавничої справи
ДК № 1633 від 24.12.2003