Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут»

ДЕЯКІ ПОЛОЖЕННЯ НЕЛІНІЙНОЇ ДИНАМІКИ. ТЕОРІЯ ХАОСУ. ФРАКТАЛЬНА ГЕОМЕТРІЯ

МЕТОДИЧНІ ВКАЗІВКИ
до виконання комп'ютерного практикуму
з курсу (MATLAB)

Затверджено Методичною радою НТУУ «КПІ»

Київ
«Політехніка»
2005

Деякі положення нелінійної динаміки. Теорія хаосу. Фрактальна геометрія: Метод. вказівки до викон. комп'ют. практикуму з курсу (MATLAB) / Уклад.: Т. В. Гнітецька, В. А. Гнітецький. – К.: ІВЦ "Видавництво «Політехніка»", 2005. – 32 с.

Гриф надано Методичною радою НТУУ «КПІ»

(Протокол № 10 від 23.06.2005 р.)

Навчальне видання

ДЕЯКІ ПОЛОЖЕННЯ НЕЛІНІЙНОЇ ДИНАМІКИ. ТЕОРІЯ ХАОСУ. ФРАКТАЛЬНА ГЕОМЕТРІЯ

МЕТОДИЧНІ ВКАЗІВКИ
до виконання комп'ютерного практикуму
з курсу (MATLAB)

Укладачі: *Гнітецька Тетяна Віталіївна*, канд. техн. наук, доц.
Гнітецький Віталій Анатолійович, канд. техн. наук

Відповідальний редактор: *В. В. Ванін*, д-р техн. наук, проф.

Рецензент: *Г. М. Коваль*, канд. техн. наук, доц.

Темплан 2005 р., поз. 2-098

За редакцією укладачів

Надруковано з оригінал-макета замовника

Підп. до друку 01.08.2005. Формат 60×84 1/16. Папір офс. Гарнітура Times.
Спосіб друку – ризографія. Ум. друк. арк. 1,86. Обл.-вид. арк. 3,09. Зам. №5-123 Наклад 100 пр.

Інформаційно-видавничий центр "Видавництво «Політехніка»" НТУУ «КПІ»
Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру видавців, виготівників і розповсюджувачів видавничої продукції ДК № 1665 від 28.01.2004
03056, Київ-56, вул. Політехнічна, 14, корп. 15,
тел./факс (044) 241-68-78, 241-66-64, e-mail: izdatelstvo@ntu-kpi.kiev.ua

1. *Вступ*

Головна мета методичних вказівок – організувати самостійну роботу студентів четвертого курсу ФМФ з курсу «Деякі положення нелінійної динаміки. Теорія хаосу. Фрактальна геометрія.» Реалізація в середовищі пакету MATLAB. ».

В процесі навчання студенти повинні отримати знання з створення та дослідження нелінійних динамічних систем а також вміти реалізовувати різноманітні алгоритми фрактальної геометрії та теорії хаосу. Зручним та сучасним для цієї роботи є система інженерних та наукових розрахунків MATLAB. Ця система дозволяє не тільки реалізовувати всі поставленні завдання, але і виконувати якісне графічне оформлення отриманих результатів. Крім того, ця система дозволяє розширювати коло розв'язуваних нею задач шляхом створення власних бібліотек та алгоритмів, а також доповнювати ці бібліотеки через Internet.

Методичні вказівки створені таким чином, що спочатку студенти мають можливість ознайомитися з основами роботи з пакетом MATLAB, далі вони вчаться створювати власні програми та бібліотеки в оболонці програмування системи MATLAB, і заключна частина присвячена безпосередньо моделюванню динамічних систем в системі MATLAB Simulink.

Лабораторні роботи підібрані таким чином, що студенти не тільки знайомляться на практиці з алгоритмами нелінійної динаміки та фрактальної геометрії, але і повністю використовують широкі можливості пакету MATLAB.

Засвоєні в цьому курсі знання дозволять студентам активно працювати і в інших галузях фізики та математики

2. *Командне вікно MATLAB*

Пакет MATLAB є дуже зручним інструментом для виконання математичних розрахунків, а також для проведення моделювання поведінки різних за складністю динамічних систем. Кожна математична функція або алгоритм реалізовано у вигляді файла з розширенням "m", доступ до якого можна отримувати через вікно дебагера. Таким чином, всі бібліотечні функції і алгоритми є відкритими, доступними для ознайомлення, а також для необхідної зміни під вимоги конкретного користувача. Мова програмування MATLAB схожа на мову C++, а самі програми пакету дозволяють доповнення модулями, реалізованими на C++. Програми, які створюються користувачем, мають те ж розширення і можуть поміщатися в ті чи інші стандартні бібліотеки, доповнюючи їх і роблячи файли користувача невід'ємною їх частиною, доступною для роботи. Можна також створювати власні бібліотеки за тією чи іншою ознакою.

При завантаженні програми MATLAB користувач потрапляє в режим роботи з командним рядком. Режим вводу команди ідентифікується символами ">>"(рис.1).

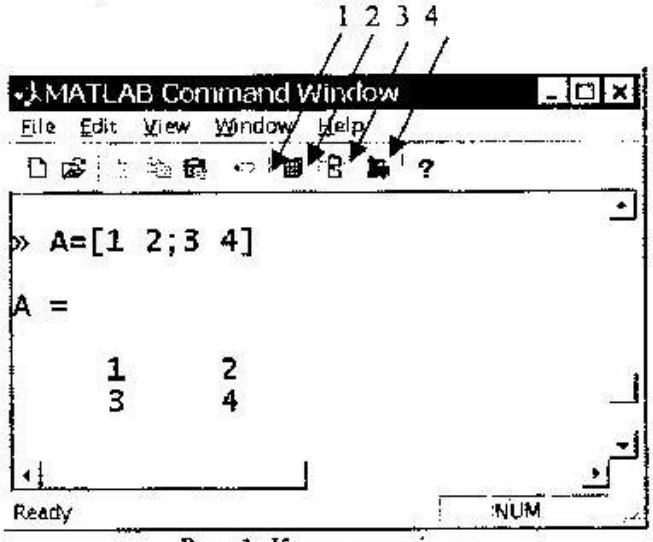

Рис. 1. Командне вікно

3. *Задання скалярних величин*

Задання змінних є простим і логічним: вводиться ім'я змінної і через знак "=" її значення, наприклад, a=3. Кожний рядок команди закінчується введенням "Enter". Після чого система підтверджує введення змінної відгуком

a=

3.

Для того, щоб система не відгукувалася на кожний рядок введенної команди, рядок необхідно завершувати ";".

Можна записувати кілька команд в один рядок, роздяляючи їх «;».

4. *Задання матриць*

Для роботи з матрицями слід дотримуватися наступних правил:

* елементи одного рядка вводяться через кому або пробіл;
* для відокремлення одного рядка матриці від іншого використовується ";";
* рядок з елементами матриці береться в квадратні дужки.

Наприклад,

>>A=[1 2; 3,4].

В результаті чого система відгукнеться

A=

1 2

3 4

Доступ до окремого елемента матриці реалізується через індекси рядка і стовбчика, заданими у круглих дужках після імені матриці, причому нумерація рядків та стовпчиків починається з "1". Наприклад, на командний рядок c=A(2,1), система відгукнеться

c =

3

Або, задавши A(2,2)=5, отримаємо матрицю

A=

1 2

3 5

Можна створювати багатовимірні матриці, задаючи елементи цих матриць як, наприклад, A(1,2,1)=3 і т.д., або ж заповнюючи їх програмним шляхом за допомогою реалізації певних алгоритмів, заданих користувачем.

5. ***Задання векторів з рівномірним кроком зміни значень***

Задання векторів виконується шляхом введення значення початку вектора, символа ":", кроку зміни по вектору, символа ":" та кінцевого значення вектора.

Якщо крок зміни по вектору дорівнює одиниці, його можна не вказувати, наприклад, командний рядок

>>f=1:4

сформує вектор

f=

1 2 3 4.

Командний рядок

>>f=1:2:10

сформує вектор

f=

1 3 5 7 9.

Задання вектора можна використовувати і при роботі з матрицями. Наприклад, запис A(:,1) означає, що ми отримаємо як результат значення першого стовпчика матриці A (значення рядка змінюється за вектором від першого до останнього, а значення стовпчика залишається незмінним і є одиницею). Можна використовувати довільні комбінації, наприклад, A(:,:), A(1:2,2) і інш.

Для того, щоб змінити послідовність рядків або стовпчиків, можна задати командний рядок D=A(:,[2,1,3,4]). Результатом буде перестановка першого і другого стовпчика в матриці A.

В якості елементів матриці або вектора можна використовувати вирази. Наприклад, результатом виконання командного рядка

>>f=[1+2, 3/4, 5*6]

буде

f=

3 0.75 30.

6. ***Робота з системою, та доступ до змінних***

Система розрізняє малі та великі літери. Тому змінні "a" та "A" вважаються двома різними змінними.

Для редагування командного рядка, який було введено неправильно або ж для повторного виконання команди слід використовувати клавішу "↑", за допомогою якої можна переглянути вже використані команди поточного сеансу роботи.

Для того, щоб в процесі роботи переглянути значення тієї чи іншої змінної, достатньо в командному рядку ввести її ім'я і система відгукнеться, вивівши на екран значення цієї змінної.

Введення десятичного значення робиться через крапку.

Якщо вводиться значення без конкретного присвоєння змінній або обчислюється вираз, результат якого не присвоюється змінній, значення потрапить в чергову змінну з ім'ям "ans", де і буде зберігатися доки не зміниться наступним "безіменним" значенням. Наприклад, командний рядок,

>>A(1,1)+A(1,2)+A(2,1)+A(2,2)

дасть результат суми заданих елементів матриці, який буде поміщено у змінну ans.

Для перегляду існуючих змінних використовуються команди:

who - виводить імена існуючих змінних;

whos - виводить імена змінних з їх характеристиками.

Доступ також можна отримати, використавши піктограму 1 з рис.1. або команди File/Show Workspace (Рис.2).

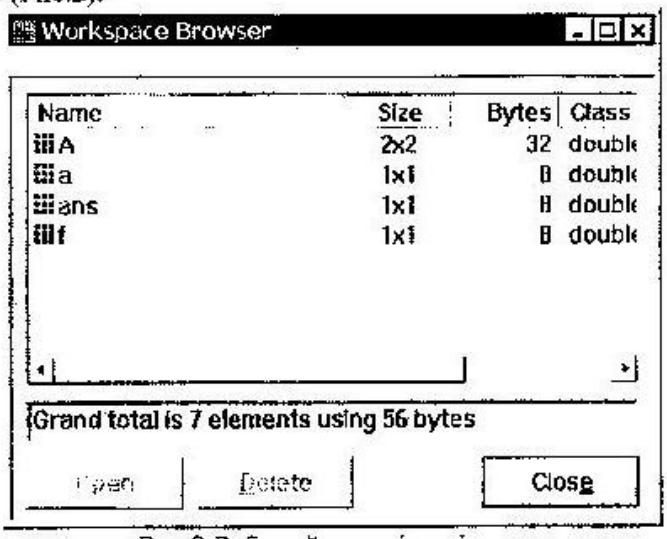

Рис.2 Робочий простір змінних

В цьому вікні змінну можна видалити кнопкою Delete або ж відредагувати використавши кнопку Open (Рис.3). Багатовимірні матриці таким чином не редагуються.

Виконавши подвійний клік на тій чи іншій змінній з вікна робочого простору відкриваємо вікно, в якому можна змінити як характеристики, так і значення змінних.

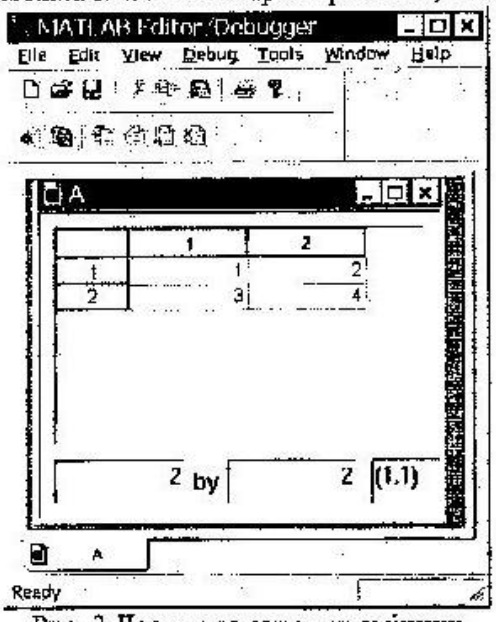

Рис. 3 Доступ до окремих змінних

Збереження робочої області виконується за допомогою команди File/Save Workspace As. Після чого змінні з поточного сеансу роботи будуть збережені у файлі з розширенням mat, та можуть бути завантажені командою File/Load Workspace в наступному сеансі роботи.

Очистити значення тієї чи іншої змінної можна за допомогою команди clear в командному рядку з наступним введенням імені цієї змінної. Якщо ввести цю команду не вказуючи необхідну змінну, будуть видалені всі поточні змінні.

7. *Деякі функції роботи з матрицями та скалярними величинами*

MATLAB має широкі можливості, які забезпечують роботу з матрицями. Власне і сама назва пакету розшифровується як лабораторія по роботі з матрицями.

Розглянемо тільки деякі з широкого класу функцій роботи з матрицями.

Нехай матриця A задана. Тоді

- транспонування матриці виконується як B=A';
- функція det(A) обчислює детермінант матриці;
- функція eig(A) знаходить власні значення матриці;
- функція diag(A) виділяє діагональ матриці;
- функція sum(A) знаходить суму елементів кожного стовпчика матриці. Якщо задати уточнення, то можна знайти суму певних елементів, наприклад, sum(A(:,1)) - сума елементів першого стовпчика матриці, а суму всіх елементів матриці можна знайти як sum(sum(A)), а sum(diag(A)) знаходить слід матриці;
- функція inv(A) знаходить інверсну (зворотнью) матрицю;
- функція fliplr(A) виконує дзеркальне відображення матриці зліва направо;
- можливо формувати матриці з наперед заданими властивостями, наприклад, для формування квадратної матриці, сума кожного рядочка, стовпчика і діагоналі якої є константою виконується за допомогою функції magic(n) - де n – число, яке задає розмірність квадратної матриці;
- функція ones(n), або ones(m,n) формує масив одиниць розмірністю n×n або m×n;
- функція rand(n) формує масив випадкових величин, розподілених по рівномірному закону, розмірністю n×n;
- функція zeros(n,m) формує матрицю нулів, заданої розмірності;
- функція length(s) знаходить довжину вектора s;
- функція sort(s) виконує сортування елементів масиву s у порядку з зростання;
- функція min(s), max(s) знаходить мінімальний або максимальний елемент масиву s, відповідно;
- функція interp(x,y,xi) будує *кусочно-лінійну* інтерполяційну криву для одновимірного масиву y заданого на сітці x. Причому вихідний масив може бути перевизначений на більш мілкій сітці xi, наприклад,

x=0:10; y=sin(x); xi=0:0.25:10; yi=interp1(x,y,xi).

Знаки +, -, *, /, ^ - виконують відповідно операції суми, віднімання, множення, ділення та піднесення у степінь. Якщо операція застосовується до скалярних величин, то вона проводиться для скалярних величин. Якщо операндами є матриці, то операції виконуються за відповідними алгоритмами роботи з матрицями. Якщо одним операндом є скаляр, а другим матриця, то кожний елемент матриці буде, наприклад, помножено або поділено на заданий скаляр. Наприклад, A*pi - домножить матрицю на pi, де pi зарезервована константа 3.14.

Інколи виникає необхідність не перемножити матрицю на матрицю, а перемножити відповідні елементи однієї матриці на відповідні елементи другої матриці. В таких випадках використовуються алгоритми поелементного множення, які ідентифікуються "." перед необхідним операндом. Зверніть увагу на відмінність результатів:

Нехай x=[1 2 3]; y=[4 5 6]

тоді командний рядок x*y' дасть результат

ans=

32,

тоді як командний рядок x.*y дасть результат

ans=

4 10 18.

Командний рядок x*y дасть помилку, оскільки в алгоритмі множення матриць повинні співпадати кількість рядків та стовпчиків у відповідних операндах.

Також помилковим буде задання x^y, тоді як командний рядок x.^y дасть результат

ans=

1 32 729.

Не менш широкою є і бібліотека функцій роботи зі скалярними величинами. Розглянемо деякі з запропонованих функцій.

Нехай задана змінна f, тоді:

- sqrt(f) - квадратний корінь від змінної;
- exp(f) - експонента;
- log(f) - натуральний логарифм;
- log2(f) - логарифм з основою 2;
- log10(f)-десятичний логарифм;
- тригонометричні функції sin(f), cos(f), asin(f), acos(f), tan(f), atan(f), sinh(f) -синус, косинус, арксинус, арккосинус, тангенс, арктангенс, гіперболічний синус та інші. Тригонометричні функції сприймають значення у радіанах;
- abs(f) - модуль;
- fix(f) - відсікання дробової частини;
- floor(f) -округлення до цілого меншого за f.

Слід зазначити, що MATLAB має алгоритми обчислення цих функцій і для матриць. Доданням в кінець функції "m", ми ідентифікуємо її як функцію для матриці.Так, наприклад, sqrtm(A), expm(A), logm(A) виконають необхідні обчислення для матриці A.. Якщо спеціальної функції для матриці не існує, як у випадку тригонометричних функцій, використовують функцію funm(A,'sin'), параметрами якої є матриця (не вектор), для якої проводиться обчислення і відповідна функція застосування. Якщо вказані в переліку вище функції застосовуються для векторів або матриць, вибрана операція буде застосована для кожного елемента вектора або матриці окремо, наприклад, для матриці

a=[1 2;3 4];

команда

» sqrt(a)

дасть

ans =

1.0000 1.4142
1.7321 2.0000,

тоді як для команди

» sqrtm(a)

результатом буде

ans =

0.5537 + 0.4644i 0.8070 - 0.2124i
1.2104 - 0.3186i 1.7641 + 0.1458i,

тобто матриця ans, яка при перемноженні сама на себе дасть вихідну матрицю:

» ans*ans

ans =

1.0000 + 0.0000i 2.0000 + 0.0000i
3.0000 - 0.0000i 4.0000.

8. *Робота з комплексними числами*

Комплексне число задається як i або j. Причому, якщо необхідно задати матрицю, елементами якої є комплексні числа, то можна задавати безпосередньо кожний елемент, як комплексний, або ж окремо дійсну частину матриці плюс комплексну частину матриці.

Два наступних командних рядка задають одну і ту ж матрицю різними способами, наприклад,

```
>>D=[1  2; 3  4]+i*[5  6;  7  8];
>>D=[1+5i, 2+6i; 3+7i; 4+8i];
```

При обчисленні виразів до комплексних чисел автоматично застосовуються алгоритми роботи з комплексними числами.

Можна виконати візуалізацію комплексних функцій. Для цього необхідно:

1). задати множину значень змінної, на якій буде обчислюватися комплексна функція за допомогою функції cplxgrid, наприклад, cplxgrid(20) ;

2). за допомогою функції cplxmap, виконати візуалізацію результуючої поверхні, наприклад cplxmap(z, z.^2), параметрами якої є множина значень, яка задана в попередньому пункті, та безпосередньо комплексна функція (Рис.4).

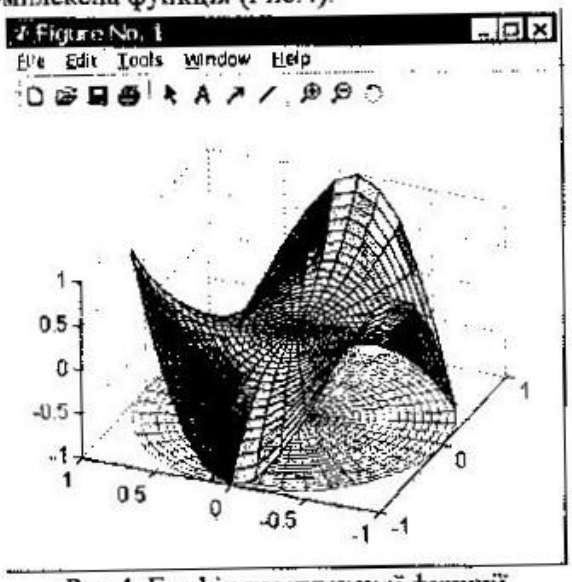

Рис.4. Графік комплексної функції

9. *Використання довідкової системи*

Система MATLAB дозволяє розглянути можливості запропонованих бібліотек за допомогою команд demo, tour, або використовуючи піктограму 4 рис 1. У вікні, що відкриється в результаті натискання піктограми обирається необхідний розділ та команда у цьому розділі. В результаті буде виведено коротку характеристику необхідної команди та приклади її застосування. Знайти необхідну команду можна також ввівши її назву у тому ж вікні замість рядка "MATLAB Help Topics " і натиснувши Enter. Якщо така команда існує, у вікно буде виведена необхідна інформація.

Можна використовувати допомогу безпосередньо із командного вікна, ввівши командний рядок

help "назва команди", наприклад,

```
>>help exp.
```

10. *Команди візуалізації векторів*

Для виведення графіка необхідно задати функцію та множину значень, для яких буде виконана візуалізація функції.

В таблиці 1 зведено деякі команди, що виконують візуалізацію векторів та їх короткі характеристики.

Таблиця 1

Назва функції	Призначення
plot(x,y)	будує лінійний графік залежності векторів x по y, причому в якості параметрів можна вказувати необхідний колір, символ виведення та тип лінії, наприклад, plot(x,y,'r+--')
loglog(x,y)	логарифмічний графік вектора по осям x,y
semilogx(x,y)	логарифмічний графік вектора тільки по осі x
semilogy(x,y)	логарифмічний графік вектора тільки по осі y
fill(x,y,<колір>)	замальовує полігон, вершини якого задаються векторами x,y та кольором c, наприклад, x=[1 3 5 2]; y=[3 1 4 5]; fill(x,y,1)
polar(x,y)	графік вектора в полярних координатах
bar(x)	будує послідовні колонки висотою значень вектора
stairs(x)	будує "сходинки" висотою значень вектора
errorbar(x,y)	будується графік з послідовними значеннями вектора x з міткою похибки заданою вектором y
hist(y,n)	будує гістограму для n інтервалів (y - кількість елементів, що потрапляють у заданий інтервал)
comet(y)	будує графік у вигляді рухомої голови комети
comet(x,y)	рух голови комети по траєкторії, заданій x,y
fplot	будує графік функції , виконуючи згладжування по точкам функції

Приклад використання:

задаємо область значень, на яких буде обчислюватися функція

>>i=1:0.1:5;

використовуємо функцію візуалізації

>>plot(i, sin(i)).

Якщо задавати функцію одним параметром, то вона будується в залежності від свого порядкового номеру. Наприклад, команда plot(i), побудує першу точку з координатами 1.0, 1 наступну 1.1, 2; 1.2,3 і т.д.

11. ***Команди візуалізації матриць***

Для візуалізації матриці необхідно задати множину значень (матрицю значень), на яких буде обчислюватися функція, яка задає поверхню.

Короткий перелік деяких функцій, які застосовуються для візуалізації матриць наведено в Таблиці 2.

Таблиця 2.

Назва функції	Призначення
contour	виводить контурну карту поверхні за заданою кількістю рівнів на площині, наприклад, contour(z,20) - 20 рівнів по поверхні z
contour3	виводить тривимірну контурну карту поверхні за заданою кількістю рівнів
pcolor	виводить на площині поверхню "псевдокольорами" коли різна "висота" поверхні ідентифікується іншим кольором.
image	виводить матрицю з існуючим малюнком
mesh	виводить поверхню у вигляді сітки
meshc	виводить поверхню у вигляді сітки, що доповнена площиною з зображенням контурної карти поверхні
surf	виводить поверхню
surfc	виводить поверхню з доповненням зображення контурною картою
surfl	виводить поверхню з доданням джерела освітлення

Існують функції для виведення базових тривимірних фігур, таких як сфера та циліндр, наприклад, sphere,cilinder, відповідно.

Для задання області значень використовується функція meshgrid, яка і формує матрицю значень для наступних обчислень.

наприклад,

>>[u,v]=meshgrid(0:pi/16:2*pi), якщо області мають однакові межі,

або

>>[x,y]=meshgrid(0:pi/16:2*pi, 0:pi/8:pi), якщо області мають різні межі.

В результаті будуть сформовані матриці x,y, послідовний і сукупний перебір значень яких і сформує поверхню для заданої функції, наприклад:

>>surf(x.^2+y).

Оскільки x та y є матрицями, слід уважно задавати функцію, обираючи операції роботи з матрицями чи операції поелементної роботи з матрицями.

Наприклад, використовуючи параметричні рівняння еліпсоїда (Рис.5), задасмо

[u,v]=meshgrid(0:pi/16:2*pi);

» x=sin(u).*cos(v);

» y=sin(u).*sin(v);

» z=cos(u);

» surf(x,y,z)

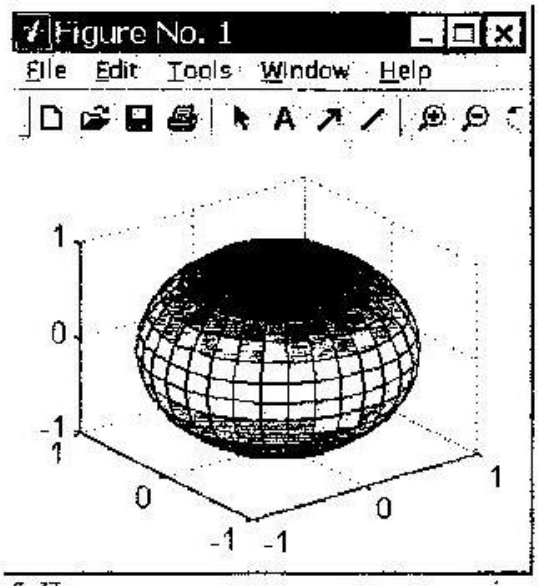

Рис.5. Поверхня параметрично заданого еліпсоїда

12. *Створення анімації*

За допомогою команд getframe можна копіювати зображення з поточного вікна фігури у якості послідовних елементів деякої матриці. Далі за допомогою команди movie виконується програвання набраних зображень задану кількість разів.

Наприклад:

>>M(:,1)=getframe

>>% змінити зображення у вікні фігури

>>M(:,2)=getframe

>>% і т.д.

>>movie(M,10)

13. *Оформлення графіка*

Оформлення графіка можна виконувати як за допомогою командного рядка, так і використовуючи можливості, запропоновані безпосередньо у вікні фігури та зведені в підменю "Tools". Прямо у вікні можна виконати збільшення, зменшення графічного

фрагмента, обернути зображення, а також нанести необхідний текст, допоміжні лінії та стрілки. Через меню "Tools", попередньо обравши елемент, який буде редагуватися інструментом "стрілка" з піктографічного меню, можна отримати доступ для встановлення підписів осей та малюнка в цілому, задання кордонів значень по відповідним осям для відображення зображення, а також змінити колір, товщини та тип ліній окремих елементів зображення, обрати необхідний шрифт для надписів.

Для перенесення зображення через буфер в інші прикладні програми використовується команда Edit/Copy figure у вікні фігури.

Для оформлення зображення через командний рядок використовуються наступні команди:

grid - виводить у вікно фігури координатну сітку;

title('назва графіка') - задає надпис зображення у вікні фігури;

xlabel('надпис') - задає надпис вздовж осі x;

ylabel('надпис') - задає надпис вздовж осі y;

xlim([a,b]) – задає кордони відображення від a до b по осі x;

ylim([a,b]) – задає кордони відображення від a до b по осі y;

zoom on (off) – включає (виключає) режим збільшення (переміщення миші вверх) або зменшення (при переміщенні миші вниз) графіка у вікні фігури;

rotate3d – вмикає режим обертання графіка.

Колір для графіка можна задати безпосередньо при його побудові за допомогою додаткового параметра, наприклад,

plot(sin(i),'red')

Графік також можна вивести окремими символами. Наприклад, plot(sin(i),'*') - кожна точка графіка буде виведена символом "*".

Для поверхонь також можна змінити тип кольору за допомогою команди colormap(параметр), де в якості параметра передається один з бібліотечних варіантів, таких як gray, cool, bone, cooper, pink, flag, prism, jet та інш.

Починаючи з п'ятої версії пакету, можливе оформлення графіка безпосередньо через вікно фігури, використовуючи меню Tools, або відповідні піктограми для збільшення, зменшення або ж нанесення надписів на графік. Так, наприклад, вікно Tools/Axes Properties зображене на Рис.6 дозволяє задавати назву графіка, надписи вздовж осей, межі відображення та тип графіка.

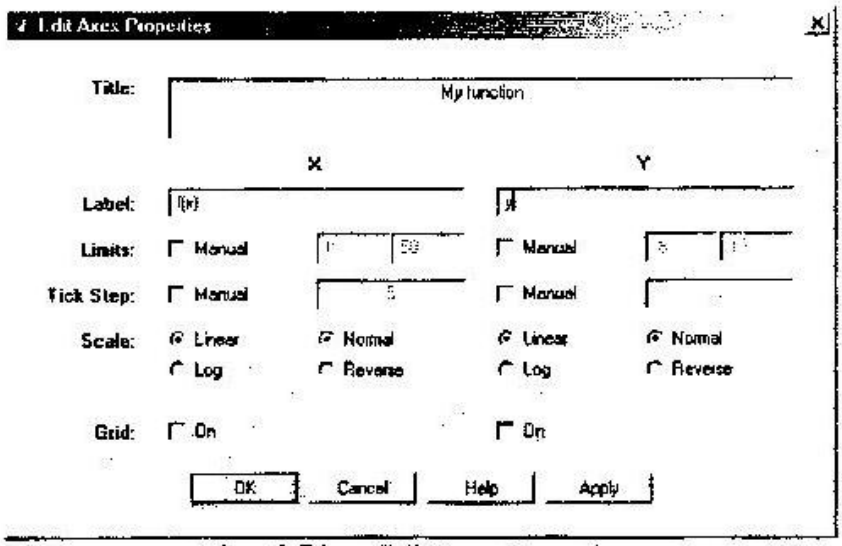

Рис.6. Вікно Tdit Axes Properties

Можна розбивати вікна фігури на кілька підвікон за допомогою команди subplot, де в якості параметрів передаються кількість підвікон по вертикалі, по горизонталі та номер підвікна, яке вважається поточним і буде приймати наступне зображення. Нумерація підвікон відбувається вертикальними рядами зверху вниз та зліва на право.

Наприклад,

>>subplot(2,2,1) розбиває вікно фігури на чотири підвікна та визначає перше підвікно поточним. Далі виконується необхідний вивід в поточне вікно і поточним встановлюється наступне вікно >>subplot(2,2,1) і т.д. (рис.7).

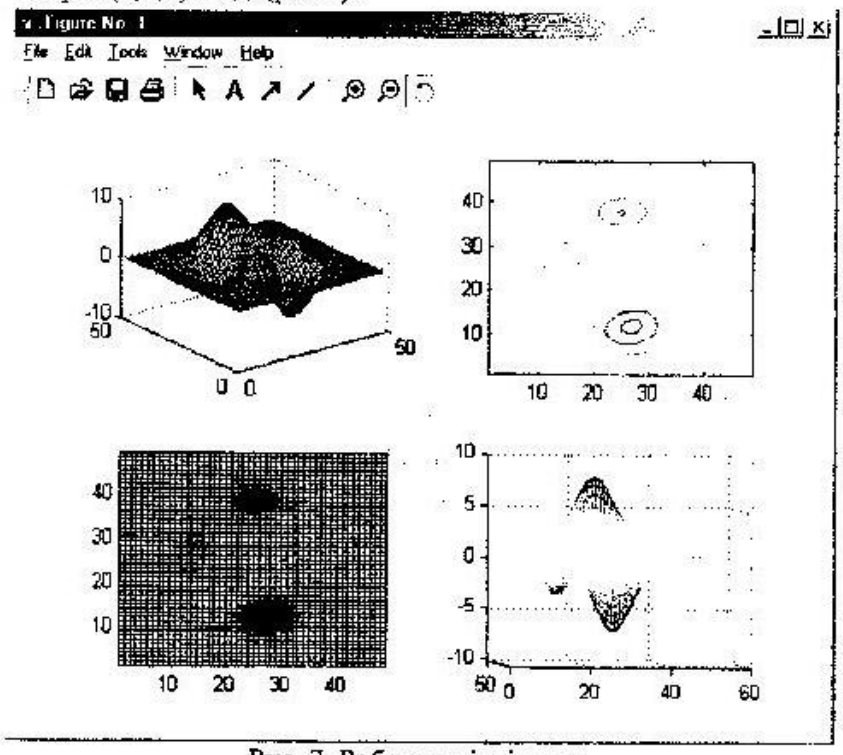

Рис. 7. Робота з підвікнами

Можна також виводити кілька зображень в вікно фігури або у підвікно. Для команди hold встановлюється значення on, відключення – off. Наприклад, командою

>>hold on встановлюється режим, при якому наступний графік буде виведено в те ж вікно фігури.

Отриманий малюнок можна зберегти в окремому файлі з розширенням fig виконанням команди File/Save Figure. Також можна завантажувати вже існуючі малюнки, файли яких мають таке ж розширення.

14. *Створення script -файлів та функцій*

Скріпт - файл є послідовністю команд, які зберігаються у зовнішньому файлі.

На відміну від скріпта функція може приймати зовні параметри з наступним їх використанням. Крім того, змінні, які використовуються в процесі роботи скріпт - програми додаються в область значень параметрів системи, а змінні, які використовуються функцією є локальними, вони знищуються після завершення роботи функції і не мають перекриття з областю параметрів системи.

Скріпт - файли та функції створюються за допомогою Editor/Debugger, який викликається піктограмою "чистий лист" в командному вікні пакета. Editor/Debugger є оболонкою для створення та відлагоджування програм на внутрішній мові програмування для пакету MatLab, яка є подібною за своїми структурами до мови С++.

Рядок, який починається зі знаку "%", сприймається в програмі як коментар.

Задання змінних є аналогічним до задання змінних через командне вікно і не потребує їх попереднього об'явлення у якості змінної того чи іншого типу.

Розглянемо деякі структури мови програмування для пакету MATLAB, наприклад,

структура перевірки умови:

if умова

послідовність команд

else
послідовність команд
end,
структура виконання циклу:
for a=a1:step:a2
послідовність команд
end
структура виконання циклу по умові:
while умова
послідовність команд
end
структура розгалудження:
switsh <вираз>
case <значення>
послідовність команд
case…

otherwise
послідовність команд
end
переривання циклів for та while:
break

Рекомендується також використовувати замість команд організації циклів обчислення значень для векторів, оскільки останній варіант працює в сотні разів швидше.

Наприклад, скріпт-файли

% варіант 1 n=0; for i=1:10 n=n+1 y(n)=i^2 end	% варіант 2 i=i:10 y=i^2

Варіант 2 працює в 200 разів швидше.

Файл-функції починаються ключовим словом function після чого іде ім'я змінної, яка передається як результат виконання функції, і через знак "=" назва функції (краще така ж як і назва файла) та в круглих дужках параметри, що передаються у функцію.

Коментарі, що вводяться одразу за першим рядком, сприймаються як довідкова інформація до поточної функції і можуть бути викликані з командного рядка через команду help "ім'я функції".

Далі записують тіло функції. Наприклад,

```
function k=fibonachi(n)
% обчислення заданого числа з ряду Фібоначі
if n>2
        k=fibonachi(n-1)+fibonachi(n-2);
else
        k=1;
end
k
```

Нагадуємо, що як і для командного рядка кранка з комою в кінці рядка означає подавлення виводу відгуку виконання команди на екран. І, навпаки, якщо необхідно вивести на екран значення деякої змінної, достатньо просто ввести її ім'я або не завершувати

необхідний рядок крапкою з комою. Так останній рядок функції виводить на екран значення обчисленого k. В даному прикладі також використано рекурсивний виклик функції (коли функція викликає сама себе при різних значеннях вхідних параметрів). Файл функції або створений скріпт - файл зберігається у вказаному користувачем каталозі з розширенням m.

Скріпт-файл можна запускати командою File/Run Script.

Файл-функцію викликають за іменем з командного рядка, вказавши необхідні параметри для її роботи, наприклад,

fibonachi(3),

або запускають безпосередньо з вікна Editor/Debugger командою Tools/Run.

Якщо функція не викликається за іменем, необхідно встановити шлях до каталога, де функція була збережена. Це можна зробити за допомогою команди path, наприклад,

path(path,'D:\MATLAB\WORK') - до існуючого переліку шляхів додає ще один до робочого каталога.

Шлях можна встановити також через вікно, що викликається піктограмою 3 рис.1., де у спеціальному вікні вказується шлях до необхідного файла (рис.8).

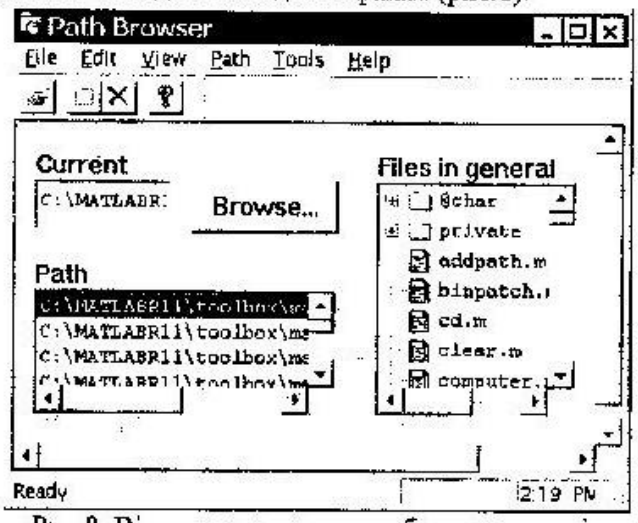

Рис.8. Вікно встановлення робочих каталогів

Скріпт-файл також можна викликати за іменем, якщо всі шляхи для його роботи встановлено.

15. *Робота з поліномами*

Нехай необхідно задати поліном вигляду $y=x^3-30*x+30$. Для цього необхідно сформувати вектор, який містить значення при степінях змінної полінома у порядку зменшення. Так у нашому випадку p=[1 0 -30 30].

Для обчислення значень полінома на заданному інтервалі використовується функція polyval(), наприклад,

```
>>x=-8:0.1:8;
>>y=polyval(p,x)
>>plot (x,y)
>>grid.
```

Для обчислення коренів полінома використовується функція roots(), наприклад,

```
>>result=roots(p)
```

та виведення цих коренів на попередньо виведений графік полінома

```
>>hold on
>>plot(result,[0,0,0],'*').
```

Для обчислення площі полінома на інтервалі, наприклад, між двома коренями, необхідно створити файл функції, яка задає поліном:

```
function y=poly(x)
p=[1  0  -30  30]
y=polyval(p,x).
```

Після створення файла функції можна використовувати функцію quad, де в якості одного з параметрів задається ім'я файла функції для полінома, наприклад,

```
area=quad('poly',result(1),result(3)).
```

16. *Розв'язок систем лінійних та нелінійних рівнянь*

Для розв'язку систем лінійних та нелінійних рівнянь використовується функція solve. Зліва від знаку "дорівнює" вказуються змінні, відносно яких повинні розв'язуватися рівняння. Всі інші змінні будуть сприйматися як константи. Необхідні рівняння задаються як параметри функції solve, наприклад,

```
x=solve('a*x^2+b*x+c=0').
```

Якщо рівняння дорівнює "0", знак рівності і "0" можна не вказувати, наприклад,

```
x=solve('a*x^2+b*x+c').
```

Для кількох змінних задається, відповідно,

```
[x,y]=solve('x^2+x*y+y=3','x^2-4*x+3=0').
```

Для розв'язку систем лінійних та нелінійних диференціальних рівнянь використовується функція dsolve, де перша похідна позначається через "D", друга похідна через "D2". Початкові значення, якщо вони є, додаються в якості останнього параметра функції.

Наприклад,

```
y=dsolve('Dy=-a*y')
```

або з початковими значеннями

```
y=dsolve('Dy=-a*y','y(0)=1')
y=dsolve('D2y=-a^2*y','y(0)=1, Dy(pi/a)=0');
y=dsolve('(Dy)^2+y^2=1','y(0)=0').
```

У випадках, коли аналітичного розв'язку системи диференціальних рівнянь не існує, слід застосовувати методи чисельного інтегрування.

Одним із методів застосування чисельного інтегрування є використання команди dee. Після її використання відкривається вікно (рис.9), в якому містяться приклади розв'язку систем нелінійних диференціальних рівнянь.

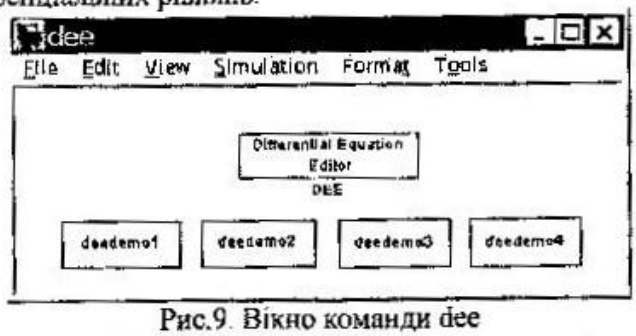

Рис.9. Вікно команди dee

Розглянемо правила вводу рівнянь на налагоджування системи під розв'язок конкретної задачі на одному із запропонованих прикладів (deedemo2). Для цього виконаємо подвійний клік на кнопці обраного прикладу.

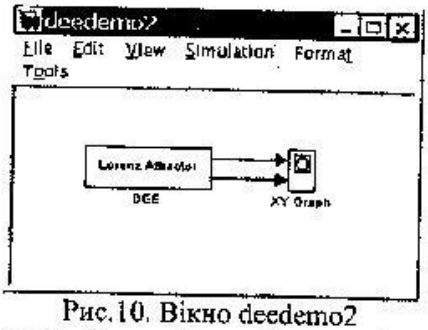

Рис.10. Вікно deedemo2

У вікні (рис.10) знаходиться два блоки. Перший з них містить безпосередньо систему диференціальних рівнянь, другий є блоком для візуалізації результатів обчислення (див. розділ далі). Розглянемо детально перший блок, для цього виконаємо на ньому подвійний клік.

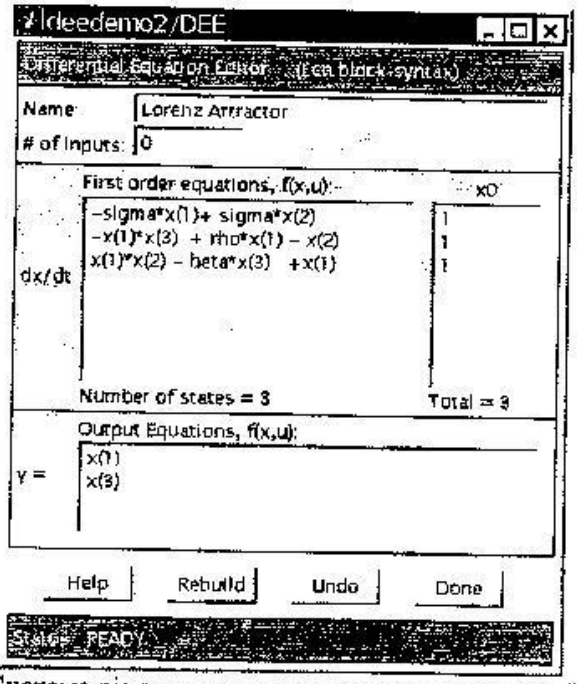

Рис.11. Система рівнянь для знаходження аттрактора Лоренца

Перше поле Name задає ім'я для обраної системи рівнянь і може бути довільним.

Поле # of inputs задає кількість входів в систему (вони відображаються на першому блоці рис.11). Тобто система може бути відкритою і приймати в процесі інтегрування ззовні значення тих чи інших змінних. В даному випадку зовнішніх входів немає.

Поле dx/dt задає безпосередньо систему диференціальних рівнянь. Рівняння мають бути першого порядку. Змінні іменуються як елементи вектора x, тобто x(1), x(2), x(3) і т.д. Першим задається рівняння по dx(1)/dt, причому "dx(1)/dt=" вважається введеним, тому у вікні задається тільки права частина рівняння. Другим вводиться рівняння dx(2)/dt і т.д. Початкові значення задаються в підвікні "x0" знову ж таки в порядку зростання індексів по вектору x. В самому нижньому вікні задаються змінні, які будуть подаватися на виходи з блоку системи Лоренца. Змінні sigma, beta, rho є константами і їх значення вже введено в область значення параметрів системи MATLAB.

Після заповнення (ознайомлення) системи натискаємо "Done" і повертаємося в попереднє вікно.

Другим важливим етапом є вибір метода інтегрування, задання кроку інтегрування та часу інтегрування системи. Командою Simulation/Parameters... відкривається вікно (рис.12), в якому і виконуються необхідні налагоджування системи чисельного інтегрування.

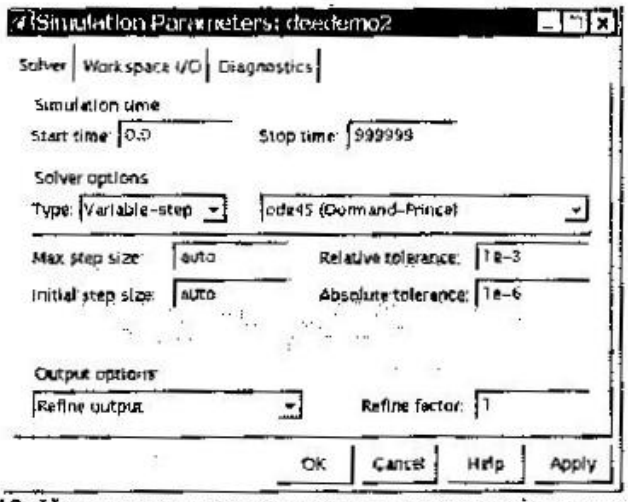

Рис.12. Налагоджування системи чисельного інтегрування

Зверху задається початковий і кінцевий час інтегрування системи в секундах (в нашому випадку від 0.0 до 999999). В області "Solver options" обирається тип метода інтегрування зі змінним або з фіксованим кроком. Відповідно до обраного типу пропонується обрати і метод інтегрування (для нашого прикладу ode45 (Dormand-Prince)). Нижнє поле відповідає за налагоджування конкретного метода і може мати різний вигляд. Якщо необхідно в цьому полі задається крок інтегрування. На інших вкладках з цього ж вікна можна встановити додаткові вхідні та вихідні значення.

Останнім етапом є запуск системи на інтегрування. Запуск виконується командою Simulation/Start вікна deedemo2. Після чого результати обчислення будуть виводитися в обраний блок виводу. В даному випадку на графік.

Якщо система має багато параметрів, їх можна підключити зовнішнім файлом у вікні системи deedemo2 командою Edit/Block Properties в полі Open Function. Це ж вікно містить багато команд для редагування зображення самих блоків. Наприклад, обертання блоків, нанесення надписів та підписів, задання кольору, масштабування зображення. При необхідності користувач може їх розглянути самостійно.

Для створення власної системи і її інтегрування у вікні dee або у вікні відкритої системи рівнянь обирається команда File/New.../Model, після чого блок "Differential Equation Editor DEE" перетягується за допомогою мишки у нове вікно, після чого виконується введення системи диференціальних рівнянь та налагоджування системи для її інтегрування, як це вказувалося вище.

Метод dee є достатньо зручним, коли мова іде про систему диференціальних рівнянь з неперервними правими частинами. Коли ж виникає задача розв'язання системи диференціальних рівнянь з розривними правими частинами, більш зручним є метод Simulink. За допомогою цього метода можна створювати різні за складністю моделі динамічних систем, використовуючи широкі бібліотеки запропонованих блоків, в тому числі і різних нелінійностей. Вікно моделі Simulink є схожим до вікна моделі dee і має ті ж вікна вибору та налагоджування метода інтегрування. Виклик метода Simulink можна реалізувати через натискання піктограми 3 у командному вікні MatLab (рис.1) або ж командою simulink в командному вікні. Відкривається вікно бібліотек, в кожній з яких знаходяться підбібліотеки з обраного розділу. На останньому рівні знаходяться безпосередньо блоки, з яких і будується модель користувача. Внизу вікна відображається вигляд обраного блоку та його коротка характеристика (рис.13).

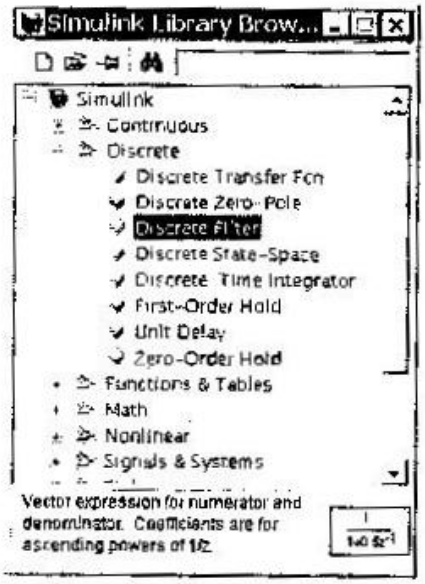

Рис 13.Бібліотека Simulink

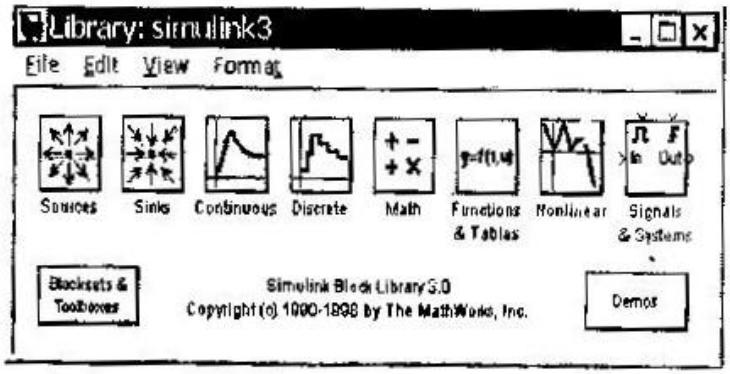

Рис 14.Бібліотека Simulink3

Аналогічним за складом бібліотек є вікно, що викликається командою simulink3 (рис.14). Воно має вигляд бібліотеки, як було зображено в попередніх версіях пакету. Подвійний клік на піктограмі обраної підбібліотеки з цього вікна відкриває доступ до вікна з блоками.

Для створення моделі користувача необхідно обрати команду File/New.../Model. Якщо необхідний блок з бібліотеки обрано, мишкою його перетягують у вікно моделі. Безпосередньо у вікні моделі (не в бібліотеці) виконується налагоджування блоку. Подвійний клік на блоці відкриває вікно налагоджування блоку , яке має різний вигляд в залежності від обраного блоку. В цих вікнах, зокрема, можуть задаватися кількість входів у блок, функції обчислення, величина змінних і т.і. Блоки, як правило, мають входи (стрілки направлені в середину блоку) та виходи (стрілки спрямовані назовні блоку). Для того, щоб сполучити вхід одного блока з виходом іншого, достатньо, не відпускаючи ліву кнопку миші, провести лінію від виходу до входу. Якщо сполучення виконалося, на екрані з'явиться лінія, що сполучить обрані блоки і чорна стрілка на вході необхідного блоку. Лінія сполучення може бути ламаною і проводитися в кілька етапів (з відпусканням миші). Блоки переміщуються на екрані за допомогою транспортування мишою. Якщо вони вже мають сполучення з іншими блоками, зв'язки не порушуються. Як і при роботі з методом dee, через меню View та Format можна виконувати зміну розміру, кольору, виконувати обертання і т.і. для блоків. Для обрання групи блоків достатньо не відпускаючи курсор миші розтягнути навколо них "прямокутник" або обирати кліками послідовно, утримуючи клавішу Shift. Для нанесення на полі моделі коментарів необхідно виконати подвійний клік на полі в необхідному місці, після чого вводити текст коментаря. Для зміни назви окремого блока в полі моделі виконується клік на назві блока, після чого вносяться необхідні зміни. На робочому полі моделі не може бути два блоки з однаковими назвами, тому по замовчуванню блоки з однаковую назвою маркуються індексами 1,2, і т.д.

Розглянемо характеристики окремих блоків з розділів бібліотеки в порядку запропонованому у вікні Library Simulink3 (рис.14). Бібліотеки з тією ж назвою знаходяться і в Library Simulink (рис.13). Не маючи можливості детально розглянути всі блоки, обмежимося необхідними для виконання завдань, поставлених в другій лабораторній роботі (див. додаток).

Бібліотека Sources (рис.15) містить різноманітні варіанти вхідних сигналів.
Блок Constant - подає на вхід константу заданої величини.

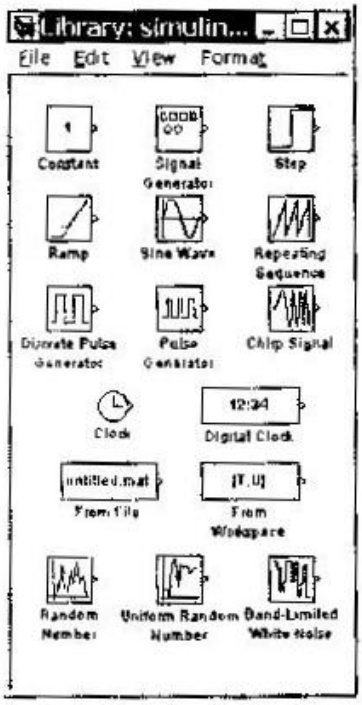

Рис.15. Бібліотека Sources

Блок Signal Generator - вихідним з блока є сигнал необхідної частоти та амплітуди, можливе задання різних форм сигнала.

Блок Step -генерує сходинку в заданної величини в заданий момент часу.

Блок Ramp - генерує послідовно зростаючий сигнал. У вікні налагоджування задаються кількість одиниць зростання з 1 сек., час початку зростання та початковий рівень зростання відповідно.

Блок Sine Wave - вихідним з блоку є сигнал сінуса заданої у вікні налагоджування амплітуди та частоти.

Блок Repeating Sequence - генерує послідовність, що повторюється. У вікні налагоджування задаються вектор моментів часу та вектор значень сигналу у відповідні моменти. Далі ця ж послідовність повторюється на протязі всього часу інтегрування.

Блок Sherp Signal - генерує сигнал зі зростаючою чатотою. У вікні налагоджування задається початкова частота, час та значення частоти у вказаний момент часу відповідно.

Блок Clock - виходом з цього блока є поточне значення часу при інтегруванні системи.

Блок Load from File - дозволяє завантажити змінну з раніше збереженої області значення парамстрів, вказавши ім'я файла та час початку послідовного завантаження змінної.

Блок From Workspace - дозволяє завантажити змінну з робочої області параметрів системи. Причому першим вказується вектор (стовбчик) моментів часу, а другим вектор (стовбчик) значень, які будуть зчитані у вказані моменти часу та час початку завантаження.

Інші блоки цієї бібліотеки дозволяють генерувати імпульсні сигнали з заданими характеристиками та сигнали генераторів шумів.

Бібліотека Sinks (рис. 16) містить блоки, відповідальні за виведення результатів інтегрування в чисельному чи графічному вигляді.

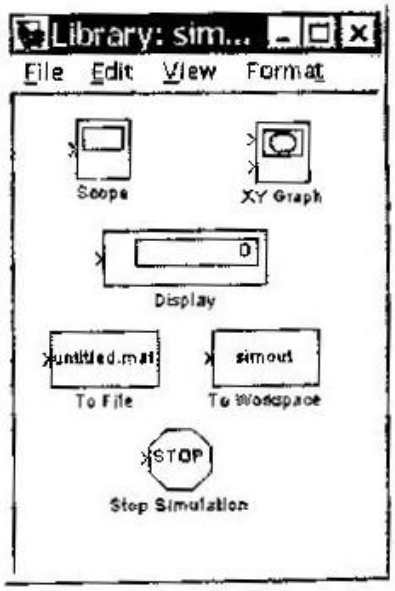

Рис.16. Бібліотека Sinks

Блок Scope - виводить залежність вхідного сигнала від часу і працює у якості осцилографа. Після або під час інтегрування подвійний клік на блоці дозволяє відкрити вікно "осцилографа". За допомогою піктограм внутрішнього вікна можна збільшити (зменшити) фрагмент сигнала, задати кількість екранів осцилографа в одному вікні, задати ім'я змінної, в яку при необхідності буде збережено вхідний сигнал осцилографа та інш.

Блок XY Graf - малює графік вхідних змінних. У вікні налагоджування вводяться мінімальні та максимальні межі відображення по осям. Також задається інтервал вибору значень з вхідного вектора. Якщо значення береться для неперервного інтегрування, воно має бути "-1".

Блок Display - виведення в чисельному вигляді значення вхідного сигнала в процесі інтегрування.

Блок To File - збереження вхідного сигнала в зовнішньому файлі. В подальшому файл можна завантажувати як робочу область.

Блок To Workspace - збереження вхідного сигнала під

вказаним іменем та у вказаному форматі в робочий простір параметрів системи.
Блок STOP - зупиняє моделювання системи в момент, коли вхідний сигнал стає відмінним від нуля.

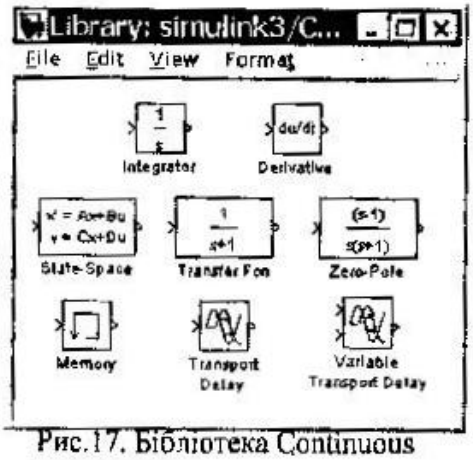

Рис.17. Бібліотека Continuous

Бібліотека Continuous (рис.17).
Блок Integrator виконує інтегрування вхідного сигналу. Початкове значення інтегрування можна вводити як через вікно налагоджування блока у рядку Initial input, так і використавши можливість зовнішнього задання початкових значень, коли initial conditions source=external.
Блок Derivate - знаходить похідну в кожний момент часу для вхідного сигналу.
Слід зауважити, що для чисельних методів інтегрування краще застосовувати блок Integrator ніж блок Derivate, тому що він працює стабільніше.
Блок Transfer Fcn - дозволяє задати перехідну функцію, використовуючи оператори Лапласа.
Блок Memory - дозволяє зробити затримку вхідного сигнала на один крок інтегрування.
Блок Transport Delay - дозволяє зробити затримку вхідного сигнала на вказаний інтервал часу.
Блок State Space - обчислює вихідний сигнал "y" по вхідному "x" за рівняннями вигляду, вказаному в блоці.

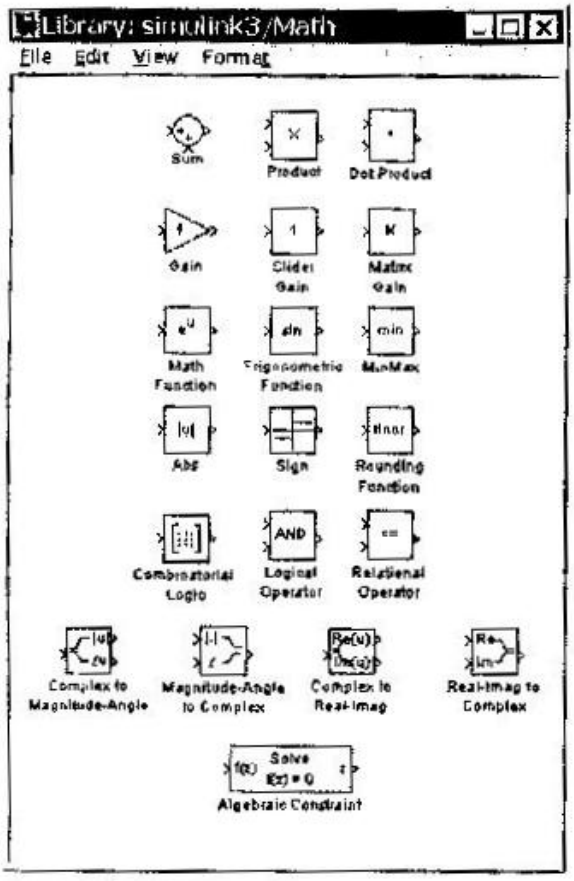

Рис.18. Бібліотека Math

Бібліотека Math (рис.18)- містить різноманітні математичні фуикції.
Розглянемо деякі з них.
Блок Sum - знаходить суму вхідних сигналів. У вікні налагоджування задається кількість входів для блока, а також послідовність знаків для кожного входу (+ або -) відповідно.
Блок Product - знаходить добуток вхідних сигналів. У вікні налагоджування також можна задати кількість входів для блоку.
Блок Gain – виконує множення вхідного сигналу на вказану у блоці константу.
Блок Slider Gain - множення вхідного сигналу на константу, де константа задається у вікні з бігунком.
Блок Math Function - дозволяє застосувати до вхідного сигналу обрану з бібліотеки математичну функцію.
Блок Trigonometric Function - дозволяє застосувати до вхідного сигналу обрану з бібліотеки тригонометричну функцію.
Блок MinMax - обирає і утримує в продовж інтегрування мінімальне (максимальне) значення на інтервалі.

Блок Abs - знаходить модуль вхідного сигналу.
Блок Sign - приймає значення +1 або -1 в

залежності від знаку вхідного сигналу.

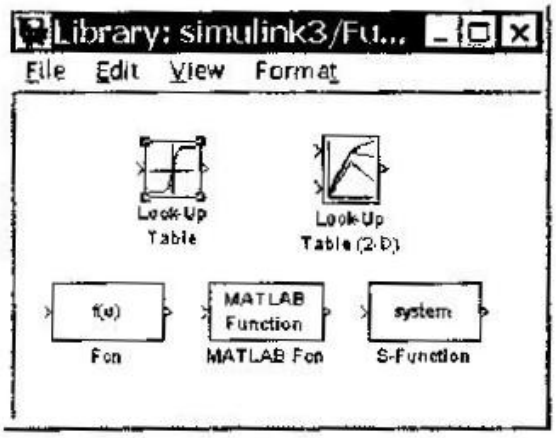

Рис.19. Бібліотека Function and Tables

Блок Relational Operator - дозволяє порівнювати вхідні сигнали між собою. Виходом є 1 (істина) чи 0 (не істина) в залежності від результату. Перший сигнал стає зліва від оператора відношення, другий - справа.

Бібліотека Function and Tables (рис.19)- дозволяє задавати для вхідного сигналу функції перетворення та таблиці.

Блок Fcn - дозволяє для вхідного сигналу використовувати математичні вирази. Вхідний сигнал у виразі позначається через змінну "u".

Блок Matlab Fcn - дозволяє для вхідного сигналу використовувати функції - оброблювачі, в тому числі написані користувачем. Функція повинна повертати вектор, який і сприймається у якості вихідного сигналу блока.

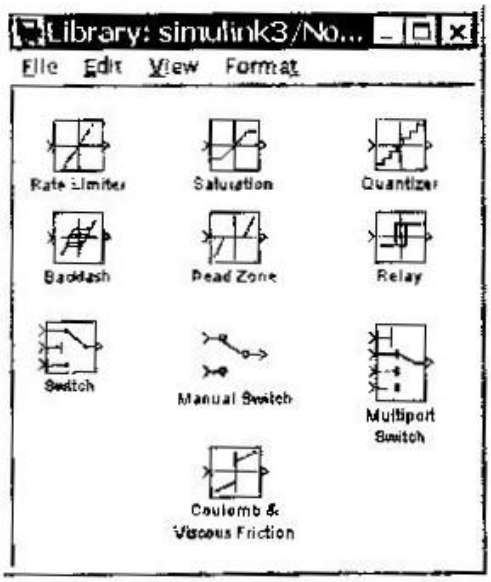

Рис.20. Бібліотека Nonlinear

Бібліотека Nonlinear (рис.20)- задає різні типи нелінійностей.

Блок Saturation - нелінійність "Щілина".

Блок Dead Zone - нелінійність "Мертва зона".

Бібліотека також містить різні типи перемикачів, як за умовою, так і вручну.

Бібліотека Signals&Systems (рис.21) містить блоки для роботи з підсистемами. У випадку, коли модель системи громіздка, її можна розділити на підсистеми, задавши необхідні входи та виходи блоками *In Out*, тим самим зв'язавши підсистему з загальною моделлю.

Для створення підсистеми обирається група блоків, які в неї ввійдуть, після чого у вікні моделі командою Edit /Create Subsystem вона створюється. Обрані блоки заміняються блоком підсистеми, в яку можна зайти подвійним кліком: відкривається окреме вікно для редагування підсистеми.

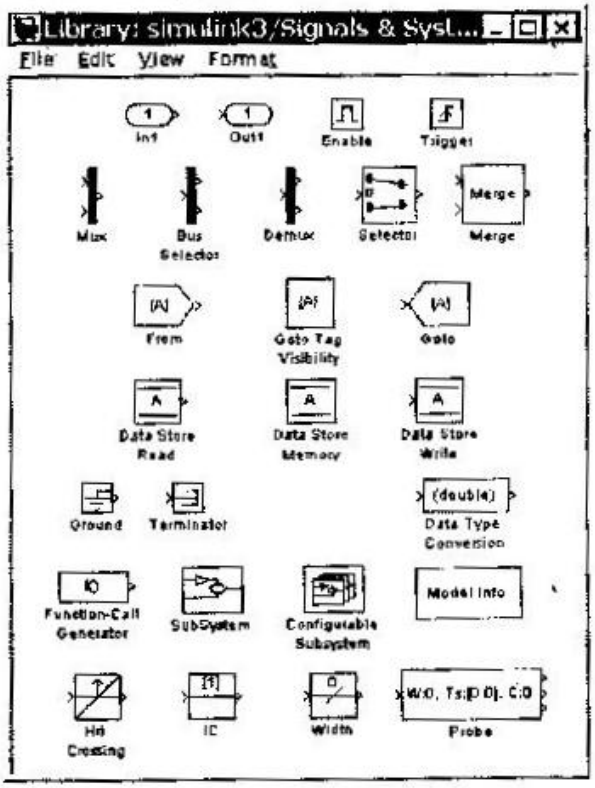

Рис.21. Бібліотека Signals&Systems

Блоки Enable та Trigger - дозволяють керувати моментами підключення системи в момент подання керуючого сигнала.
Блоки Mux, Bus, Selector Demux - є шинами.
В цій же бібліотеці присутні блоки безумовних переходів, інформації про поточну модель, заземлення та інш.

17. *Приклад реалізації побудови моделі*

Розглянемо приклад побудови Simulink-моделі для диференціального рівняння Van der Pol-a, яке, відповідно, задається у вигляді

$$\frac{d^2x}{dt} = \frac{dx}{dt}(1-x^2) - x.$$

Відомо, що рівняння другого порядку можна перезадати і як систему з двох рівнянь першого порядку, а саме

$$\frac{dx(1)}{dt} = x(1)(1-x(2)^2) - x(2)$$

$$\frac{dx(2)}{dt} = x(1)$$

де було виконано заміну $x' = x(2), \frac{dx}{dt} = x(1)$. Саме цей приклад розглянуто в deedemo1 метода dee.

Оскільки у нас система другого порядку, для побудови Simulink-моделі почнемо із задання двох інтеграторів. Як вказувалося вище, для чисельних методів інтегрування краще використовувати саме блок інтегрування замість блоку похідної, оскільки він працює стабільніше (рис. 22).

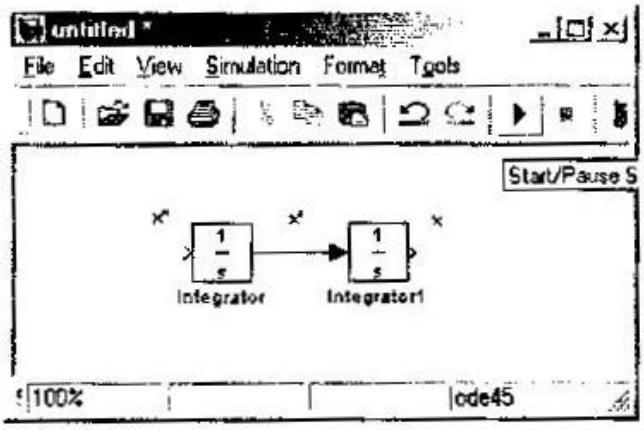

Рис.22. Початок створення моделі

Нанесення коментарів, в даному випадку це "x,x',x''", виконується подвійним кліком на пустому полі вікна моделі, після чого вводиться необхідний текст. На вхід першого інтегратора подасться вхідний сигнал x''. Після інтегрування матимемо x'. Після повторного інтегрування матимемо x. Далі виконуємо обчислення виразу $(1-x^2)$. Для цього використаємо блок Fcn і у вікні налагоджування задаємо вираз 1-u*u. На вхід цього блоку подаємо значення x. Далі необхідно перемножити вихідний сигнал блоку Fcn з x', тобто отримати вираз $x'(1-x^2)$. Обираємо блок Product , на один вхід якого подаємо вихідний сигнал блоку Fcn, на другий x. Для виконання під'єднання проводимо лінію із входу блока Product до лінії x' в момент, коли курсор зміниться на здвоєне перехрестя, відпускаємо кнопку миші і з'являється точка під'єднання (рис.23).

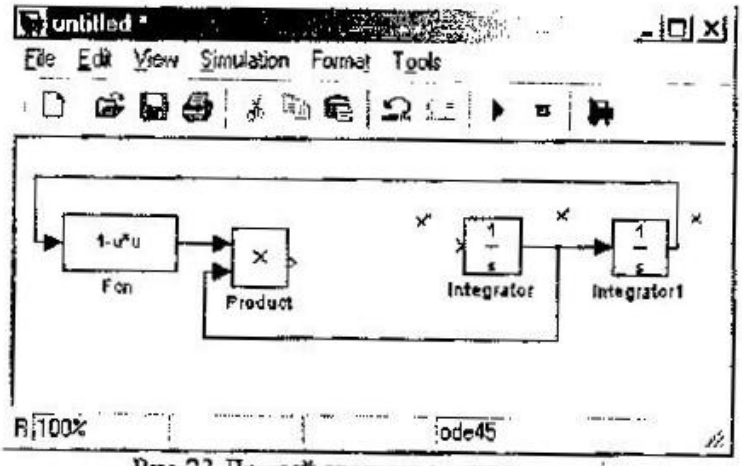

Рис.23 Другий етап створення моделі

Далі залишилося від отриманого виразу відняти значення x. Для цього обираємо блок Sum, у вікні налагоджування якого ставимо знаки "-+". Після цього на вхід зі знаком "+" подаємо вихідний сигнал з блоку Product, а на знак "–" подаємо x. Результатом виразу є x'', тобто вихід з блоку Sum подасться на вхід першого інтегратора (рис.24).

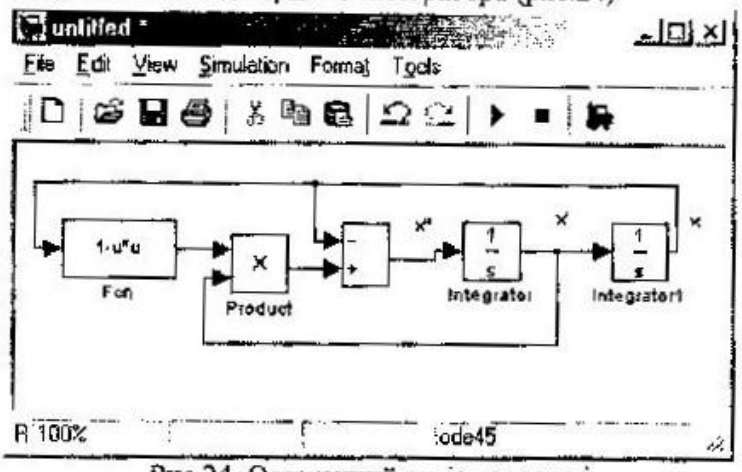

Рис.24. Остаточний варіант моделі

Система зібрана. Залишилося задати початкові умови інтегрування, обрати метод інтегрування та виконати необхідну візуалізацію.

Нехай ми хочемо на осцилографі спостерігати одразу кілька вихідних сигналів розв'язку системи x та x'. Для цього обираємо блок графічного виводу Scope, але перед ним ставимо блок Mux, на вхід якого і подаємо сигнали x та x'. Вихід блока Mux з'єднуємо зі входом блока Scope (Рис.25).

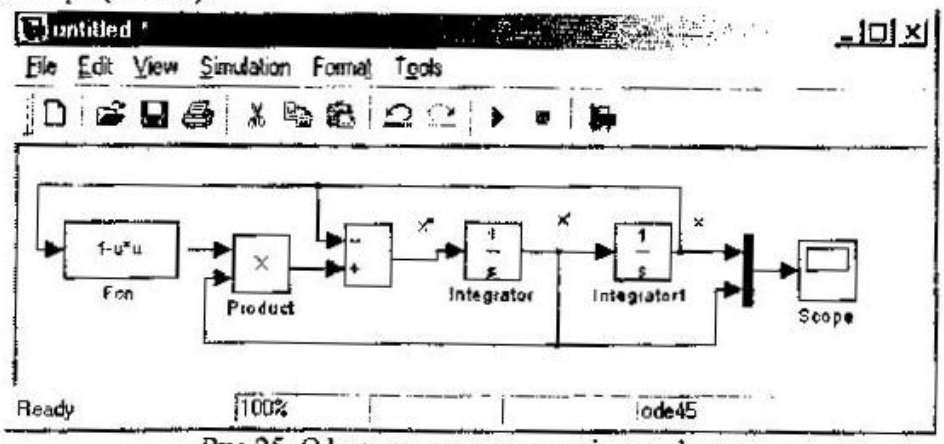

Рис.25. Оформлення результатів розв'язку

Задаємо початкові значення інтегрування. У вікні налагоджування інтеграторів у рядку Initial Conditions задаємо необхідні початкові значення. Для блоку Integrator, наприклад, 0, а для блоку Integrator1, наприклад, 1.

Визначаємося з часом та методом інтегрування. Обираємо меню Simulations/Parameters, і задаємо час інтегрування від 0 до 50, крок інтегрування 0.1, метод інтегрування Runge-Kutta (рис.26) і запускаємо систему на виконання.

Задача виконана.

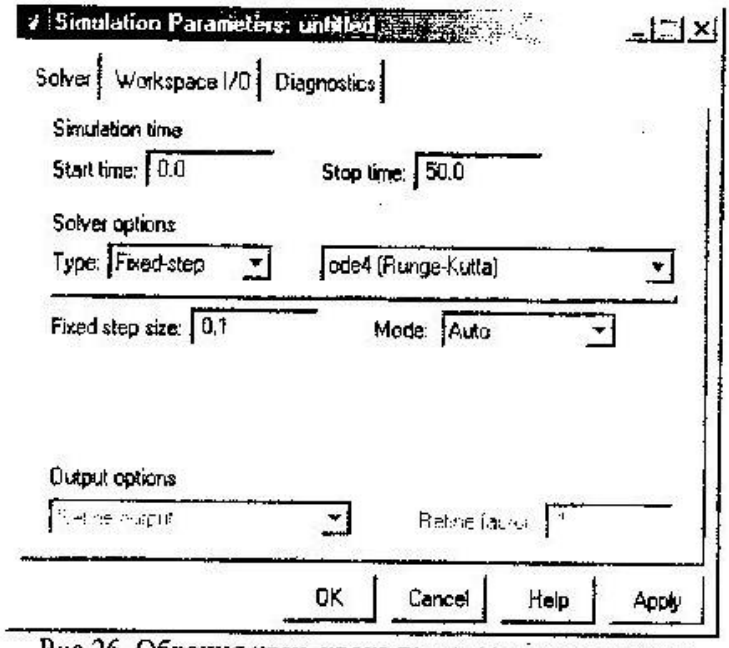

Рис.26. Обрання часу, крока та метода інтегрування

Результатом буде виведення графіка на осцилограф (рис.27):

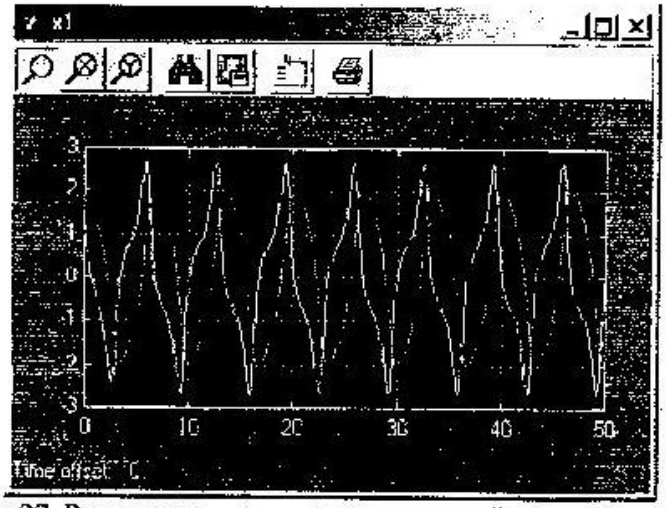

Рис.27. Результат моделювання, виведений на осцилограф