Національний технічний університет України
«Київський політехнічний інститут»
Кафедра приладів і систем орієнтації і навігації

Ю. Ф. Лазарєв

Довідник з MATLAB

Рекомендовано Вченою Радою
приладобудівного факультету НТУУ «КПІ»
як електронний навчальний посібник
з курсового і дипломного проектування

Київ – НТУУ "КПІ" – 2013

УДК 681.3(0.75)
Л17

Відповідальний редактор
к. т. н., доцент Бондар Павло Михайлович

Лазарєв Ю. Ф.
Л17 Довідник з MATLAB / Електронний навчальний посібник з курсового і дипломного проектування. – К.: НТУУ "КПІ", 2013. – 132 с.

ПЕРЕДМОВА

Навчальний посібник призначено для україномовних студентів вищих технічних навчальних закладів. В ньому наводяться відомості, що дозволяють студенту при виконанні курсових і дипломного проектів вільно використовувати переваги комп'ютерної системи Matlab для утворення програмних моделей технічних систем і комп'ютерного моделювання їх поводження у заданих умовах експлуатації.

Складається з пяти розділів: "Командне вікно", "Операції з числами", "Операції з векторами", "Операції з матрицями" і "Пакет програм візуального програмування SIMULINK", які у стислому вигляді знайомлять читача зі змістом основних операторів, функцій і процедур Matlab, правилами їх використання.

1. КОМАНДНЕ ВІКНО MATLAB

Після виклику Matlab із середовища Windows на екрані виникає вікно, зображене на рис. 1.1.

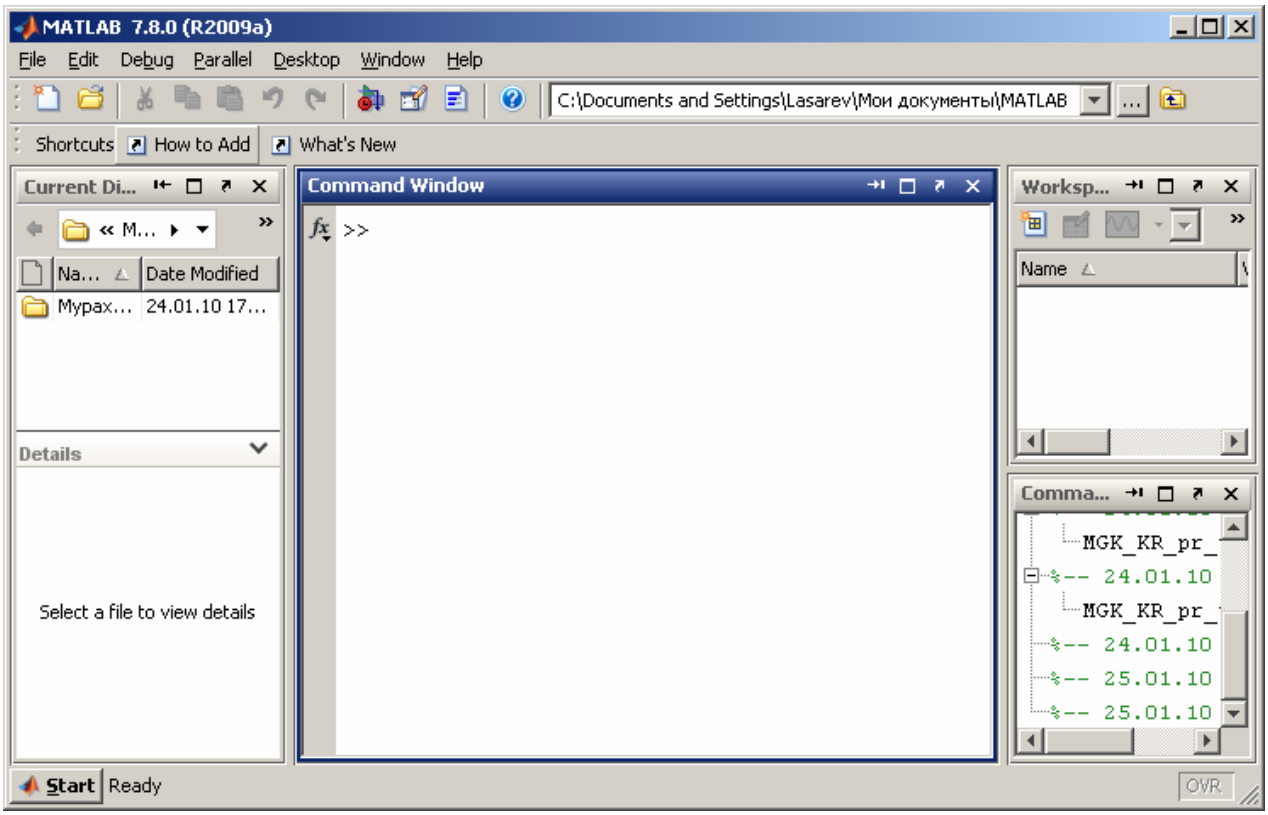


Рис. 1.1. Вигляд повного вікна Matlab

Якщо закрити усі бокові допоміжні підвікна, залишиться одне вікно, яке називають "командним" (Command Window) середовища Matlab (рис. 1.2).

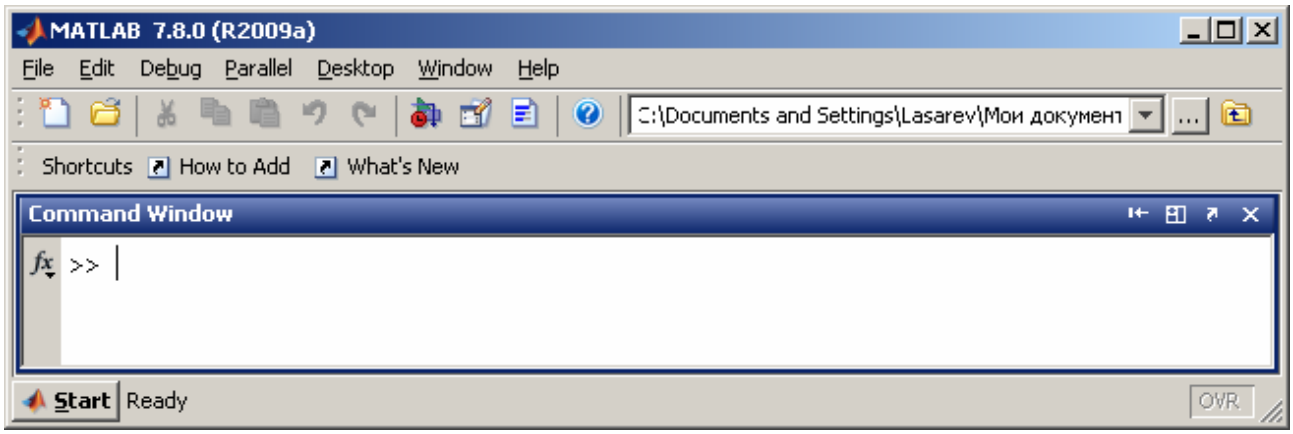


Рис. 1.2. Вигляд командного вікна Matlab

Це вікно є головним у Matlab. У ньому виникають символи команд, що набираються користувачем на клавіатурі дисплея, відображуються результати виконання цих команд, текст програми, яка виконується, і інформація про помилки виконання програми, розпізнані системою .

Ознакою того, що Matlab готова до сприйняття і виконання чергової команди, є поява в останньому рядку текстового поля командного вікна знака запрошення (>>), праворуч якого миготить вертикальна риса.

У верхній частині вікна (під заголовком) розміщений рядок меню, в якому містяться меню ***File***, ***Edit***, ***Debug, Parallel, Desktop, Windows***, ***Help***. Щоб відчинити якесь меню, потрібно встановити на ньому курсор миші і клацнути її лівою кнопкою.

Тут відзначимо лише, що *для виходу із середовища Matlab* достатньо відчинити меню ***File*** і обрати у ньому команду ***Exit MATLAB*****,** або просто зачинити командне вікно, клацнувши лівою клавішою миші, коли курсор миші встановлений на зображенні верхньої крайньої правої кнопки цього вікна (з позначенням хрестика)**.**

2. ОПЕРАЦІЇ З ЧИСЛАМИ

2.1. Введення дійсних чисел

Введення чисел із клавіатури здійснюється по загальних правилах, прийнятих для мов програмування високого рівня:

для відділення дробової частини мантиси числа застосовується десяткова крапка (замість коми при звичайному записі);

десятковий показник числа записується у вигляді цілого числа після попереднього запису символу "e";

між записом мантиси числа й символом "e" (який відокремлює мантису від показника) не повинно бути ніяких символів, включаючи і символ пропуску.

Якщо, наприклад, ввести в командному вікні Matlab рядок

120357.9245e-78,

то після натискання клавіші <Enter> у цьому вікні виникне запис (рис. 2.1):

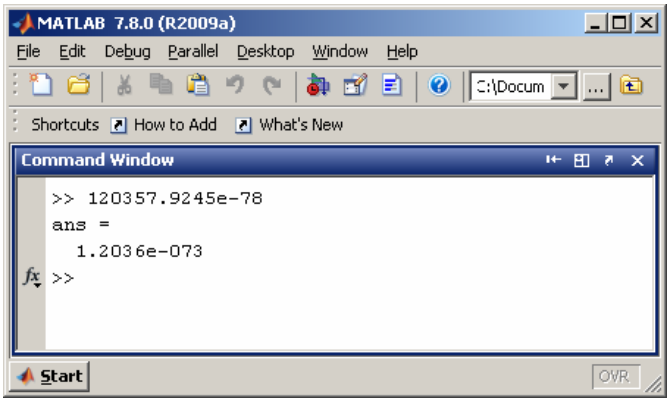


Рис. 2.1. Введення і відображування чисел

Слід зазначити, що результат виводиться у виді (форматі), що визначається попередньо встановленим форматом подання чисел. Цей формат може бути встановлений за допомогою команди ***Preferences*** меню ***File*** (рис. 2.2).

Після її виклику на екрані з'явиться однойменне вікно (рис. 2.3). У поділі ***Command Window*** одна з ділянок цього вікна має назву ***Numeric Format***. Її призначено для встановлення і змінювання формату подання чисел, які виводяться в командне вікно в процесі розрахунків.

Передбачені такі формати:

Short (default) - стислий запис (застосовується за умовчанням);
Long - довгий запис;
Hex - запис у виді шістнадцяткового числа;
Bank - запис до сотих часток;
Plus - записується тільки знак числа;
Short E - стислий запис у форматі із плаваючою комою;
Long E - довгий запис у форматі із плаваючою комою;
Short G - друга форма стислого запису у форматі з плаваючою комою;
Long G - друга форма довгого запису у форматі з плаваючою комою;

Rational - запис у вигляді раціонального дробу.

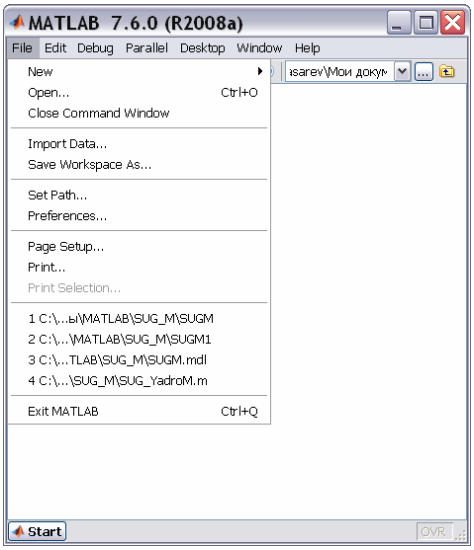


Рис. 2.2. Меню "File"

Обираючи за допомогою мишки потрібний вид подання чисел, можна забезпечити надалі виведення чисел у командне вікно саме в цій формі.

Як видно з рис. 2.1, число, яке виведено на екран, не збігається з введеним. Це обумовлено тим, що встановлений за замовчуванням формат подання чисел (Short G) не дозволяє вивести більше 5 значущих цифр числа. Насправді введене число усередині Matlab зберігається з усіма введеними його цифрами. Наприклад, якщо обрати мишкою селекторну кнопку **Long E** (тобто установити цей формат подання чисел), то, повторюючи ті ж дії, отримаємо результат, відображений на рис. 2.4, де вже всі цифри відображені вірно.

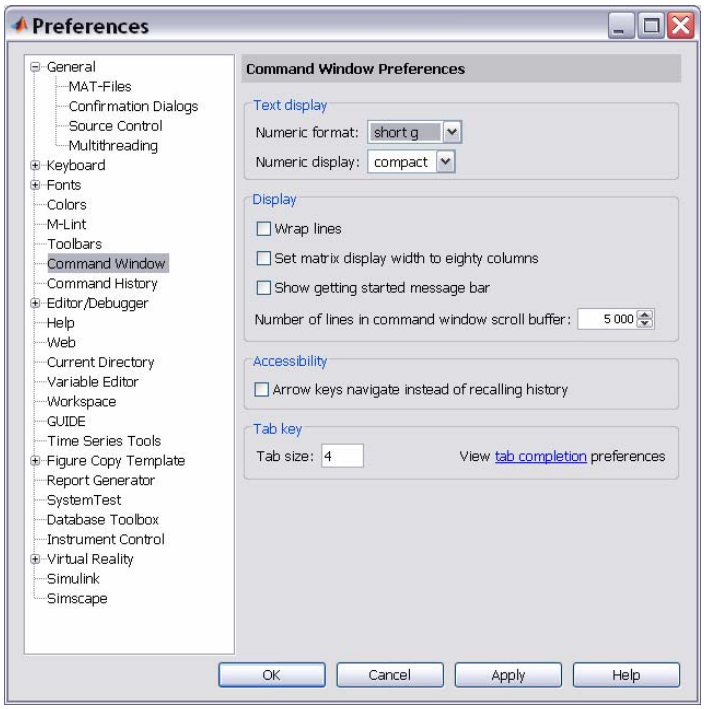


Рис. 2.3. Вікно Preferences меню "File"

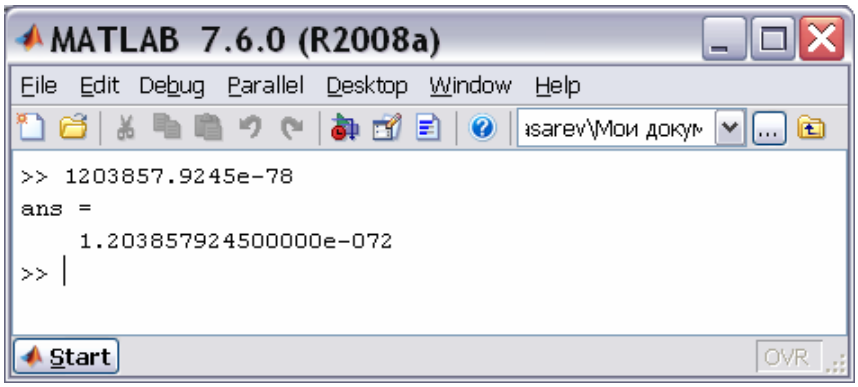


Рис. 2.4. Виведення числа у форматі Long E

Слід пам’ятати:

- уведене число й результати всіх обчислень у системі Matlab зберігаються в пам'яті ПК із відносною похибкою біля 1.10^{-16} (тобто з точними значеннями в 15 десяткових розрядах);

- діапазон подання модуля дійсних чисел лежить у проміжку між 10^{-308} *і* 10^{+308}.

2.2. Найпростіші арифметичні дії

В арифметичних виразах мови Matlab використовуються такі знаки арифметичних операцій:

\+ – додавання;
– – віднімання;
* – множення;
/ – ділення зліва праворуч;
\ – ділення справа ліворуч;
^ - –піднесення до степеня.

Використання Matlab у режимі калькулятора може відбуватися шляхом простого запису в командний рядок послідовності арифметичних дій з числами, тобто звичайного арифметичного виразу, наприклад:

(4.5)^2*7.23 – 3.14*10.4.

Якщо після введення із клавіатури цієї послідовності натиснути клавішу <Enter>, у командному вікні виникне результат виконання у виді, поданому на рис. 2.5, тобто на екран під ім’ям системної змінної ***ans*** виводиться результат дії останнього виконаного оператора.

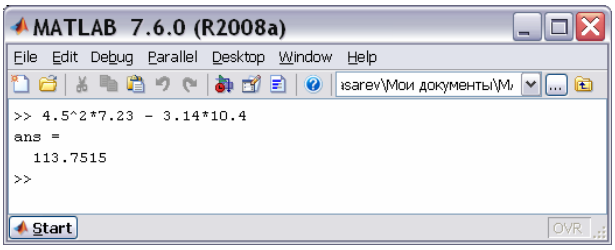


Рис. 2.5. Введення арифметичного виразу і відображення результату

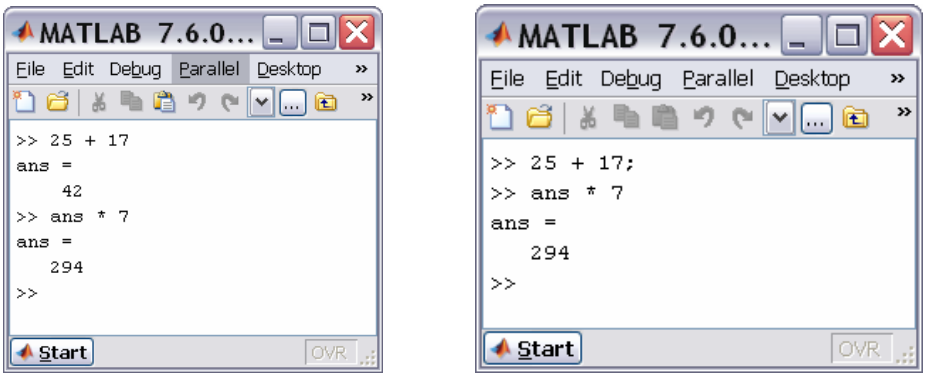
Взагалі виведення проміжної інформації у командне вікно підпорядковується таким правилам:

- *якщо запис оператора не закінчується символом ';', результат дії цього оператора одразу ж виводиться в командне вікно;*
- *якщо оператор закінчується символом ';', результат його дії не відображується в командному вікні;*
- *якщо оператор не містить знака присвоювання (=), тобто є просто записом деякої послідовності дій над числами і змінними, значення результату присвоюється спеціальній системній змінній за ім'ям* ***ans****;*
- *отримане значення змінної* ***ans*** *можна використовувати в наступних операторах обчислень, використовуючи це ім'я* ***ans****; при цьому варто пам'ятати, що значення системної змінної* ***ans*** *змінюється після дії чергового оператора без знака присвоювання;*
- *у загальному випадку форма подання результату в командне вікно має вид:*

<Ім'я змінної> = <результат>.

Приклад.

Нехай потрібно обчислити вираз (25+17)*7. Це можна зробити таким чином. Спочатку набираємо послідовність **25+17** і натискаємо <Enter>. Одержуємо на екрані результат у виді **ans = 42**. Тепер записуємо послідовність **ans*7** і натискаємо <Enter>. Одержуємо **ans = 294** (рис. 2.6а). Щоб запобігти виведення проміжного результату дії 25+17, достатньо після запису цієї послідовності додати символ ' ; '. Тоді будемо мати результати у виді, поданому на рис. 2.6б.



а) *б)*

Рис. 2.6. Використання системної змінної ***ans***

Особливістю Matlab як калькулятора є можливість використання імен змінних для запису проміжних результатів у пам'ять ПК. Для цього застосовується операція присвоювання, що вводиться знаком рівності ' = ' у відповідності зі схемою :

<Ім'я змінної> = <вираз>[;]

Ім'я змінної може містити до 30 символів і повинно не збігатися з іменами функцій, процедур системи і системних змінних. При цьому система розрізнює великі й малі букви в змінних. Так, імена 'amenu' , 'Amenu', 'aMenu' у Matlab позначають різні змінні.

Вираз праворуч від знака присвоювання може бути просто числом, арифметичним виразом, рядком символів (тоді ці символи потрібно укласти в апострофи) або символьним виразом. Якщо вираз не закінчується символом ' ; ', після натискання клавіші <Enter> у командному вікні виникне результат виконання у виді :

<Ім'я змінної> = <результат>.

Наприклад, якщо ввести в командне вікно рядок 'x = 25 + 17', то на екрані виникне запис (рис. 2.7) :

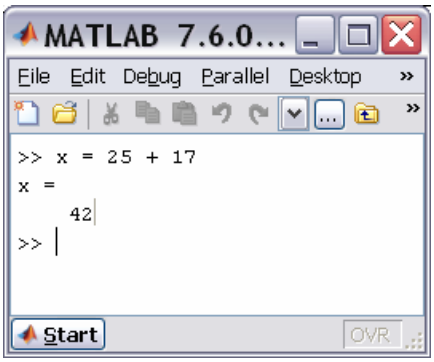


Рис. 2.7. Присвоювання значення змінній

Система Matlab має кілька імен змінних, що використовуються самою системою і входять до складу зарезервованих:

i, ***j*** – уявна одиниця (корінь квадратний з -1);

pi – число π (зберігається у виді 3.141592653589793);

inf – позначення машинної нескінченності;

NaN – позначення невизначеного результату (наприклад, типу 0/0 або inf/inf);

eps – похибка операцій над числами із плаваючою комою

ans – результат останньої операції без знака присвоювання;

realmax – максимальна величина числа, що може бути використана;

realmin – мінімальна величина числа, що може бути використана.

Ці змінні можна використовувати в математичних виразах.

2.3. Введення комплексних чисел

Мова системи Matlab, на відміну від багатьох мов програмування високого рівня, містить у собі дуже просту в користуванні вбудовану арифметику комплексних чисел. Більшість елементарних математичних функцій побудовано у такий спосіб, що аргументи припускаються комплексними числами, а результати також формуються як комплексні числа. Ця особливість мови робить її дуже привабливою й корисною для інженерів і науковців.

Для позначення уявної одиниці в мові Matlab зарезервовано два ймення ***i*** і ***j***. Уведення із клавіатури значення комплексного числа здійснюється шляхом запису в командне вікно рядка виду:

*<ім'я комплексної змінної> = <значення ДЧ> + **i** [j] *<значення УЧ>,*

де ДЧ – дійсна частина комплексного числа, УЧ – уявна частина. Приклад наведено на рис. 2.8:

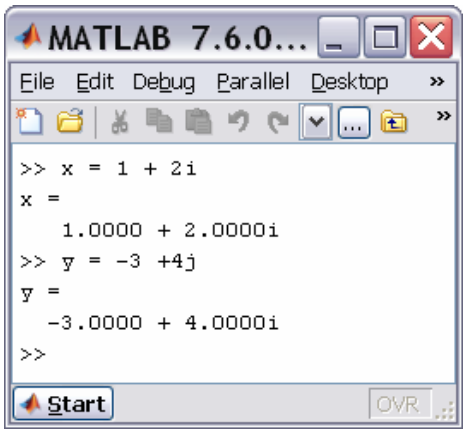


Рис. 2.8. Введення комплексних чисел і виведення результату

З нього видно, у якому виді система виводить комплексні числа на екран (і "до друку").

2.4. Елементарні математичні функції

Загальна форма використання функції у Matlab така:

<ім'я результату> = <ім'я функції>(<перелік аргументів або їх значень>).

У мові Matlab передбачені наступні елементарні арифметичні функції.

Тригонометричні й гіперболічні функції

sin(Z) – синус числа Z;
sinh(Z) – гіперболічний синус;
asin(Z) – арксинус (у радіанах, у діапазоні від $-\pi/2$ до $+\pi/2$);
asinh(Z) – обернений гіперболічний синус;
cos(Z) – косинус;
cosh(Z) – гіперболічний косинус;
acos(Z) – арккосинус (у діапазоні від 0 до π);
acosh(Z) – обернений гіперболічний косинус;
tan(Z) – тангенс;
tanh(Z) – гіперболічний тангенс;
atan(Z) – арктангенс (у діапазоні від $-\pi/2$ до $+\pi/2$);
atan2(X,Y) – чотириквадрантний арктангенс (кут у діапазоні від $-\pi$ до $+\pi$ між горизонтальним правим променем і променем, що проходить через точку з координатами X і Y);
atanh(Z) – обернений гіперболічний тангенс;
sec(Z) – секанс;
sech(Z) – гіперболічний секанс;

asec(Z) – арксеканс;
asech(Z) – обернений гіперболічний секанс;
csc(Z) – косеканс;
csch(Z) – гіперболічний косеканс;
acsc(Z) – арккосеканс;
acsch(Z) – обернений гіперболічний косеканс;
cot(Z) – котангенс;
coth(Z) – гіперболічний котангенс;
acot(Z) – арккотангенс;
acoth(Z) – обернений гіперболічний котангенс.

Експоненціальні функції

exp(Z) – експонента числа Z;
log(Z) – натуральний логарифм;
log10(Z) – десятковий логарифм;
sqrt(Z) – квадратний корінь із числа Z;
abs(Z) – модуль числа Z.

Цілочислові функції

fix(Z) – округлення до найближчого цілого убік нуля;
floor(Z) – округлення до найближчого цілого убік від'ємної не скінченності;
ceil(Z) – округлення до найближчого цілого убік додатної не скінченності;
round(Z) – звичайне округлення числа Z до найближчого цілого;
mod(X,Y) – цілочислове ділення ділення X на Y;
rem(X,Y) – обчислення остачі від ділення X на Y;
sign(Z) – обчислення сигнум-функції числа Z (0 при Z=0, -1 при Z<0, 1 при Z>0).

2.4. Спеціальні математичні функції

Крім елементарних у мові Matlab передбачено цілу низку спеціальних математичних функцій. Нижче наведено перелік і стислий зміст цих функцій. Правила звернення до них і використання користувач може відшукати в описах цих функцій, що виводяться на екран, якщо набрати команду ***help*** і вказати в тому ж рядку ім'я функції.

Функції перетворення координат

cart2sph – перетворення декартових координат у сферичні;
cart2pol – перетворення декартових координат у полярні;
pol2cart – перетворення полярних координат у декартові;
sph2cart – перетворення сферичних координат у декартові.

Функції Бесселя

besselj – функція Бесселя першого роду;
bessely – функція Бесселя другого роду;
besseli – модифікована функція Бесселя першого роду;
besselk – модифікована функція Бесселя другого роду.

Бета-функції

beta – бета-функція;
betainc – неповна бета-функція;
betaln – логарифм бета-функції.

Гамма-функції

gamma – гамма-функція;
gammainc – неповна гамма-функція;
gammaln – логарифм гамма-функції.

Еліптичні функції й інтеграли

ellipj – еліптичні функції Якобі;
ellipke – повний еліптичний інтеграл;
expint – функція експоненціального інтегралу.

Функції похибок

erf – функція похибок;
erfc – додаткова функція похибок;
erfcx – масштабована додаткова функція похибок;
erfinv – обернена функція похибок.

Інші функції

gcd – найбільший загальний дільник;
lcm – найменше загальне кратне;
legendre – узагальнена функція Лежандра;
log2 – логарифм за основою 2;
pow2 – піднесення 2 до зазначеного степеня;
rat – подання числа у виді раціонального дробу;
rats – подання чисел у виді раціонального дробу.

2.5. Елементарні дії з комплексними числами

Найпростіші дії з комплексними числами – додавання, віднімання, множення, ділення й піднесення до степеня – здійснюються за допомогою звичайних арифметичних знаків +, –, *, /, \ і ^ відповідно.

Приклади використання наведені на рис. 2.9

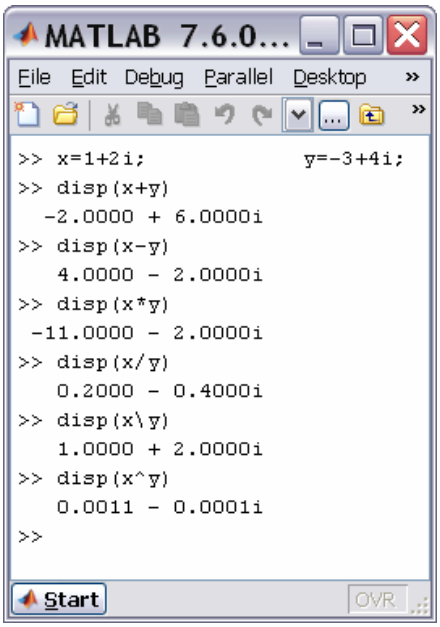


Рис. 2.9. Приклади використання функції ***disp***

Примітка. У наведеному фрагменті використана функція ***disp*** (від слова 'дисплей'), яка також дозволяє виводити в командне вікно результати обчислень або деякий текст. При цьому чисельний результат, як очевидно, виводиться вже без указівки йменння змінної або ***ans***.

2.6. Функції комплексного аргументу

Практично *всі* елементарні *математичні функції*, наведені раніше, *обчислюються за комплексних значень аргументу* й одержують у результаті цього комплексні значення результату.

Завдяки цьому, наприклад, функція ***sqrt*** обчислює, на відміну від інших мов програмування, квадратний корінь із від'ємного аргументу, а функція ***abs*** при комплексному значенні аргументу обчислює модуль комплексного числа. Приклади наведені на рис. 2.10.

У Matlab є кілька додаткових функцій, розрахованих тільки на комплексний аргумент:

real(Z) – виділяє дійсну частину комплексного аргументу Z;
imag(Z) – виділяє уявну частину комплексного аргументу;
angle(Z) – обчислює значення аргументу комплексного числа Z (у радіанах від $-\pi$ до $+\pi$);
conj(Z) – видає число, комплексно спряжене щодо Z.

Приклади наведені на рис. 2.11.

Крім того, у Matlab є спеціальна функція ***cplxpair***(V), що здійснює сортування заданого вектора V із комплексними елементами у такий спосіб, що комплексно-спряжені пари цих елементів розташовуються у вихідному векторі в порядку зростання їхніх дійсних частин, при цьому елемент із від'ємною уявною частиною завжди розташовується першим. Дійсні елементи завершують комплексно-спряжені пари.

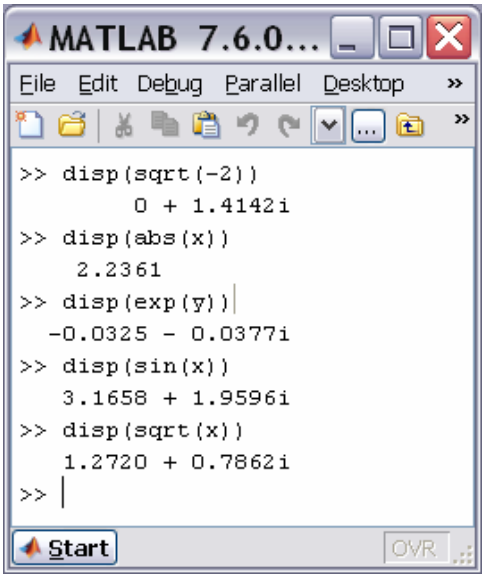


Рис. 2.10. Застосування функцій з комплексним аргументом

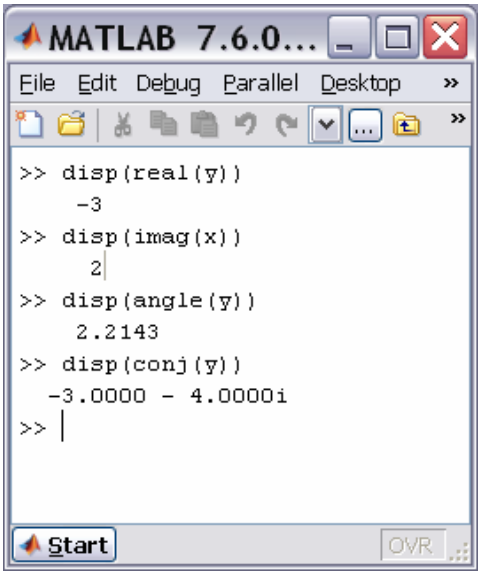


Рис. 2.11. Застосування функцій комплексного аргументу

Наприклад (*надалі в прикладах команди, що набираються із клавіатури, будуть написані масним шрифтом, а результат їхнього виконання – звичайним шрифтом*):

```
» v = [ -1, -1+2i,-5,4,5i,-1-2i,-5i]
v =
  Columns 1 through 4
  -1.0000          -1.0000 + 2.0000i                  -5.0000                   4.0000
  Columns 5 through 7
       0 + 5. 0000i          -1. 0000 – 2. 0000i       0 – 5. 0000i
» disp(cplxpair(v))
  Columns 1 through 4
  -1. 0000 – 2. 0000i  -1. 0000 + 2. 0000i       0 – 5. 0000i       0 + 5. 0000i
  Columns 5 through 7
  -5. 0000          -1. 0000          4. 0000
```

Пристосованість більшості функцій Matlab до оперування з комплексними числами дозволяє значно простіше будувати обчислення з дійсними числами, результат яких є комплексним, наприклад, знаходити комплексні корені квадратних рівнянь.

2.7. Знайомство з програмуванням в Matlab

Робота в режимі калькулятора в середовищі Matlab, незважаючи на досить значні можливості, має істотні незручності. Неможливо повторити всі попередні обчислення й дії при нових значеннях початкових даних без повторного набирання всіх попередніх операторів. Не можна повернутися назад і повторити деякі дії, або за деякою умовою перейти до виконання іншої послідовності операторів. І взагалі, якщо кількість операторів є значною, стає проблемою налагодити правильну їхню роботу через неминучі помилки при набиранні команд. Тому складні, із перериваннями, складними переходами по певних умовах, із часто повторюваними однотипними діями обчислення, які, до

того ж, необхідно проводити неодноразово при змінених первинних даних, потребують їхнього спеціального оформлення у виді записаних на диску файлів, тобто у виді програм. Перевага програм у тому, що, унаслідок того, що вони зафіксовані у виді записаних файлів, стає можливим багаторазове звернення до тих самих операторів і до програми в цілому. Це дозволяє спростити процес налагоджування програми, зробити процес обчислень більш наочним і прозорим, а завдячуючи цьому – різко зменшити можливість появи принципових помилок при розробці програм. Крім того, у програмах виникає можливість автоматизувати також і процес змінювання значень первісних параметрів у діалоговому режимі.

Створення програми в середовищі Matlab здійснюється за допомогою вбудованого редактора. Вікно цього вбудованого редактора виникає на екрані, якщо перед цим використано команду "M-file" із поділу *New* або обрано назву одного з існуючих М-файлів при виклику команди *Open M-file* із меню ***File*** командного вікна. У першому випадку вікно текстового редактора є порожнім (рис. 2.12), у другому – у ньому міститься текст викликаного М-файлу. В обох випадках вікно текстового редактора готове для введення нового тексту або коригування існуючого.

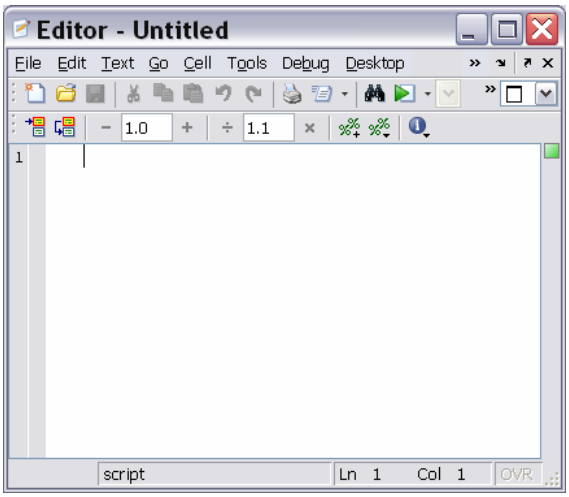


Рис. 2.12. Вікно вбудованого текстового редактора Matlab

Запис тексту програми (М-файлу) мовою Matlab має підпорядковуватися таким правилам.

1. Зазвичай кожний оператор записується в окремому рядку тексту програми. Ознакою кінця оператора є символ (він не виникає у вікні) повернення каретки й переходу на наступний рядок, який вводиться в програму при натисканні клавіші <Enter>, тобто при переході при записі тексту програми на наступний рядок.

2. Можна розміщувати кілька операторів в одному рядку. Тоді попередній оператор у тому ж рядку має закінчуватися символом ' ; ' або ' , '.

3. Можна довгий оператор записувати в декілька рядків. При цьому попередній рядок оператора має завершуватися трьома крапками (' ... ').

4. Якщо черговий оператор не закінчується символом ' ; ', результат його дії при виконанні програми буде виведений у командне вікно. Щоб запобігти виведенню на екран результатів дії оператора програми, запис цього оператора в тексті програми має закінчуватися символом ' ; '.

5. Рядок програми, що починається із символу ' % ', не виконується. Цей рядок сприймається системою Matlab як *коментар*. Тому для введення коментарю в будь-яке місце тексту програми достатньо почати відповідний рядок із символу ' % '.

6. Рядки коментарю, які передують першому виконуваному (тобто такому, що не є коментарем) оператору програми, сприймаються системою Matlab як опис програми. Саме ці рядки виводяться в командне вікно, якщо в ньому набрано команду

help <ім'я файла>

7. У програмах мовою Matlab *відсутній символ закінчення тексту програми.*

8. У мові Matlab *змінні не описуються і не оголошуються*. Будь-яке нове ім'я, що зустрічається в тексті програми при її виконанні, сприймається системою Matlab як ім’я матриці. Розмір цієї матриці встановлюється при введенні значень її елементів або визначається діями по встановленню значень її елементів, описаними у попередньому операторі або процедурі. Ця особливість робить мову Matlab дуже простою у вжитку і привабливою. У мові MatLAB неможливо використання вхідної матриці або змінної, у якій попередньо не введені або обчислені значення її елементів (а значить – і визначені розміри цієї матриці). У противному випадку при виконанні програми Matlab виникне повідомлення про помилку – "Змінна не визначена".

9. Імена змінних можуть містити лише букви латинського алфавіту або цифри і мають починатися з букви. Загальна кількість символів в імені може сягати 30. В іменах змінних можуть використовуватися як великі, так і малі букви. Особливістю мови Matlab є те, що *великі й малі букви в іменах розрізнюються системою*. Наприклад, символи "a" і "A" можуть використовуватися в одній програмі для позначення різних величин.

3. ОПЕРАЦІЇ З ВЕКТОРАМИ

Matlab є системою, спеціально призначеною для здійснювання складних обчислень із векторами, матрицями й поліномами.

Під вектором у Matlab розуміється одновимірний масив чисел, а під матрицею – двовимірний масив. При цьому за замовчуванням припускається, що будь-яка задана змінна є вектором або матрицею. Наприклад, окреме задане число система сприймає як матрицю розміром (1*1), а вектор-рядок із N елементами – як матрицю розміром (1*N).

3.1. Введення векторів і матриць

Початкові значення векторів можна задавати із клавіатури шляхом поелементного введення. Для цього в рядку треба спочатку вказати ім'я вектора, потім поставити знак присвоювання '=', а далі, – відкривальну *квадратну* дужку, а за нею ввести задані значення елементів вектора, *відділяючи їх пропусками або комами*. Закінчується рядок записом квадратної дужки, що закриває.

Наприклад, запис рядка V = [1.2 -0.3 1.2e-5] задає вектор V, що містить три елементи зі значеннями 1.2, -0. 3 і 1.2e-5 (рис. 3.1):

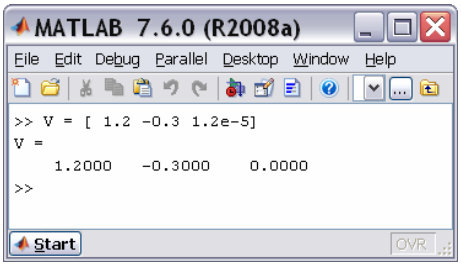


Рис. 3.1. Введення вектора

Після введення вектора система виводить його на екран. Те, що в наведеному прикладі останній елемент виведений як 0, обумовлено встановленим форматом *short*, відповідно до якого виводяться дані на екран.

Довгий вектор можна вводити частинами, які потім об'єднувати за допомогою операції *об'єднання векторів у рядок :* v = [v1 v2]. Приклад наведений на рис. 3.2.

Мова Matlab дає користувачеві можливість *скороченого введення вектора, значення елементів якого є арифметичною прогресією*. Якщо позначити: nz – початкове значення цієї прогресії (значення першого елемента вектора); kz – кінцеве значення прогресії (значення останнього елемента вектора); h – різницю прогресії (крок), то вектор можна ввести за допомогою короткого запису

V = nz : h : kz .

Наприклад, введення рядка V = - 0.1 : 0.3 : 1.4 приведе до результату, показаному на рис. 3.3.

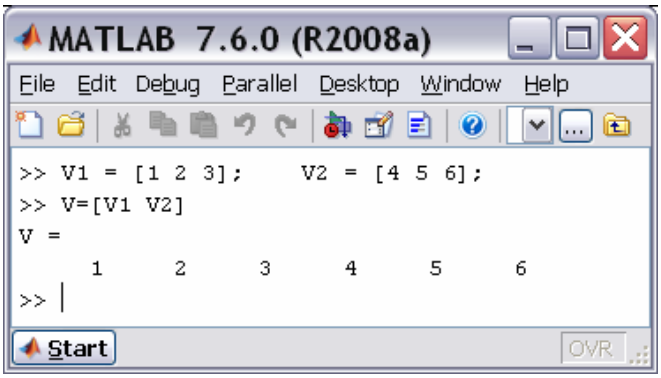


Рис. 3.2. Конкатенація (об'єднання) векторів

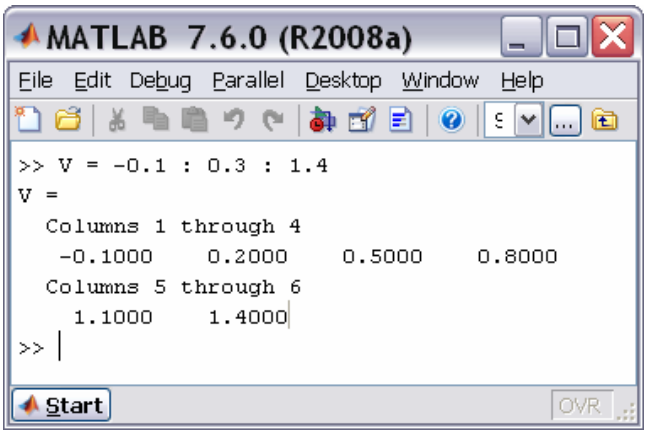


Рис. 3.3. Введення вектора арифметичної прогресії

Якщо середній параметр (різниця прогресії) не зазначений, то він за замовчуванням приймається рівним одиниці. Наприклад, команда

>> -2. 1 : 5

приводить до формування такого вектора

```
ans =
  -2.1000 -1.1000 -0.1000 0.9000 1.9000 2.9000 3.9000   4. 9000
```

У такий спосіб уводяться вектори-рядки. *Вектор-стовпець* вводиться аналогічно, але *значення елементів відокремлюються знаком ";".*

Введення значень елементів *матриці* здійснюється в Matlab у квадратних дужках, *по рядках*. При цьому елементи рядка матриці один від одного відокремлюються пропуском або комою, а *рядки один від одного відокремлюються знаком ";"* (рис. 3.4).

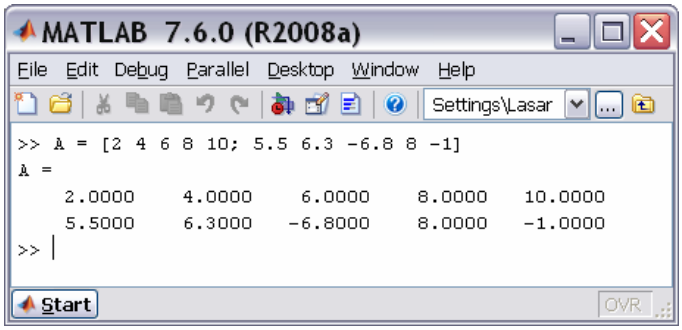


Рис. 3.4. Приклад введення матриці

3.2. Дії над векторами

Розрізнюватимемо дві групи дій над векторами:

а) векторні дії – тобто такі, що передбачені векторним зчисленням у математиці;

б) *дії по перетворенню елементів* – це дії, що перетворюють елементи вектора, але не є операціями, дозволеними математикою з векторами.

3.2.1. Векторні дії над векторами

Додавання векторів. Як відомо, складатися (підсумовуватися) можуть тільки вектори однакового типу (тобто такі, які обидва є або векторами-рядками, або векторами-стовпцями), що мають однакову довжину (тобто однакову кількість елементів). Якщо X і Y є саме такими векторами, то їхню суму Z можна одержати, увівши команду Z = X + Y, наприклад:

```
» x = [1 2 3] ;    y = [ 4 5 6];
» v = x + y
v =    5    7    9
```

Аналогічно за допомогою арифметичного знака "– " здійснюється ***віднімання векторів***, що мають однакову структуру (Z = X – Y).

Наприклад:

```
» v = x – y
v =   -3   -3   -3
```

Транспонування вектора здійснюється застосуванням знака апострофу, що записується одразу за записом імені вектора, який транспонується. Наприклад:

```
» x'
ans =
          1
          2
          3
```

Множення вектора на число здійснюється в Matlab за допомогою знака арифметичного множення (*) у такий спосіб: Z = X*r або Z = r*X, де r – деяке дійсне число.

Приклад:

```
» v = 2*x
  v =    2    4    6
```

Множення двох векторів визначено у математиці тільки для векторів однакового розміру (довжини) і лише тоді, коли один із векторів-множників є рядком, а другий – стовпчиком. Тобто, якщо вектори X і Y є рядками, то математичний зміст мають лише дві форми множення цих векторів: U = X' * Y і V = X * Y'. Причому в першому випадку результатом буде квадратна матриця, а в другому – число.

У MatLAB множення векторів здійснюється застосуванням звичайного знака множення (*), який записується між множниками-векторами.

Приклад:

```
» x = [1 2 3] ;    y = [ 4 5 6];
» v = x' * y
v =
```

```
    4     5     6
    8    10    12
   12    15    18
» v = x * y'
v =    32
```

Для *трикомпонентних векторів* у Matlab існує функція ***cross***, яка дозволяє знайти ***векторний добуток двох векторів***. Для цього, якщо задані два трикомпонентних вектори v1 і v2, достатньо ввести оператор

cross(v1, v2).

Приклад:

```
» v1 = [1 2 3];   v2 = [4 5 6];
» cross(v1,v2)
ans =   -3     6    -3
```

На цьому перелік припустимих математичних операцій з векторами вичерпується.

3.2.2. Поелементне перетворення векторів

У мові Matlab є ряд операцій, які перетворюють заданий вектор в інший того ж розміру й типу, але в той же час не є математичними операціями з вектором як математичним об'єктом. Усі ці операції перетворюють елементи вектора як елементи звичайного одновимірного масиву чисел. До таких операцій належать, наприклад, *усі елементарні математичні функції*, наведені раніше і які залежать від одного аргументу. У мові Matlab запис, наприклад, виду Y = sin(X), де X – деякий відомий вектор, приводить до формування нового вектора Y, що має той самий тип і розмір, але елементи якого дорівнюють синусам відповідних елементів вектора-аргументу X. Наприклад:

```
» x = [ -2,-1,0,1,2];
» y = sin(x)
y =   -0. 9093   -0. 8415        0    0. 8415    0. 9093
» z = tan(x)
z =    2. 1850   -1. 5574        0    1. 5574   -2. 1850
» v = exp(x)
v =         0. 3679    1. 0000    2. 7183    7. 389
```

Крім цих операцій у MatLAB передбачено декілька операцій *поелементного перетворення*, що здійснюються за допомогою знаків звичайних арифметичних дій. *Ці операції застосовуються до векторів однакового типу й розміру*. *Результатом їх є вектор того ж типу й розміру*.

Додавання (віднімання) числа до (від) кожного елемента вектора. Здійснюється за допомогою знаку "+ " ("– ").

Поелементне множення векторів. Проводиться за допомогою сукупності знаків " .* ", що записується між іменами векторів, які перемножуються. У результаті утворюється вектор, кожний елемент якого є добутком відповідних елементів векторів –"співмножників".

Поелементне ділення векторів. Здійснюється за допомогою сукупності знаків "./". Результат – вектор, кожний елемент якого є часткою від ділення відповідного елемента першого вектора на відповідний елемент другого вектора.

Поелементне ділення векторів в оберненому напрямку. Здійснюється за допомогою сукупності знаків ".\". В результаті отримують вектор, кожний елемент якого є часткою від ділення відповідного елемента другого вектора на відповідний елемент першого вектора.

Поелементне піднесення до степеня. Здійснюється за допомогою сукупності знаків ".^". Результат – вектор, кожний елемент якого є відповідним елементом першого вектора, піднесеним до степеня, розмір якого дорівнює значенню відповідного елемента другого вектора.

Приклади

```
» x = [1,2,3,4,5];          y = [-2,1,4,0,5];
» disp(x + 2)
    3    4    5    6    7
» disp(y – 3)
   -5   -2    1   -3    2
» disp(x. *y)
   -2    2   12    0   25
» disp(x. /y)
Warning: Divide by zero
  -0. 5000   2. 0000   0. 7500      Inf   1. 0000
» disp(x. \y)
  -2. 0000   0. 5000   1. 3333        0   1. 0000
» disp(x. ^y)
        1         2        81         1      3125
```

Вищевказані операції дозволяють дуже просто обчисляти (а потім – будувати графіки) складних математичних функцій, не використовуючи при цьому оператори циклу, тобто робити побудову графіків у режимі калькулятора.

Для цього достатньо задати значення аргументу як арифметичну прогресію так, як це було показано раніше, а потім записати потрібну функцію, використовуючи знаки поелементного перетворення векторів.

Наприклад, нехай потрібно обчислити значення функції:

$$y = ae^{-hx}\sin x$$

при значеннях аргументу ***x*** від 0 до 10 із кроком 1. Обчислення масиву значень цієї функції у зазначених умовах можна здійснити за допомогою лише двох простих операторів :

```
» a = 3; h = 0.5;
» x = 0:1:10;
» y = a * exp(-h*x) . * sin(x)
y =
  Columns 1 through 7
        0 1.5311 1.0035 0.0945 -0.3073 -0.2361 -0.0417
  Columns 8 through 11
   0. 0595   0. 0544   0. 0137   -0. 0110
```

3.2.3. Операції з поліномами

В системі Matlab передбачені деякі додаткові можливості математичного оперування з поліномами.

Поліном (багаточлен) як функція визначається виразом :

$$P(x) = a_n \cdot x^n + ... + a_2 \cdot x^2 + a_1 \cdot x + a_0 .$$

В системі Matlab поліном задається й зберігається у виді вектора, елементами якого є коефіцієнти полінома від a_n до a_0 :

$$P = [a_n \ ... \ a_2 \ a_1 \ a_0].$$

Уведення полінома у Matlab здійснюється у такий самий спосіб, як і введення вектора довжиною n+1, де n – порядок полінома.

Множення поліномів. Добутком двох поліномів степенів n і m відповідно, як відомо, називають поліном степеня n+m, коефіцієнти якого визначають простим перемножуванням цих двох поліномів. Фактично операція множення двох поліномів зводиться до побудови розширеного вектора коефіцієнтів по заданих векторах коефіцієнтів поліномів-співмножників. Цю операцію в математиці називають *згорткою векторів* (а самий вектор, одержуваний у результаті такої процедури – *вектором-згорткою двох векторів*). У Matlab її здійснює функція ***conv***(P1, P2).

Аналогічно, функція ***deconv***(P1, P2) здійснює ***ділення*** полінома P1 на поліном P2, тобто *обернену згортку векторів* P1 і P2. Вона визначає коефіцієнти полінома, що є часткою від ділення P1 на P2.

Приклад :

```
» p1 = [1,2,3];   p2 = [1,2,3,4,5,6];
» p = conv(p1,p2)
p =    1    4   10   16   22   28   27   18
» deconv(p,p1)
ans =    1    2    3    4    5    6
```

У загальному випадку ділення двох поліномів призводить до одержання двох поліномів – полінома-результату (частки) і полінома-остачі. Щоб одержати обидва ці поліноми, треба оформити звернення до функції у такий спосіб:

[Q,R] = ***deconv***(B,A) .

Тоді результат буде виданий у виді вектора Q з остачею у виді вектора R таким чином, що буде виконане співвідношення

B = conv(A,Q) + R.

Система Matlab має функцію ***roots***(P), що ***обчислює вектор, елементи якого є коренями заданого полінома*** P.

Нехай потрібно знайти корені полінома:

$$P(x) = x^5 + 8x^4 + 31x^3 + 80x^2 + 94x + 20 .$$

Нижче показано, як просто це зробити :

```
» p = [1,8,31,80,94,20];
» disp(roots(p))
  -1.0000 + 3.0000i
  -1.0000 – 3.0000i
  -3.7321
  -2. 0000
  -0. 2679
```

Обернена операція – побудова вектора ***p*** коефіцієнтів полінома за заданим вектором його коренів – здійснюється функцією ***poly***:

p = poly

Тут ***r*** – заданий вектор значень коренів, ***p*** – обчислений вектор коефіцієнтів полінома. Наведемо приклад:

```
»  p = [1,8,31,80,94,20]
p =    1    8   31   80   94   20
» r = roots(p)
r =
  -1.0000 + 3.0000i
  -1.0000 – 3.0000i
  -3.7321
  -2. 0000
  -0. 2679
» p1 = poly
p1 =8. 0000   31. 0000   80. 0000   94. 0000   20. 0000
```

Зауважимо, що *одержуваний вектор не показує старшого коефіцієнта, який за замовчуванням покладається рівним одиниці*.

Ця ж функція у випадку, коли аргументом її є деяка квадратна матриця А розміром (n*n), *будує вектор характеристичного полінома цієї матриці*. Звернення

p = ***poly***(A)

формує вектор ***p*** коефіцієнтів полінома

$$p(s) = \det(s \cdot E - A) = p_1 s^n + p_2 s^{n-1} + ... + p_{n-1} s^2 + p_n s + p_{n+1}.$$

де Е – позначення одиничної матриці розміром (n*n).

Розглянемо приклад:

```
» A = [1 2 3; 5 6 0; -1 2 3]
A =
    1    2    3
    5    6    0
   -1    2    3
» p = poly(A)
p =
   1. 0000  -10. 0000   20. 0000  -36. 0000
```

Для ***обчислення значення полінома за заданим значенням його аргументу*** в Matlab передбачена функція ***polyval***. Звернення до неї здійснюється за схемою:

y = polyval(p,x),

де ***p*** – заданий вектор коефіцієнтів полінома, а ***x*** – задане значення аргументу.

Приклад:

```
» y = polyval(p,2)
y = 936
```

Якщо як аргумент поліному зазначена матриця Х, то функція ***polyval(p,X)*** обчислює матрицю Y, кожний елемент якої є значенням зазначеного полінома при значенні аргументу, рівному відповідному елементу матриці Х, наприклад:

```
p =    1    8   31   80   94   20
» X = [1 2 3; 0 -1 3; 2 2 -1]
X =
    1    2    3
    0   -1    3
    2    2   -1
» disp(polyval(p,X))
       234        936       2750
        20        -18       2750
       936        936        -18
```

У цьому випадку функція обчислює значення полінома для кожного елемента матриці X, і тому розміри вхідної й вихідної матриць однакові size(Y) = size(X).

Обчислення похідної від полінома здійснюється функцією ***polyder.*** Ця функція створює вектор коефіцієнтів полінома, який є похідною від заданого полінома. Вона має три види звернень:

dp = polyder(p) за заданим поліномом ***p*** обчислює вектор ***dp***, елементи якого є коефіцієнтами полінома-похідної від заданого:

```
» dp = polyder(p)
dp =    5   32   93   160   94;
```

dp = polyder(p1,p2) обчислює вектор ***dp***, елементи якого є коефіцієнтами полінома-похідної від добутку двох поліномів ***p1*** і ***p2***:

```
»  p1 = [1,8,31,80,94,20];
»  p2 = [1,2,16];
» p = conv(p1,p2)
p =
  Columns 1 through 6
        1    10     63     270     750      1488
  Columns 7 through 8
     5.□.      320
» dp = polyder(p)
dp =
  Columns 1 through 6
        7    60     315      1080      2250      2976
  Column 7
      1544
» dp1 = polyder(p1,p2)
dp1 =
  Columns 1 through 6
        7    60     315      1080      2250      2976
  Column 7
      1544;
```

[***q,p***] = ***polyder(p1,p2)*** обчислює похідну від відношення (***p1/p2***) двох поліномів ***p1*** і ***p2*** і видає результат у виді відношення (***q/p***) поліномів ***q*** і ***p***:

```
»  p1 = [1,8,31,80,94,20];
» p2 = [1,2,16];
» [q,p] = polyder(p1,p2)
q =         3        24       159       636      1554      2520      1464
p =    1    4   36   64   256
» z = deconv (q,p)
z =    3   12    3
» y = deconv(p1,p2)
y =    1    6    3  -22
» z1 = polyder(y)
z1 =    3   12    3.
```

3.2.4. Виведення графіків. Процедура plot

Виведення графіків у системі Matlab є настільки простою і зручною процедурою, що нею можна користуватися навіть при обчисленнях у режимі калькулятора.

Основною функцією, яка забезпечує побудову графіків на екрані дисплея, є функція ***plot***. Загальна форма звернення до цієї процедури така:

plot(x1,y1,s1,x2,y2,s2,...).

Тут x1, y1 – задані вектори, елементами яких є масиви значень аргументу (x1) і функції (y1), що відповідають першої кривої графіка; x2, y2 – масиви значень аргументу й функції другої кривої і т.д. При цьому передбачається, що значення аргументу відкладаються уздовж горизонтальної осі графіка, а значення функції – уздовж вертикальної осі. Змінні s1, s2,... є символьними (їхня вказівка не є обов'язковою). Кожна з них може містити до трьох спеціальних символів, що визначають відповідно: а) тип лінії, що з'єднує окремі точки графіка; б) тип точки графіка; в) колір лінії. Якщо змінні s не зазначені, то тип лінії за умовчанням - відрізок прямої, тип точки – пиксел, а колір установлюється (у версії 5) за такою черговістю: - *синій, зелений, червоний, блакитний, фіолетовий, жовтий, чорний і білий* - у залежності від того, яка по черзі лінія виводиться на графік. Наприклад, звернення виду ***plot***(x1,y1,x2,y2,...) призведе до побудови графіка, у якому перша крива буде лінією з відрізків прямих синього кольору, друга крива – такого ж типу зеленою лінією і т.д.

Графіки в Matlab завжди *виводяться в окреме (графічне) вікно*, яку називають *фігурою*.

Наведемо приклад. Нехай потрібно вивести графік функції

$y = 3\sin(x + \pi/3)$

на проміжку від -3π до $+3\pi$ із кроком $\pi/100$.

Спочатку треба сформувати масив значень аргументу x:

```
x = -3*pi : pi/100 : 3*pi,
```

потім обчислити масив відповідних значень функції:

```
y = 3*sin(x+pi/3)
```

і, нарешті, побудувати графік залежності y(x).

У цілому в командному вікні ця послідовність операцій буде виглядати так:

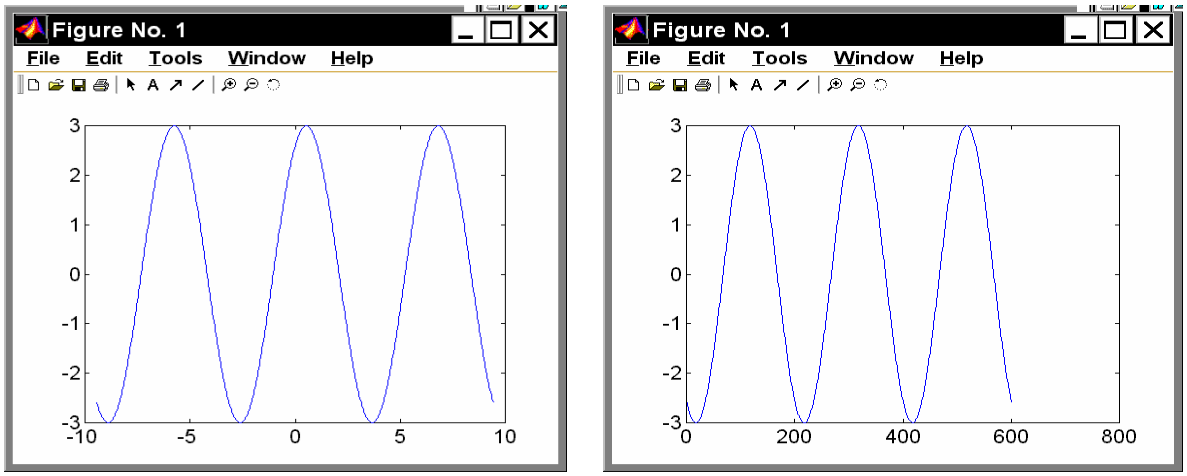
```
» x = -3*pi:pi/100:3*pi;
» y = 3*sin(x+pi/3);
» plot(x,y)
```

У результаті на екрані з'явиться додаткове вікно із графіком (див. рис. 3.5а).

Якщо вектор аргументу при зверненні до функції ***plot*** *не зазначено явно, то система обирає за умовчанням як аргумент номер елемента вектора функції.* Наприклад, якщо ввести команду

```
» plot(y),
```

то результатом буде поява графіка у виді, наведеному на рис. 3.5б.



а) *б)*

Рис. 3.5. Виведення графіка синусоїди

Графіки, наведені на рис. 3.5, мають декілька недоліків:

- на них не нанесено сітку з координатних ліній, що утруднює "зчитування" графіків;
- немає загальної інформації про криві графіка (заголовка);
- невідомо, які величини відкладені по осях графіка.

Перший недолік усувається за допомогою функції ***grid***. Якщо цю функцію записати одразу після звернення до функції ***plot***:

```
» x = -3*pi:pi/100:3*pi;
» y = 3*sin(x+pi/3);
» plot(x,y),  grid,
```

то графік буде споряджений координатною сіткою (рис. 3.6).

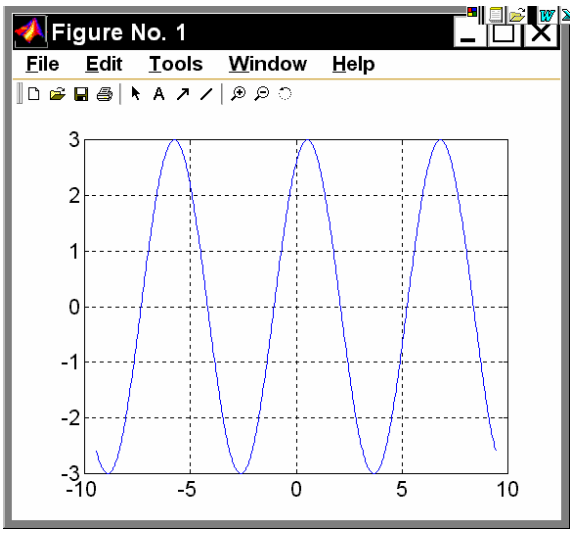


Рис.3.6. Застосування ***grid***

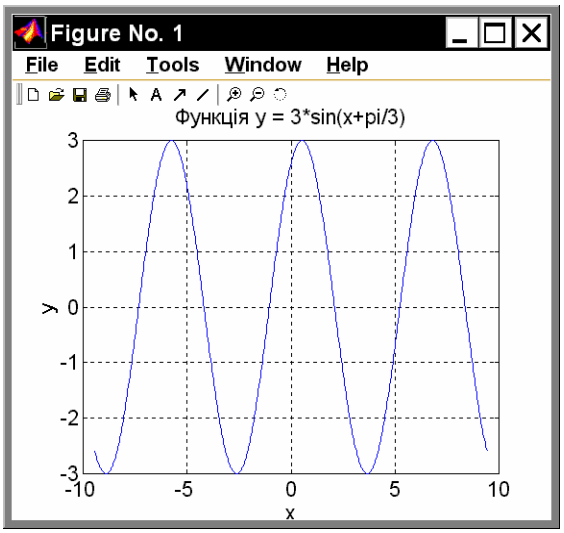


Рис. 3.7. Застосування ***title, xlabel, ylabel***

Цінною особливістю графіків, побудованих у системі Matlab, є те, що *сітка координат завжди відповідає "цілим" крокам змінювання*, що робить графіки "читабельними", тобто за графіком можна робити "відлік" значення функції при будь-якому заданому значенні аргументу і навпаки. Такої властивості не

має жодний із графічних пакетів-додатків до мов програмування високого рівня.

Заголовок графіка виводиться за допомогою процедури ***title.*** Якщо після звернення до процедури ***plot*** викликати ***title*** у такий спосіб:

title('<текст>'),

то понад графіком з'явиться текст, записаний між апострофами у дужках. При цьому варто пам'ятати, що *текст завжди має міститися в апострофах*.

Аналогічно можна вивести пояснення до графіка, що розміщаються уздовж горизонтальної осі (функція ***xlabel***) і уздовж вертикальної осі (функція ***ylabel***).

Наприклад, сукупність операторів

```
» x = -3*pi : pi/100 : 3*pi;
» y = 3*sin(x+pi/3);
» plot(x,y), grid
» title('Функція y = 3*sin(x+pi/3)');
» xlabel('x');    ylabel('y');
```

призведе до оформлення поля фігури у виді, поданому на рис. 3.7. Очевидно, така форма вже цілком задовольняє вимоги, що висуваються до інженерних графіків.

Не більш складним є виведення у середовищі Matlab графіків функцій, заданих *параметрично*. Нехай, наприклад, необхідно побудувати графік функції y(x), що задана параметричними формулами:

$$x = 4e^{-0,05t}\sin t \qquad y = 0,2e^{-0,1t}\sin 2t.$$

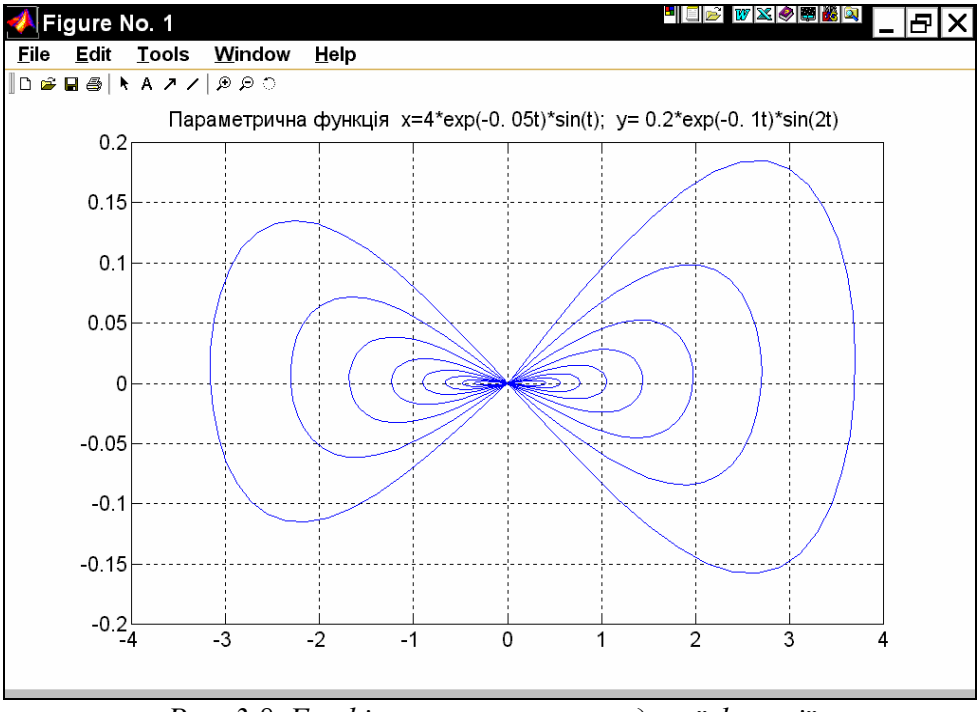


Рис. 3.8. Графік параметрично заданої функції

Виберемо діапазон змінювання параметра *t* від 0 до 50 із кроком 0.1. Тоді, набираючи сукупність операторів

```
» t = 0:0.1:50;
» x = 4*exp(-0.05*t).*sin(t);
```

```
» y = 0.2*exp(-0. 1*t). *sin(2*t);
» plot(x,y)
» title('Параметрична функція  x=4*exp(-0. 05t)*sin(t);  y= 0.2*exp(-0. 1t)*sin(2t) ')
» grid,
```

одержимо графік рис. 3.8.

3.2.5. Спеціальні графіки

Великою зручністю, наданою системою Matlab, є зазначена раніше можливість не вказувати аргумент функції при побудові її графіка. У цьому випадку як аргумент система приймає номер елемента вектора, графік якого будується. Користуючись цим, наприклад, можна побудувати "графік вектора":

```
» x = [ 1 3 2 9 6 8 4 6];
» plot (x)
» grid
» title('Графік вектора X')
» ylabel('Значення елементів')
» xlabel(' Номер елемента').
```

Результат поданий на рис. 3.9.

Ще більш наочним є подання вектора у виді *стовпцевої діаграми* за допомогою функції ***bar*** (див. рис. 3.10):

```
» bar(x)
» title('Графік  вектора X')
» xlabel(' Номер елемента')
» ylabel('Значення елементів')
```

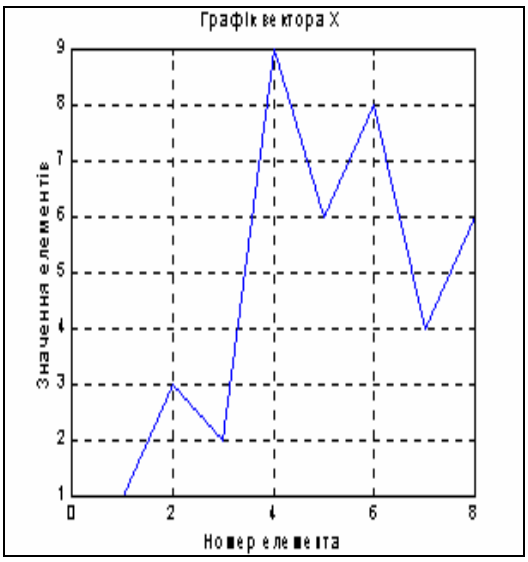


*Рис. 3.9. Застосування **plot***

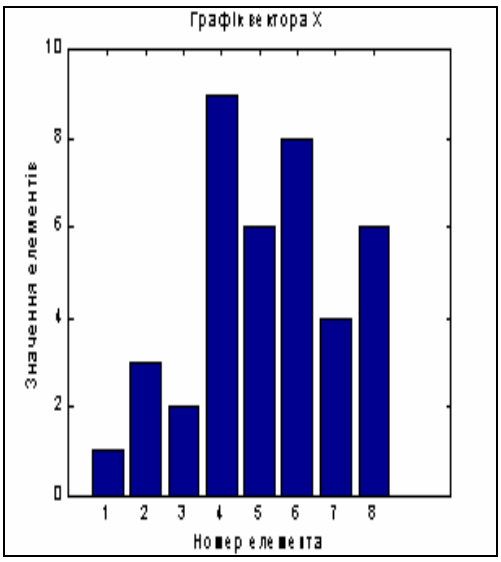


*Рис 3.10. Застосування **bar***

Якщо функція задана своїми значеннями при дискретних значеннях аргументу, і невідомо, як вона може змінюватися в проміжках між значеннями аргументу, зручніше подавати графік такої функції у виді окремих вертикальних ліній для кожного із заданих значень аргументу. Це можна зробити, застосовуючи процедуру ***stem***, звернення до якої цілком аналогічно зверненню до процедури ***plot***:

```
x = [ 1 3 2 9 6 8 4 6];
stem(x,'k')
grid
```

```
set(gca,'FontName','Arial','FontSize',14),
title('Графік вектори X')
ylabel('Значення елементів')
xlabel(' Номер елемента')
```

На рис. 3.11 зображено одержаний при цьому графік.

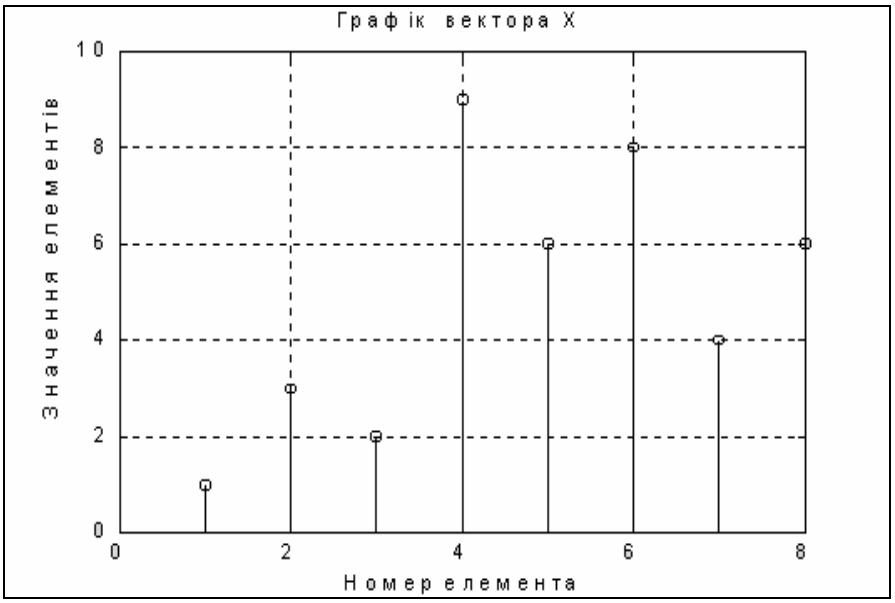


Рис. 3.11. Застосування ***stem***

Інший приклад – побудова графіка функції $y = e^{-x^2}$ у виді стовпцевої діаграми (рис. 3.12):

```
» x = - 2.9 : 0.2 : 2.9;
» bar(x, exp(-x . * x))
» title('Стовпцева діаграма функції y = exp(-x^2)')
» xlabel (' Аргумент x')
» ylabel (' Значення функції y')
```

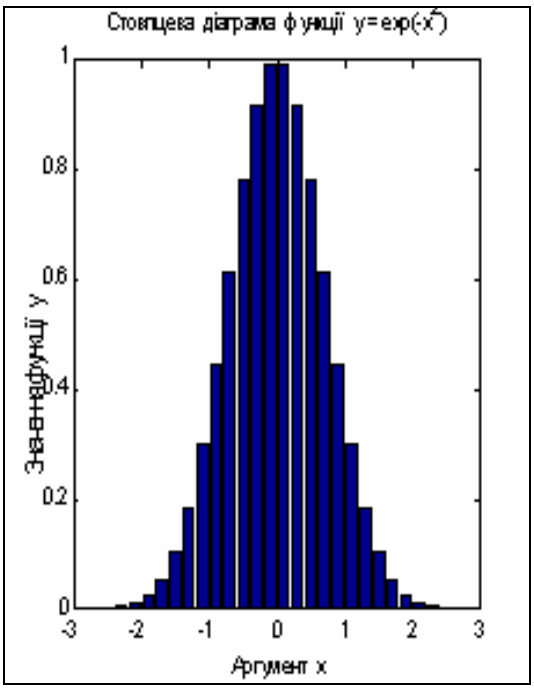


Рис. 3.12. Процедура ***bar***

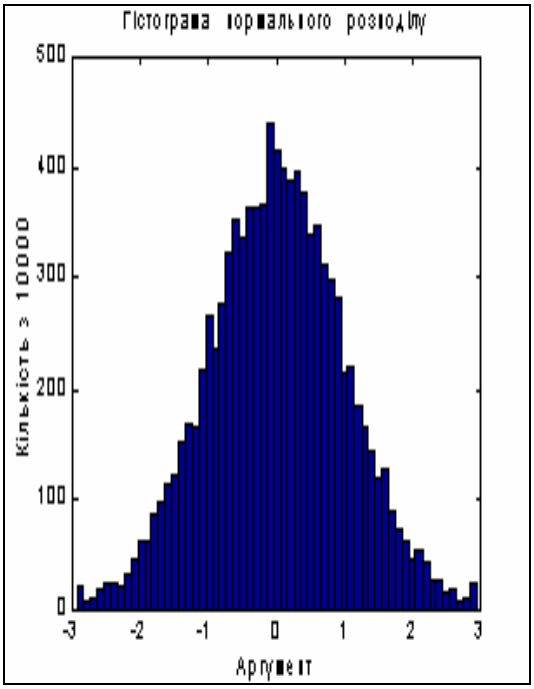


Рис. 3.13. Процедура ***hist***

Ще одна корисна інженеру функція – ***hist*** (побудова графіка *гістограми* заданого вектора). Стандартне звернення до неї має вид:

hist(y,x),

де y – вектор, гістограму якого потрібно побудувати; x – вектор, що визначає інтервали змінювання першого вектора, усередині яких підраховується кількість елементів вектора "y".

Ця функція робить дві операції

- підраховує кількість елементів вектора "y", значення яких потрапляють усередину відповідного діапазону, зазначеного вектором "x";

- будує стовпцеву діаграму підрахованих чисел елементів вектора "y" як функцію діапазонів, зазначених вектором "x" .

Як приклад роздивимося побудову гістограми випадкових величин, що формуються вмонтованою функцією ***randn***. Візьмемо загальну кількість елементів вектора цих випадкових величин 10 000. Побудуємо гістограму для діапазону змінювання цих величин від -2,9 до +2,9. Інтервали змінювання нехай будуть рівні 0,1. Тоді графік гістограми можна побудувати за допомогою сукупності таких операторів:

```
» x = -2.9:0.1:2.9;
» y = randn(10000,1);
» hist(y,x)
» ylabel('Кількість з 10000')
» xlabel( 'Аргумент')
» title('Гістограма нормального розподілу')
```

Результат поданий на рис. 3.13. З нього, зокрема, випливає, що вмонтована функція ***randn*** достатньо вірно відображає нормальний гауссовий закон розподілу випадкової величини.

Процедура ***comet***(x,y) ("комета") будує графік залежності y(x) поступово у часі у виді траєкторії комети. При цьому зображувальна точка на графіку має вид маленької комети (із голівкою й хвостиком), що плавно переміщується від однієї точки до іншої. Наприклад, якщо ввести сукупність операторів:

```
» t = 0:0.1:50;
» x = 4 * exp(-0.05*t) .* sin(t);
» y = 0.2 * exp(-0. 1*t) . * sin(2*t);
» comet(x,y),
```

то графік, наведений на рис. 3.8, буде побудований як траєкторія послідовного руху комети. Ця обставина може бути корисною при побудові просторових траєкторій для виявлення характеру змінювання траєкторії з часом.

Matlab має декілька функцій, що дозволяють будувати графіки в логарифмічному масштабі.

Функція ***logspace*** із зверненням

x = ***logspace***(d1, d2, n)

формує вектор-рядок "x", що містить "n" рівновіддалених у логарифмічному масштабі одна від одної точок, що покривають діапазон від 10^{d1} до 10^{d2}.

Функція ***loglog*** цілком аналогічна функції ***plot***, але графіки по обох осях будуються в логарифмічному масштабі.

Для побудови графіків, які використовують логарифмічний масштаб тільки по однієї з координатних осей, користуються процедурами ***semilogx*** і ***semilogy***. Перша процедура будує графіки з логарифмічним масштабом уздовж горизонтальної осі, друга – уздовж вертикальної осі.

Звернення до останніх трьох процедур цілком аналогічно зверненню до функції ***plot***.

Як приклад розглянемо побудову графіків амплітудно-частотної й фазочастотної характеристик ланки, що описується передатною функцією:

$$W(p) = \frac{p+4}{p^2+4p+100}.$$

Для цього треба, по-перше, створити поліном чисельника Pc = [1 4] і знаменника передатної функції Pz = [1 4 100]. По-друге, визначити корені цих двох поліномів:

```
» P1 = [1 4];     P2 = [1 4 100];
» roots(P1)
ans =    -4
» roots(P2)
ans =
-2. 0000e+000 +9. 7980e+000i
-2. 0000e+000 -9. 7980e+000i
```

По-третє, задати діапазон змінювання частоти так, щоб він охоплював усі знайдені корені:

om0 = 1e-2; omk = 1e2.

Тепер потрібно задатися кількістю точок майбутнього графіка (наприклад, n = 41), і сформувати масив точок по частоті

OM = ***logspace***(-2,2,41),

де значення -2 і +2 відповідають десятковим порядкам початкового om0 і кінцевого omk значень частоти.

Користуючись функцією ***polyval***, можна обчислити спочатку вектор "ch" комплексних значень чисельника частотної передатної функції, що відповідають заданій передатній функції за Лапласом, якщо як аргумент функції ***polyval*** використати сформований вектор частот OM, елементи якого помножені на уявну одиницю (див. визначення частотної передатної функції). Аналогічно обчислюється комплекснозначний вектор "zn" знаменника ЧПФ.

Вектор значень АЧХ (амплітудно-частотної характеристики) можна знайти, вираховуючи модулі векторів чисельника і знаменника ЧПФ і поелементно ділячи отримані вектори. Щоб знайти вектор значень ФЧХ (фазочастотної характеристики) треба розділити поелементно комплекснозначні вектори чисельника й знаменника ЧПФ і визначити вектор аргументів елементів отриманого вектора. Для того щоб фази подати в градусах, отримані результати варто помножить на 180 і розділити на π.

Нарешті, для побудови графіка АЧХ у логарифмічному масштабі, достатньо застосувати функцію ***loglog***, а для побудови ФЧХ зручніше скористатися функцією ***semilogx.***

У цілому послідовність дій може бути такою:

```
» OM = logspace(-2,2,40)
```

```
» ch = polyval(P1,i*OM);
» zn = polyval(P2,i*OM);
» ACH = abs(ch). /abs(zn);
» loglog(OM,ACH);      grid;
>> title('Графік Амплітудно-Частотної Характеристики')
>>       xlabel('Частота  (рад/с)');
>>       ylabel('Відношення  амплітуд')
» FCH = angle(ch. /zn)*180/pi;
» semilogx(OM,FCH);  grid,
>>       title('Фазо-Частотна Характеристика'),
>> xlabel('Частота  (рад/с)'),
>>       ylabel('Фаза  (градуси)')
```

В результаті утворюються графіки, зображені на рис. 3.14 і 3.15.

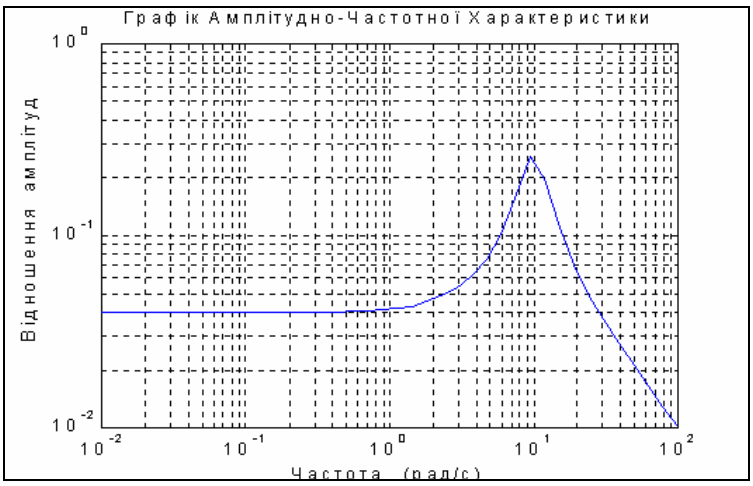


Рис. 3.14. Графік АЧХ

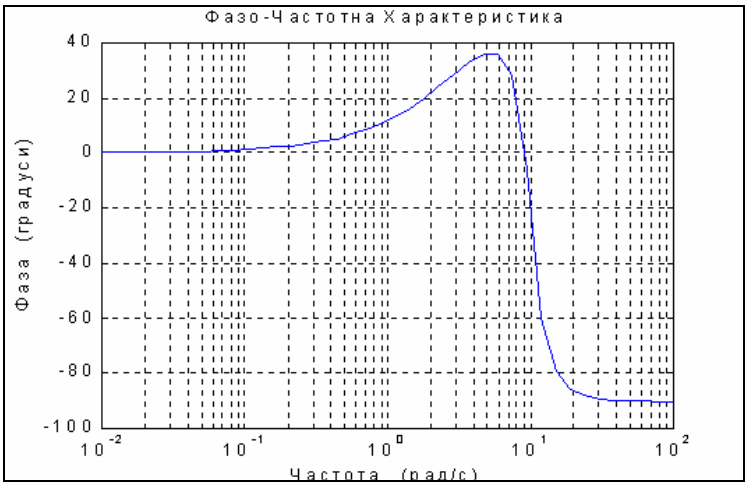


Рис. 3.15. Графік ФЧХ

3.2.6. Додаткові функції графічного вікна

Зазвичай графіки, одержувані за допомогою процедур ***plot***, ***loglog,*** ***semilogx*** і ***semilogy***, автоматично будуються в таких масштабах по осях, щоб у поле графіка помістилися всі обчислені точки графіка, включаючи максимальні і мінімальні значення аргументу й функції. Проте Matlab має можливості вста-

новлення й інших режимів масштабування. Це досягається за рахунок використання процедури ***axis***.

Команда

axis([xmin xmax ymin ymax])

установлює жорсткі межі поля графіка в одиницях величин, що відкладаються по осях.

Команда ***axis***('auto') повертає масштаби по осях до їх штатних значень (прийнятих за замовчуванням).

Команда ***axis***('ij') переміщує початок відліку в лівий верхній ріг і реалізує відлік від верхнього лівого рогу (матрична система координат).

Команда ***axis***('xy') повертає декартову систему координат із початком відліку в лівому нижньому рогові.

Команда ***axis***(‘square') установлює однаковий діапазон змінювання змінних по осях графіка.

Команда ***axis***(‘equal') забезпечує однаковий масштаб по обох осях графіка.

В однім графічному вікні, але на окремих графічних полях можна побудувати декілька графіків, використовуючи процедуру ***subplot***. Звернення до цієї процедури має передувати зверненню до процедур ***plot***, ***loglog, semilogx*** і ***semilogy*** і мати такий вид:

subplot(m,n,p).

Тут m – указує, на скільки частин розділяється графічне вікно по вертикалі, n – по горизонталі, а p – номер підвікна, у якому буде будуватися графік. При цьому підвікна нумеруються зліва праворуч порядково зверху униз (так, як по рядках читається текст книги).

Наприклад, два попередні графіки можна помістити в одне графічне вікно в такий спосіб:

```
subplot(2,1,1);
loglog(OM,ACH,'k'); grid;
set(gca,'FontName','Arial','FontSize',14),
title('Амплітудно-Частотна Характеристика')
ylabel('Амплітуда')
subplot(2,1,2);
semilogx(OM,FCH,'k'); grid
set(gca,'FontName','Arial','FontSize',14),
title('Фазо-Частотна Характеристика')
xlabel('Частота  (рад/с)'),     ylabel('Фаза (гр.)')
```

Результат поданий на рис. 3.16.

Команда ***text***(x, y, ‘<текст>’) дозволяє розмістити зазначений текст на поле графіка, при цьому початок тексту поміщається в точку з координатами x і y. Значення зазначених координат повинні бути подані в одиницях величин, що відкладаються по осях графіка, і знаходитися усередині діапазону змінювання цих величин. Часто це незручно, бо потребує попереднього знання цього діапазону, що рідко коли є можливим.

Більш зручним для розміщення тексту усередині поля графіка є використання команди ***gtext***(‘<текст>’), яка висвічує в активному графічному вікні перехрестя, переміщення якого за допомогою "мишки" дозволяє вказати місце

початку виведення зазначеного тексту. Після цього натисканням лівої клавіші "мишки" або будь-якої клавіші текст вводиться в зазначене місце.

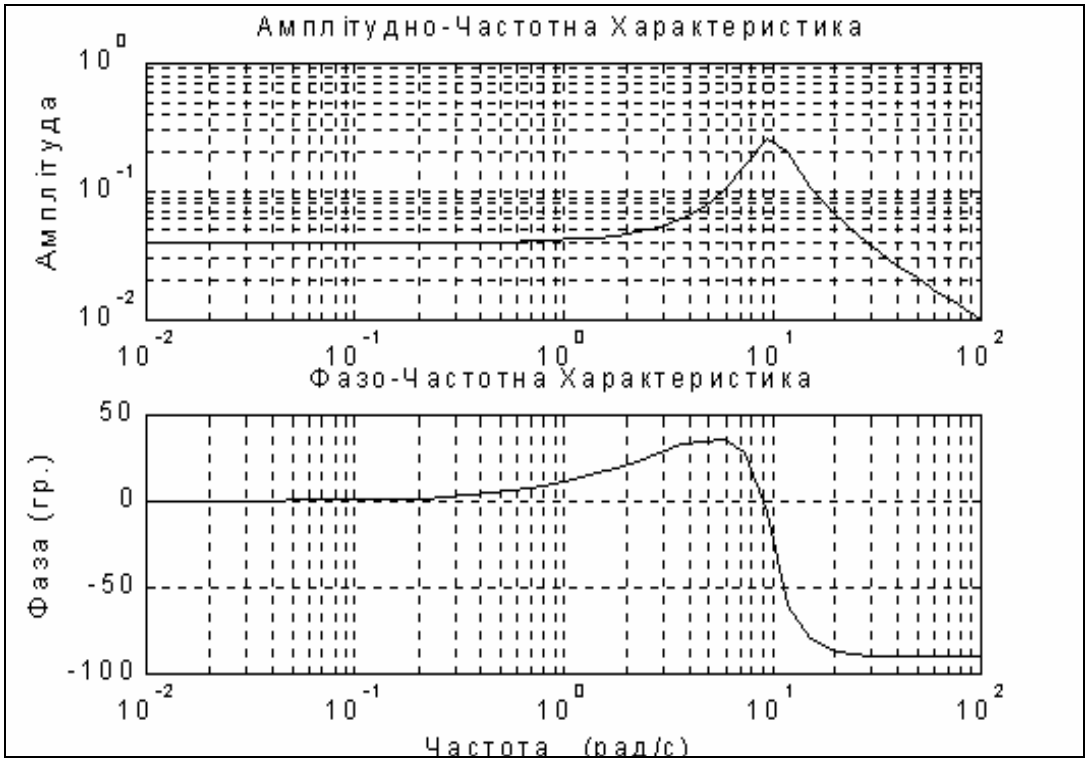


Рис.3.16. Використання функції ***subplot***

Щоб створити декілька графічних вікон, у кожному з яких розташовані відповідні графіки, можна скористатися командою ***figure***, яка створить таке графічне вікно, залишаючи попередні.

Нарешті, для того, щоб декілька послідовно обчислюваних графіків були зображені в одному графічному вікні в одному стилі і одному графічному полі, можна використати команду ***hold on***, тоді кожний такий графік буде будуватися в тому ж попередньо відкритому графічному вікні, тобто кожна нова лінія буде додаватися до раніше побудованих. Команда ***hold off*** виключає режим зберігання графічного вікна, встановленого попередньою командою.

3.2.7. Вставлення графіків до документу

Щоб вставити графік із графічного вікна (фігури) до документу Word, слід скористатися командами меню, розташованого у верхній частині вікна фігури. У меню ***Edit*** оберіть команду ***Copy Figure***. Перейдіть у вікно документу і натисніть клавіші <Ctrl>+<C>. Вміст графічного вікна виникне у тому місці документу, де містився курсор.

3.2.8. Апроксимація та інтерполяція даних

Поліноміальна апроксимація даних вимірів, що сформовані як деякий вектор Y, при деяких значеннях аргументу, які утворюють вектор X такої ж довжини, що й вектор Y, здійснюється процедурою ***polyfit***(X, Y, n). Тут n – порядок апроксимувального полінома. Результатом дії цієї процедури є вектор довжиною (n+1) із коефіцієнтів апроксимувального полінома.

Нехай масив значень аргументу є таким:

x = [1 2 3 4 5 6 7 8],

а масив відповідних значень виміряної величини – таким:

y = [-1. 1 0.2 0.5 0.8 0.7 0.6 0.4 0.1].

Тоді, застосовуючи зазначену функцію при різних значеннях порядку апроксимуючого полінома, одержимо:

```
» x = [1 2 3 4 5 6 7 8];
» y = [ -1.1 0.2 0.5 0.8 0.7 0.6 0.4 0.1];
» polyfit(x,y,1)
ans =    0. 1143   -0. 2393
» polyfit(x,y,2)
ans =   -0. 1024    1. 0357   -1. 7750
» polyfit(x,y,3)
ans =    0. 0177   -0. 3410    1. 9461   -2. 6500
» polyfit(x,y,4)
ans =   -0. 0044    0. 0961   -0. 8146    3. 0326   -3. 3893.
```

Це означає, що задану залежність можна апроксимувати або прямою

$$y(x) = 0{,}1143x - 0{,}2393,$$

або квадратною параболою

$$y(x) = -0{,}1024x^2 + 1{,}0357x - 1{,}775,$$

або кубічною параболою

$$y(x) = 0{,}0177x^3 - 0{,}341x^2 + 1{,}9461x - 2{,}65,$$

або параболою четвертого степеня

$$y(x) = -0{,}0044x^4 + 0{,}0961x^3 - 0{,}8146x^2 + 3{,}0326x - 3{,}3893.$$

Побудуємо в одному графічному полі графіки заданої дискретної функції й графіки всіх отриманих при апроксимації поліномів:

```
 x = [1 2 3 4 5 6 7 8];
 y = [ -1.1 0.2 0.5 0.8 0.7 0.6 0.4 0.1];
 P1=polyfit(x,y,1) ;
 P2=polyfit(x,y,2);
 P3=polyfit(x,y,3);
 P4=polyfit(x,y,4) ;
stem(x,y);
x1 = 0.5 : 0.2 : 8.5;
y1=polyval(P1,x1);
y2=polyval(P2,x1);
y3=polyval(P3,x1);
y4=polyval(P4,x1);
hold on
plot(x1,y1,'.-',x1,y2,'x-',x1,y3,'o-',x1,y4,'^-'),
grid, set(gca, 'FontName', 'Arial Cyr', 'FontSize', 16),
title('Поліноміальна апроксимація ');
xlabel('Аргумент');
ylabel('Функція')
legend(' Задані значення','1','2','3','4',0)
```

Результат поданий на рис. 3.17.

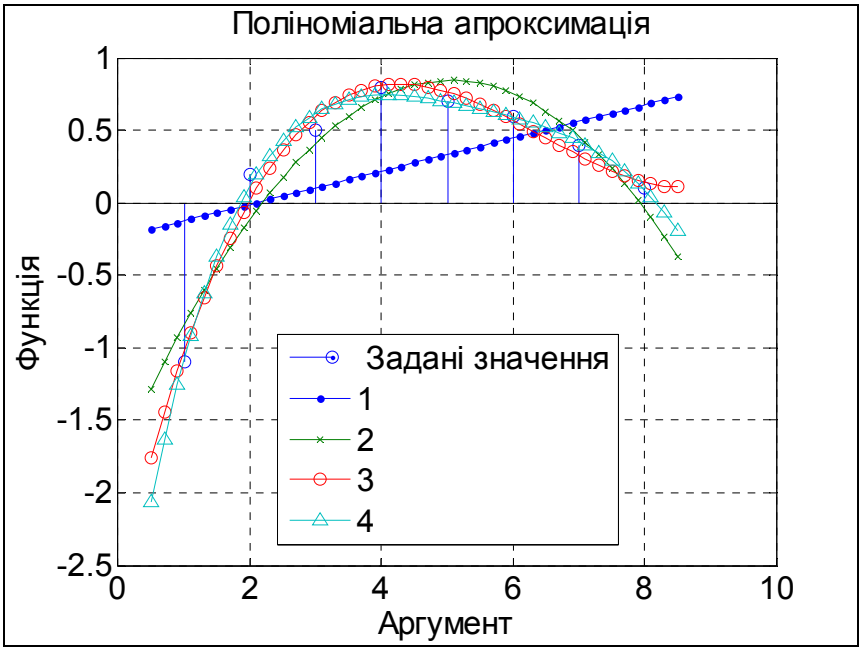


Рис. 3.17. Поліноміальна апроксимація функцією ***polyfit***

Функція ***spline***(X,Y,Xi) здійснює ***інтерполяцію кубічними сплайнами***. При зверненні

Yi = ***spline***(X,Y,Xi)

вона інтерполює значення вектора Y, заданого при значеннях аргументу, поданих у векторі X, і видає значення інтерполювальної функції у виді вектора Yi при значеннях аргументу, заданих вектором Xi. У випадку, коли вектор X не зазначений, за замовчуванням приймається, що він має довжину вектора Y і кожний його елемент дорівнює номеру цього елемента.

Як приклад розглянемо інтерполяцію вектора

```
x = -0.5:0.1:0.2;
y = [-1.1 0.2 0.5 0.8 0.7 0.6 0.4 0.1];
x1 = -0.5:0.02:0.2;
y2 = spline(x,y,x1);
plot (x,y,x1,y2,'.-'), grid
set(gca,'FontName','Arial Cyr','FontSize',16),
title('Інтерполяція процедурою SPLINE ');
xlabel('Аргумент');
ylabel('Функція')
legend('лінійна','сплайнова',0)
```

Результат наведений на рис. 3.18.

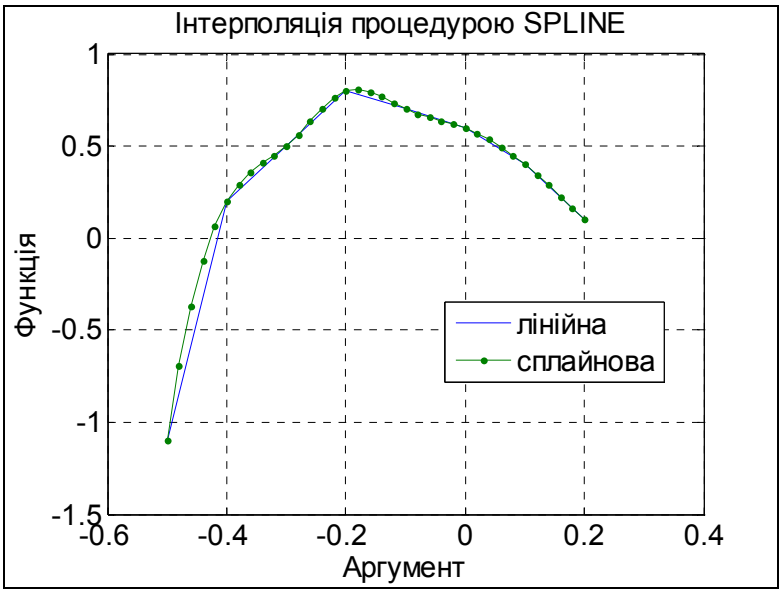


Рис. 3.18. Інтерполяція функцією ***spline***

Одновимірну табличну інтерполяцію робить процедура ***interp1***. Звернення до неї у загальному випадку має вид:

Yi = ***interp1***(X,Y,Xi,'<метод>'),

і дозволяє додатково зазначити метод інтерполяції у четвертому вхідному аргументі:

'nearest' – ступінчаста інтерполяція;
'linear' – лінійна;
'cubic' – кубічна;
'spline' – кубічними сплайнами.

Якщо метод не зазначений, здійснюється за умовчанням лінійна інтерполяція. Наприклад, (для того самого вектора):

```
x = -0.5:0.1:0.2;
y = [-1.1 0.2 0.5 0.8 0.7 0.6 0.4 0.1];
x1 = -0.5:0.02:0.2;
y1 = interp1(x,y,x1);
y4 = interp1(x,y,x1,'nearest');
y2 = interp1(x,y,x1,'cubic');
y3 = interp1(x,y,x1,'spline');
%plot (x1,y1,x1,y2,x1,y3,x1,y4), grid
plot (x1,y1,x1,y2,'.',x1,y3,x1,y4), grid
legend('лінійна','кубічна','сплайнова','ступінчаста')
set(gca,'FontName','Arial Cyr','FontSize',16),
title('Інтерполяція процедурою INTERP1 ');
xlabel('Аргумент');     ylabel('Функція')
```

Результат наведений на рис. 3.19.

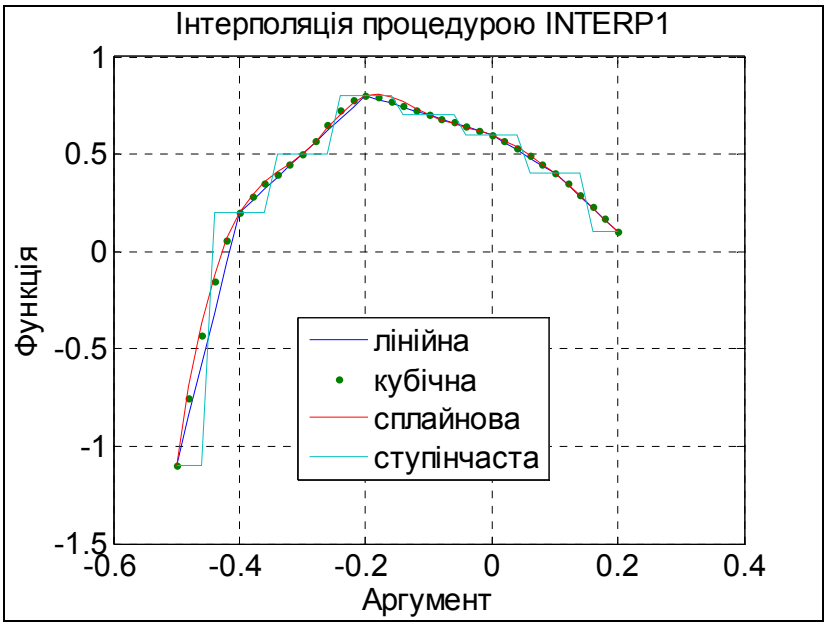


Рис. 3.19. Інтерполяція функцією ***interp1***

3.2.9. Векторна фільтрація й спектральний аналіз

У системі Matlab є декілька функцій для проведення цифрового аналізу даних спостережень (вимірів).

Так, функція ***y = filter(b,a,x)*** забезпечує формування вектора ***y*** по заданих векторах ***b, a, x*** відповідно до співвідношення:

$$y(k) = b(1)*x(k) + b(2)*x(k-1) + ... + b(nb+1)*x(k-nb) - \\ -a(2)*y(k-1) - a(3)*y(k-3) - ... - a(na+1)*y(k-na), \quad (1)$$

де вектор ***b*** має такий склад

$$\boldsymbol{b} = [\, b(1),b(2),... ,b(nb+1)],$$

а вектор ***a***

$$\boldsymbol{a} = [1,a(2),a(3),... ,a(na+1)].$$

Співвідношення (1) можна розглядати як кінцево-різницеве рівняння фільтра з дискретною передатною функцією вигляду раціонального дробу, коефіцієнти чисельника якого утворюють вектор ***b***, а знаменника – вектор ***a***, на вхід якого подається сигнал *x(t)*, а на виході формується сигнал *y(t)*.

Тоді вектор ***y*** буде являти собою значення вихідного сигналу цього фільтра в дискретні моменти часу, що відповідають заданим значенням вхідного сигналу *x(t)* (вектор ***x***).

Нижче наведений приклад застосування функції ***filter.***

```
» x = 0:0.1:1;
» b = [1 2];
» a = [ 1 0.1 4];
» y = filter(b,a,x)
y =
  Columns 1 through 7
        0    0.1000    0.3900    0.2610   -0.5861    0.3146    3.9129
  Columns 8 through 11
    0.2503  -13.4768    2.8466   56.4225
```

Функції ***fft*** (*Fast Fourier Transformation*) і ***ifft*** (*Invers Fast Fourier Transformation*) здійснюють перетворення заданого вектора, що відповідають дискретному прямому й оберненому перетворенням Фур'є.

Звернення до цих функцій виду:

y = fft (x, n); ***x = ifft (y, n)***

призводить до формування вектора ***y*** у першому випадку і ***x*** – у другому по формулах:

$$y(k) = \sum_{m=1}^{n} x(m) \cdot e^{-j \cdot 2\pi \cdot (m-1) \cdot (k-1)/n}; \quad (2)$$

$$x(m) = \frac{1}{n} \sum_{k=1}^{n} y(k) \cdot e^{j \cdot 2\pi \cdot (m-1) \cdot (k-1)/n}, \quad (3)$$

де j – позначення уявної одиниці; n – число елементів заданого вектора ***x*** (воно є також розміром вихідного вектора ***y***).

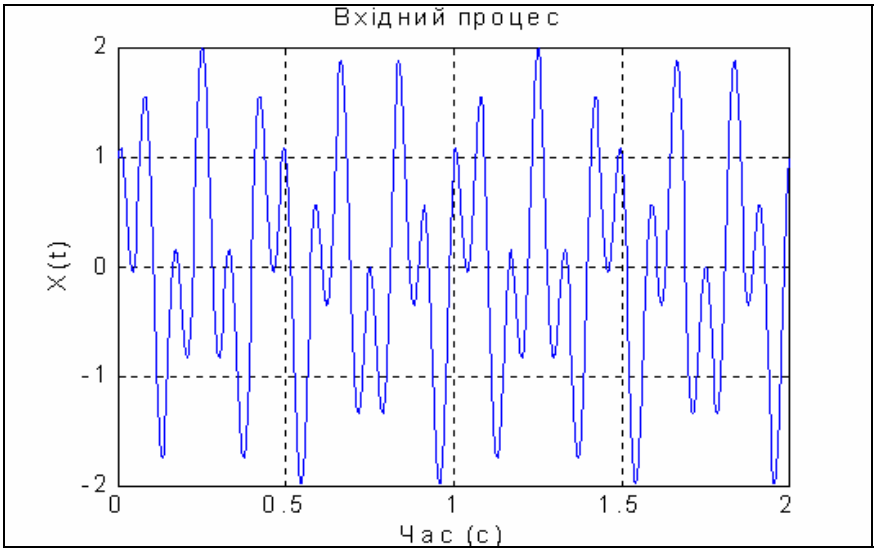


Рис. 3.20. Початковий процес

Наведемо приклад. Сформуємо вхідний сигнал у виді вектора, елементи якого дорівнюють значенням функції, що є сумою двох синусоїд із частотами 5 і 12 Гц. Знайдемо Фур'є-зображення цього сигналу і виведемо графічні подання вхідного процесу й модуля його Фур'є-зображення:

```
 t =0:0.001:2;
 x = sin(2*pi*5*t) + cos(2*pi*12*t);
plot(t, x); grid
 set(gca,'FontName','Arial Cyr','FontSize',16),
title('Вхідний процес ');
xlabel('Час (с)');
ylabel('X(t)')
 y = fft(x);
a =abs(y);
plot(a); grid
 set(gca,'FontName','Arial Cyr','FontSize',16),
title('Модуль Фурьє – зображення ');
xlabel('Номер елемента вектора');
ylabel('abs(F(X(t))')
```

Результати відображені відповідно на рис. 3.20 і 3.21.

Рис. 3.21. Результат застосування процедури ***fft***

Тепер здійснимо зворотне перетворення за допомогою функції ***ifft*** і результат також виведемо у формі графіка:

```
 z = ifft(y);
plot(t, z); grid
 set(gca,'FontName','Arial Cyr','FontSize',16),
title('Зворотне Фур''є-перетворення ');
xlabel('Час (с)');
ylabel('Z(t)')
```

На рис. 3.22 зображений результат. Розглядаючи його, можна переконатися, що відтворений процес збігається з початковим.

Рис.3.22. Результат застосування процедури ***ifft***

Уважно вивчаючи формулу дискретного перетворення Фур'є, можна дійти висновків:

а) номер m відповідає моменту часу t_m, у який виміряний вхідний сигнал $x(m)$; при цьому $t_1 = 0$;

б) номер k – це індекс значення частоти f_k, якому відповідає знайдений елемент *y(k)* дискретного перетворення Фур'є;

в) щоб перейти від індексів до часової й частотної областей, треба знати значення h дискрету (кроку) часу, через який виміряний вхідний сигнал *x(t)* і проміжок T часу, протягом якого він вимірюється; тоді крок (дискрет) по частоті в зображенні Фур'є визначиться співвідношенням:

$$Df = 1/T, \quad (4)$$

а діапазон змінювання частоти – формулою

$$F = 1/h; \quad (5)$$

так, в аналізованому прикладі (h =0. 001, T = 2, n = 21)

$$Df = 0.5; \qquad F = 1000;$$

г) із (2) випливає, що індексу k = 1 відповідає нульове значення частоти (*fo = 0)*; інакше кажучи, перший елемент вектора $y(1)$ є значенням Фур'є-зображення при нульовій частоті, тобто є просто сумою всіх заданих значень вектора ***x***; звідси одержуємо, що вектор *y(k)* містить значення Фур'є-зображення, починаючи з частоти *fo = 0* (якій відповідає k = 1) до максимальної частоти *fmax = F* (якій відповідає $k = n$); таким чином, Фур'є-зображення визначається функцією ***fft*** тільки для додатних частот у діапазоні від 0 до F; це незручно для побудови графіків Фур'є-зображення від частоти; більш зручним і звичним є перехід до вектора Фур'є-зображення, якого визначено в діапазоні частот від (-*F/2*) до $F/2$; частота $F_N = F / 2$ одержала назву *частоти Найквіста*;

д) як відомо, функція $e^{j \cdot z}$ є періодичною за z із періодом 2π; тому інформація про Фур'є-зображення при від'ємних частотах розташована в другій половині вектора *y(k)*.

Сформуємо для аналізованого прикладу масив частот, виходячи з вищезазначеного:

```
f = 0:0. 5 : 1000;
```

і виведемо графік з аргументом-частотою (рис. 3.23):

```
plot(f,a); grid
set(gca,'FontName','Arial Cyr','FontSize',16),
title('Модуль Фур'є – зображення ');
xlabel('Частота (Гц)');
ylabel('abs(F(X(t))')
```

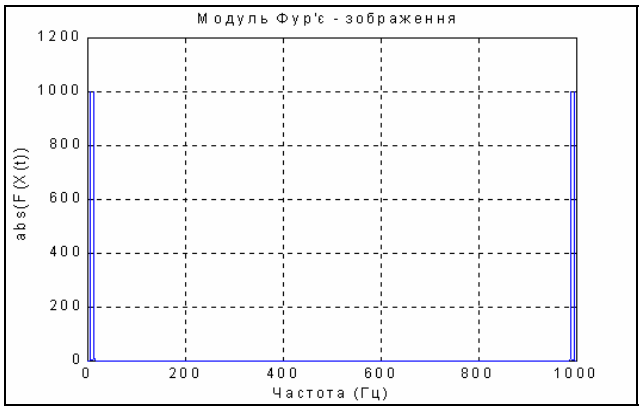


*Рис. 3.23. Результат застосування функції **fft** у частотній області*

Як випливає з розгляду рис. 3.23, за ним важко розпізнати ті частоти (5 і 12 Гц), із якими змінюється вхідний сигнал. Це – наслідок тієї обставини, яку було відзначено в примітці г). Щоб визначити справжній спектр вхідного сигналу, потрібно спочатку дещо перетворити отриманий вектор ***y*** Фур'є-зображення за допомогою процедури ***fftshift.***

Функція ***fftshift*** (звернення до неї здійснюється в такий спосіб: ***z*** = ***fftshift***(***y***)) призначена для формування нового вектора ***z*** із заданого вектора ***y*** шляхом переставлення другої половини вектора ***y*** у першу половину вектора ***z*** . При цьому друга половина вектора ***z*** складається з елементів першої половини вектора ***y***. Більш точно цю операцію можна задати співвідношеннями:

$$z(1) = y(n/2+1);\ \ldots,\ z(k) = y(n/2+k);\ldots,\ z(n/2) = y(n);\ z(n/2+1) = y(1);\ldots$$
$$\ldots,\ z(n/2+k) = y(k);\ldots\ z(n) = y(n/2).$$

Примітка. Операцію ***fftshift*** зручно використовувати для *визначення масиву Фур'є-зображення з метою побудови його графіка в частотній області. Проте цей масив не може бути використаний для зворотного перетворення Фур'є*.

Проілюструємо застосування цієї функції до попереднього приклада:

```
f1 = -500 : 0.5 : 500;      % Перебудова вектора частот
v = fftshift(y); % Перебудова вектора Фур'є-зображення
a =abs(v); % Відшукання модуля
 % Відбудова графика
plot(f1(970:1030),a(970:1030)); grid
set(gca,'FontName','Arial Cyr','FontSize',16),
title('Модуль Фур'є – зображення');
xlabel('Частота (Гц)'); ylabel('abs(F(X(t))')
```

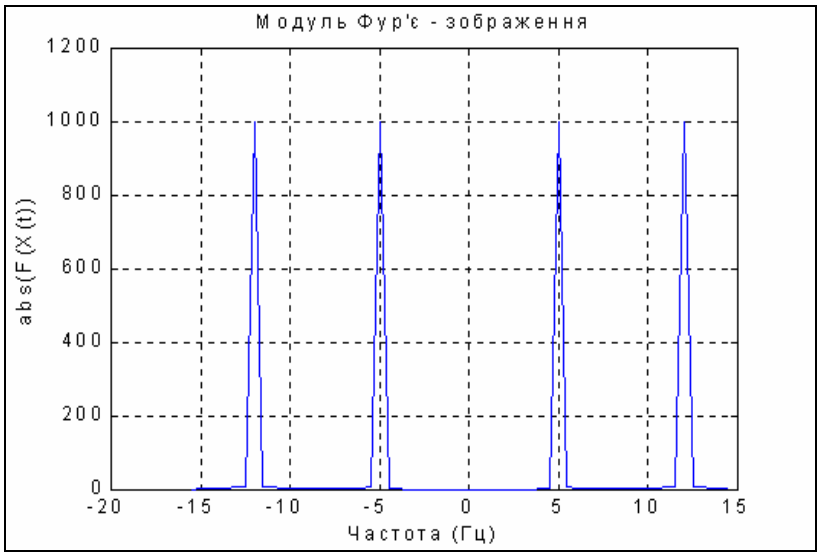


Рис. 3.24. Результат застосування процедури ***fftshift***

З графіка рис. 3.24 вже стає очевидним, що в спектрі вхідного сигналу є дві гармоніки – з частотами 5 і 12 Гц.

Залишається лише та незручність, що із графіка спектра неможливо встановити амплітуди цих гармонік. Щоб уникнути цього, потрібно весь вектор ***y*** Фур'є-зображення розділити на число його елементів (***n***), щоб отримати вектор комплексного спектра сигналу:

```
N=length(y);    a =abs(v)/N;
plot(f1(970:1030),a(970:1030)); grid
set(gca,'FontName','Arial Cyr','FontSize',16,'Color','white'),
title('Модуль комплексного спектра');
xlabel('Частота (Гц)');
ylabel('abs(F(X(t)) / N')
```

Результат наведений на рис. 3.25.

З його розгляду випливає, що "амплітуди" усіх складових гармонік рівні 0.5. При цьому потрібно взяти до уваги, що "амплітуди" розподілені між додатними й від'ємними частотами порівну, тому вони вдвічі менше за справжню амплітуду відповідної гармоніки.

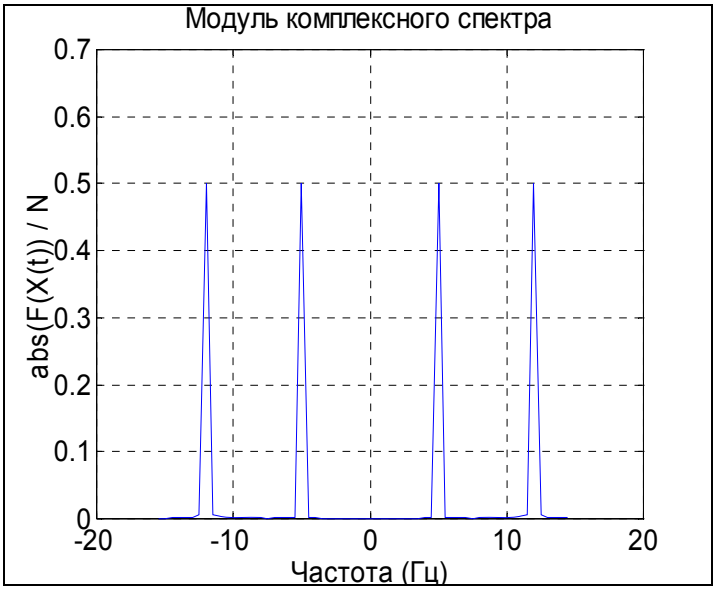


Рис. 3.25. Одержання модуля комплексного спектру

4. ОПЕРАЦІЇ З МАТРИЦЯМИ

4.1. Формування матриць

Matlab має декілька функцій, що дозволяють формувати вектори й матриці деякого певного виду. До таких функцій належать:

zeros(M,N) – створює матрицю розміром (M*N) із нульовими елементами, наприклад:

```
» zeros(3,5)
ans =
   0   0   0   0   0
   0   0   0   0   0
   0   0   0   0   0
```

ones(M,N) – створює матрицю розміром (M*N) з одиничними елементами, наприклад:

```
» ones(3,5)
ans =
   1   1   1   1   1
   1   1   1   1   1
   1   1   1   1   1
```

eye(M,N) – створює матрицю розміром (M*N) з одиницями по головній діагоналі й інших нульових елементах, наприклад:

```
» eye(3,5)
ans =
   1   0   0   0   0
   0   1   0   0   0
   0   0   1   0   0
```

rand(M,N) – створює матрицю розміром (M*N) із випадкових чисел, рівномірно розподілених у діапазоні від 0 до 1, наприклад:

```
» rand(3,5)
ans =
  2.1896e-001   6.7930e-001   5.1942e-001   5.3462e-002   7.6982e-003
  4. 7045e-002  9. 3469e-001  8. 3097e-001  5. 2970e-001  3. 8342e-001
  6. 7886e-001  3. 8350e-001  3. 4572e-002  6. 7115e-001  6. 6842e-002
```

randn(M,N) – створює матрицю розміром (M*N) із випадкових чисел, розподілених за нормальним (гауссовим) законом із нульовим математичним сподіванням і стандартним (середньоквадратичним) відхиленням, рівним одиниці, наприклад:

```
» randn(3,5)
ans =
  1.1650e+000   3.5161e-001   5.9060e-002   8.7167e-001   1.2460e+000
  6. 2684e-001 -6. 9651e-001  1. 7971e+000 -1. 4462e+000 -6. 3898e-001
  7. 5080e-002  1. 6961e+000  2. 6407e-001 -7. 0117e-001  5. 7735e-001
```

hadamard(N) – створює матрицю Адамара розміром (N*N), наприклад:

```
» hadamard(4)
ans =
   1   1   1   1
   1  -1   1  -1
```

```
     1     1    -1    -1
     1    -1    -1     1
```

hilb(N) – створює матрицю Гілберта розміром (N*N), наприклад:

```
» hilb(4)
ans =
  1.0000e+000   5.0000e-001    3.3333e-001    2.5000e-001
  5.0000e-001   3.3333e-001    2.5000e-001    2.0000e-001
  3. 3333e-001  2. 5000e-001   2. 0000e-001   1. 6667e-001
  2. 5000e-001  2. 0000e-001   1. 6667e-001   1. 4286e-001
```

invhilb(N) – створює обернену матрицю Гільберта розміром (N*N), наприклад:

```
» invhilb(4)
ans =
      16          -120          240          -140
    -120          1200         -2700         1680
     240         -2700          6480         -4200
    -140          1680         -4200         2800
```

pascal(N) – створює матрицю Паскаля розміром (N*N), наприклад:

```
» pascal(5)
ans =
     1     1     1     1     1
     1     2     3     4     5
     1     3     6    10    15
     1     4    10    20    35
     1     5    15    35    70.
```

У мові Matlab передбачено декілька функцій, що дозволяють формувати матрицю на основі іншої (заданої) або, використовуючи деякий заданий вектор. До таких функцій належать:

fliplr(A) – формує матрицю, переставляючи стовпчики відомої матриці A щодо вертикальної осі, наприклад, якщо

```
A =
     1     2     3     4     5     6
     7     8     9    10    11    12
    13    14    15    16    17    18
```

то застосування цієї функції приводить до результату

```
» fliplr(A)
ans =
     6     5     4     3     2     1
    12    11    10     9     8     7
    18    17    16    15    14    13
```

flipud(A) – формує матрицю, переставляючи рядки заданої матриці A щодо горизонтальної осі, наприклад:

```
» flipud(A)
ans =
    13    14    15    16    17    18
     7     8     9    10    11    12
     1     2     3     4     5     6
```

rot90(A) – формує матрицю шляхом "повороту" заданої матриці A на 90 градусів проти годинникової стрілки:

```
» rot90(A)
ans =
   6   12   18
   5   11   17
   4   10   16
   3    9   15
   2    8   14
   1    7   13
```

reshape(A,m,n) – утворює матрицю розміром (m*n) шляхом послідовної вибірки елементів заданої матриці A по стовпчиках і наступному розподілі цих елементів по 'n' стовпчиках, кожний з яких містить 'm' елементів; при цьому число елементів матриці A повинно дорівнювати m*n, наприклад:

```
» reshape(A,2,9)
ans =
   1   13    8    3   15   10    5   17   12
   7    2   14    9    4   16   11    6   18
```

tril(A) – утворює нижню трикутну матрицю на основі матриці A шляхом онулювання її елементів вище головної діагоналі:

```
» tril(A)
ans =
   1    0    0    0    0    0
   7    8    0    0    0    0
  13   14   15    0    0    0
```

triu(A) – утворює верхню трикутну матрицю на основі матриці A шляхом онулювання її елементів нижче головної діагоналі:

```
» triu(A)
ans =
   1    2    3    4    5    6
   0    8    9   10   11   12
   0    0   15   16   17   18
```

hankel(V) – утворює квадратну матрицю Ганкеля, перший стовпчик якої збігається із заданим вектором V, наприклад:

```
>> V = [-5 6 7 4]
V =
   -5    6    7    4

» hankel(V)
ans =
   -5    6    7    4
    6    7    4    0
    7    4    0    0
    4    0    0    0
```

Процедура ***diag(x)*** – формує або витягає діагональ матриці.

Якщо ***x*** – вектор, то функція ***diag(x)*** створює квадратну матрицю з вектором ***x*** на головній діагоналі:

```
» diag(V)
ans =
   -5    0    0    0
```

```
     0     6     0     0
     0     0     7     0
     0     0     0     4
```

Щоб установити заданий вектор на іншу діагональ, при зверненні до функції необхідно зазначити ще один параметр (ціле число) – номер діагоналі (при цьому *діагоналі відлічуються від головної нагору*), наприклад:

```
» diag(V,  -1)
ans =
     0     0     0     0     0
    -5     0     0     0     0
     0     6     0     0     0
     0     0     7     0     0
     0     0     0     4     0
```

Якщо ***x*** – матриця, то функція ***diag*** створює вектор-стовпчик, що складається з елементів головної діагоналі заданої матриці ***x,*** наприклад, для матриці А, зазначеної перед прикладом застосування процедури ***fliplr***:

```
» diag(A)
ans =
     1
     8
    15
```

Якщо при цьому зазначити додатково номер діагоналі, то можна одержати вектор-стовпчик з елементів будь-якої діагоналі матриці ***x***, наприклад:

```
» diag(A,3)
ans =
     4
    11
    18
```

Функція ***zeros***(1,N) формує (створює) вектор-рядок із N нульових елементів. Аналогічно ***zeros***(N,1) створює вектор-стовпчик із N нулів.

Вектори, значення елементів яких є випадковими, рівномірно розподіленими, формуються в такий спосіб: ***rand***(1,n) – для вектора-рядка і ***rand***(m,1) – для вектора-стовпчика.

4.2. Витягання й вставляння частин матриць

Насамперед зазначимо, що *звернення до будь-якого елемента певної матриці в Matlab здійснюється шляхом указівки (у дужках, через кому) після йменння матриці двох цілих додатних чисел, що визначають відповідно номери рядка й стовпця матриці, на перетинанні яких розташований цей елемент.*

Нехай маємо деяку матрицю А:

```
>> A = [ 1 2 3 4; 5 6 7 8; 9 10 11 12]
A =
     1     2     3     4
     5     6     7     8
     9    10    11    12
```

Тоді одержати значення елемента цієї матриці, розташованого на перетинанні другого рядка із третім стовпчиком, можна в такий спосіб:

```
>> A(2,3)
ans =     7
```

Якщо потрібно, навпаки, встановити на це місце деяке число, наприклад, π, то це можна зробити так:

```
>> A(2, 3) = pi; A
A =
   1.0000    2.0000    3.0000    4.0000
   5. 0000   6. 0000   3. 1416   8. 0000
   9. 0000  10. 0000  11. 0000  12. 0000
```

Іноді потрібно створити меншу матрицю з більшої, формуючи її шляхом витягання з останньої матриці елементів її кількох рядків і стовпчиків, або, навпаки, вставити меншу матрицю таким чином, щоб вона стала певною частиною матриці більшого розміру. Це в Matlab робиться за допомогою знака двокрапки (" : ").

Розглянемо ці операції на прикладах.

Нехай потрібно створити вектор V1, що складається з елементів третього стовпчика попередньої матриці А. Для цього зробимо такі дії:

```
>> V1 = A(:, 3)
V1 =
    3.0000
   3. 1416
  11. 0000
```

Щоб створити вектор V2, який складається з елементів другого рядка матриці А, роблять так:

```
>> V2 = A(2, : )
V2 =   5. 0000   6. 0000   3. 1416   8. 0000
```

Припустимо, що необхідно з матриці А утворити матрицю В розміром (2*2), яка складається з елементів лівого нижнього рогу матриці А. Тоді роблять таке:

```
>> B = A(2:3, 1:2)
B =
   5    6
   9   10
```

Аналогічно можна вставити матрицю В в верхню середину матриці А:

```
>> A(1:2,2:3)=B
A =
   1    5    6    4
   5    9   10    8
   9   10   11   12
```

Як очевидно, для цього *замість указівки номерів елементів матриці можна вказувати діапазон змінювання цих номерів шляхом указівки нижньої й верхньої меж, розділяючи їх двокрапкою*.

Примітка. *Якщо верхньою межею змінювання номерів елементів матриці є її розмір у цьому вимірі, замість нього можна використовувати службове слово* ***end.*** Наприклад:

```
>> A(2:end,2:end)
ans =
   9   10    8
  10   11   12
```

Ці операції дуже зручні для формування матриць, більшість елементів яких однакові, зокрема, так званих розріджених матриць, що складаються, в основному, із нулів, за винятком окремих елементів. Для прикладу розглянемо формування розрідженої матриці розміром (5*7) з одиничними елементами в її центрі:

```
>> A = zeros(5,7);
>> B = ones(3,3);
```

```
>> A(2:4,3:5)=B
A =
   0   0   0   0   0   0   0
   0   0   1   1   1   0   0
   0   0   1   1   1   0   0
   0   0   1   1   1   0   0
   0   0   0   0   0   0   0
```

"Розтягнути" матрицю (A) *у єдиний* вектор (V) можна за допомогою звичайного запису *"V = A(:)".* При цьому створюється вектор-стовпчик із кількістю елементів (m*n), у якому стовпчики заданої матриці розміщені зверху униз у порядку самих стовпчиків:

```
» A = [1 2 3; 4 5 6]
A =
   1   2   3
   4   5   6
» v = A(☺
v =
   1
   4
   2
   5
   3
   6
```

Нарешті, "розширювати" матрицю, укладаючи її з окремих заданих матриць ("блоків") можна теж досить просто. Якщо задані кілька матриць-блоків A1, A2,... AN з однаковою кількістю рядків, то з них можна "зліпити" єдину матрицю A, об'єднуючи блоки в один "рядок" у такий спосіб:

A = [A1, A2,... , AN].

Цю операцію називають *горизонтальною конкатенацією* (зчепленням) матриць. Аналогічно, *вертикальна конкатенація* матриць реалізується (за умови, що всі складові блоки-матриці мають однакову кількість стовпчиків) аналогічним чином, шляхом застосовування для відділяння блоків замість ко-ми крапки з комою:

A = [A1; A2;... ; AN].

Наведемо приклади. Приклад горизонтальної конкатенації :

```
>> A1 =  [1 2 3; 4 5 6; 7 8 9];
>> A2 = [10;11;12];
>> A3 = [14 15; 16 17; 18 19];
>> A = [A1, A2, A3]
A =
   1   2   3  10  14  15
   4   5   6  11  16  17
   7   8   9  12  18  19
```

Приклад вертикальної конкатенації:

```
>> B1 = [1 2 3 4 5];
>> B2 = [ 6 7 8 9 10; 11 12 13 14 15];
>> B3 = [17 18 19 20 21];
>> B = [ B1; B2; B3]
B =
    1    2    3    4    5
    6    7    8    9   10
   11   12   13   14   15
   17   18   19   20   21
```

4.3. Обробка даних вимірів

Система Matlab дає користувачеві додаткові можливості для обробки даних, що задані у векторній або матричній формі.

Припустимо, що є деяка залежність ***y(x),*** яка задана рядом точок

x	2	4	6	8	10
y	5.5	6.3	6.8	8	8.6

Її можна задати в командному вікні Matlab як матрицю **xydata**, що містить два рядки – значення ***x*** і значення ***y***:

```
>> xydata =[2 4 6 8 10; 5.5 6.3 6.8 8 8.6]
xydata =
   2. 0000   4. 0000   6. 0000   8. 0000  10. 0000
   5. 5000   6. 3000   6. 8000   8. 0000   8. 6000
```

На прикладі цієї залежності розглянемо основні засоби для обробки даних.

Функція ***size***(***xydata***) призначена для визначення числа рядків і стовпчиків матриці ***xydata***. Вона формує вектор [n, p], який містить ці величини:

```
>> size(xydata)
ans =
    2    5
```

Звернення до неї виду

```
>> [n, p] = size(xydata);
```

дозволяє зберегти в пам'яті машини і використовувати потім при подальших обчисленнях дані про кількість рядків ***n*** і кількість стовпчиків ***p*** цієї матриці:

```
>> n, p
n =
    2
p =
    5
```

За допомогою цієї функції можна встановити довжину й тип (рядок або стовпчик) вектора :

```
» v = xydata(☺
v =
   2.0000
   5.5000
   4.0000
   6.3000
   6.0000
   6.8000
   8.0000
   8.0000
  10. 0000
   8. 6000
» n = size(v)
n =   10    1
» v1 = v'
v1 =
  Columns 1 through 7
   2. 0000   5. 5000   4. 0000   6. 3000   6. 0000   6. 8000   8. 0000
  Columns 8 through 10
   8. 0000  10. 0000   8. 6000
»  size(v')
ans =    1   10
```

Функція ***max***(V), де V – деякий вектор, видає значення максимального елемента цього вектора. Аналогічно, функція ***min***(V) витягає мінімальний елемент вектора V. Функції ***mean***(V) і ***std***(V) визначають, відповідно, середнє значення і середньоквадратичне відхилення від нього значень елементів вектора V.

Функція сортування ***sort***(V) формує вектор, елементи якого розподілені в порядку зростання їх значень.

Функція ***sum***(V) обчислює суму елементів вектора V.

Функція ***prod***(V) видає добуток усіх елементів вектора V.

Функція ***cumsum***(V) формує вектор того ж типу й розміру, будь-який елемент якого є сумою всіх попередніх елементів вектора V (вектор кумулятивної суми).

Функція ***cumprod***(V) створює вектор, елементи якого є добутком усіх попередніх елементів вектора V.

Функція ***diff***(V) видає вектор, що має розмір на одиницю менший за розмір вектора V, елементи якого є різницею між суміжними елементами вектора V.

Застосування описаних функцій проілюстровано нижче.

```
» v = [ 1, 0.1, 0.5, 0.1, 0.1,0.4 ];
» disp(size(v))
     5.□.  6
» disp(max(v))
    1
» disp(min(v))
   0. 1000
» disp(mean(v))
   0. 3667
» disp(std(v))
   0. 3559
» disp(sort(v))
   0. 1000   0. 1000   0. 1000   0. 4000   0. 5000   1. 0000
» disp(sum(v))
   2. 2000
» disp(prod(v))
  2. 0000e-004
» disp(cumsum(v))
   1. 0000   1. 1000   1. 6000   1. 7000   1. 8000   2. 2000
» disp(cumprod(v))
   1. 0000   0. 1000   0. 0500   0. 0050   0. 0005   0. 0002
» disp(diff(v))
  -0. 9000   0. 4000  -0. 4000        0   0. 3000
```

Якщо вказати другий вихідний параметр, то можна одержати додаткову інформацію про індекс першого елемента, значення якого є максимальним або мінімальним:

```
>> [M,n]=max(v)
M =    1
n =    1
>> [N,m]=min(v)
N =   0.1000
m =    2
```

Інтегрування методом трапецій здійснює процедура ***trapz***. Звернення до неї вигляду ***trapz***(x,y) призводить до обчислення площі під графіком

функції y(x), у якому всі точки, задані векторами x і y, з'єднані відрізками прямих. Якщо перший вектор x не зазначений у зверненні, за замовчуванням припускається, що крок інтегрування дорівнює одиниці (тобто вектор x є вектором із номерів елементів вектора y).

Приклад. Обчислимо інтеграл від функції y = sin(x) у діапазоні від 0 до π. Його точне значення дорівнює 2. Візьмемо рівномірну сітку із 100 елементів. Тоді обчислення зведуться до сукупності операцій:

```
» x = 0 : pi/100 : pi;
» y = sin(x);
>> disp(trapz(x,y))
   1.9998
```

Ті самі функції ***size, max, min, mean, std, sort, sum, prod, cumsum, cumprod, diff*** можуть бути застосовані і до матриць**.** Основною відмінністю використання як аргументів цих функцій саме матриць є те, що відповідні описані вище операції провадяться не по відношенню до рядків матриць, а до кожного зі стовпців заданої матриці. Тобто кожний стовпець матриці A розглядається як змінна, а кожний рядок – як окреме спостереження. Так, у результаті застосування функцій ***max, min, mean, std*** утворюються вектори-рядки з кількістю елементів, яка дорівнює кількості стовпців заданої матриці. Кожний елемент містить, відповідно, максимальне, мінімальне, середнє або середньоквадратичне значення елементів відповідного стовпця заданої матриці.

Наведемо приклади. Нехай маємо 3 величини y1, y2 і y3, що виміряні за деяких п'яти значень аргументу (які не зазначені). Тоді дані вимірів утворять 3 вектори по 5 елементів:

```
>> y1 = [ 5.5 6.3 6.8 8 8.6];
>>  y2 = [-1. 2 0.5 -0. 6 1 0.1];
>>  y3 = [ 3.4 5.6 0 8.4 10.3] ;  .
```

Сформуємо з них матрицю вимірів так, щоб вектори y1, y2, y3 утворювали стовпці цієї матриці:

```
» A = [ y1', y2', y3']
A =
   5.5000   -1.2000    3.4000
   6.3000    0.5000    5.6000
   6.8000   -0.6000    0
   8. 0000   1. 0000   8. 4000
   8. 6000   0. 1000   10. 3000
```

Застосуємо до цієї матриці вимірів описані вище функції. Одержимо

```
» size(A)
ans =    5    3
» max(A)
ans =   8. 6000   1. 0000   10. 3000
» min(A)
ans =   5. 5000   -1. 2000        0
» mean(A)
ans =   7. 0400   -0. 0400   5. 5400
   » std(A)
   ans =   1. 2582   0. 8735   4. 0655
```

*Якщо при зверненні до функцій **max** і **min** зазначити другий вихідний параметр, то він дасть інформацію про номер рядка, де знаходиться у відповідному стовпчику перший елемент із максимальним (або мінімальним) значенням*. Наприклад:

```
>> [M,n]=max(A)
M =    8.6000 1.0000 10.3000
n =    5    4    5
>> [N,m]=min(A)
N =    5.5000 -1.2000 0
m =    1    1    3
```

Функція ***sort*** сортує елементи кожного зі стовпців матриці. Результатом є матриця того ж розміру.

Функція ***sum*** і ***prod*** формують вектор-рядок, кожний елемент якого є сумою або добутком елементів відповідного стовпця початкової матриці.

Функції ***cumsum, cumprod*** утворять матриці того самого розміру, елементи кожного стовпця яких є сумою або добутком елементів цього ж стовпця початкової матриці, починаючи з відповідного елемента і вище.

Нарешті, функція ***diff*** створює із заданої матриці розміром (m*n) матрицю розміром ((m-1)*n), елементи якої є різницею між елементами суміжних рядків початкової матриці.

Застосовуючи ці процедури до тієї самої матриці вимірів, одержимо:

```
» sort(A)
ans =
   5.5000   -1.2000    0
   6.3000   -0.6000   3.4000
   6.8000    0.1000    5.6000
   8. 0000   0. 5000   8. 4000
   8. 6000   1. 0000  10. 3000
» sum(A)
ans =  35. 2000  -0. 2000  27. 7000
» prod(A)
ans =  1. 0e+004 *
   1. 6211   0. 0000       0
» cumsum(A)
ans =
   5.5000   -1.2000    3.4000
  11.8000   -0.7000    9.0000
  18.6000   -1.3000    9.0000
  26. 6000  -0. 3000  17. 4000
  35. 2000  -0. 2000  27. 7000
» cumprod(A)
ans =  1.0e+004 *
   0.0006   -0.0001    0.0003
   0.0035   -0.0001    0.0019
   0.0236    0.0000       0
   0. 1885   0. 0000       0
   1. 6211   0. 0000       0
» diff(A)
ans =
   0.8000    1.7000    2.2000
   0. 5000  -1. 1000  -5. 6000
   1. 2000   1. 6000   8. 4000
   0.6000  -0. 9000   1. 9000
```

Розглянемо деякі інші функції, надані користувачеві системою Matlab.

Функція ***cov***(A) обчислює *матрицю коваріацій* вимірів. При цьому утворюється квадратна симетрична матриця з кількістю рядків і стовпчиків, рівним кількості виміряних величин, тобто кількості стовпчиків матриці вимірів.

Наприклад, при застосуванні до прийнятої матриці вимірів вона дає такий результат:

```
» cov(A)
ans =
   1. 5830   0. 6845    3. 6880
   0. 6845   0. 7630    2. 3145
   3.6880    2. 3145   16. 5280
```

На діагоналі матриці коваріацій розміщені *дисперсії* виміряних величин, а поза нею – *взаємні кореляційні моменти* цих величин.

Функція ***corrcoeff***(A) обчислює *матрицю коефіцієнтів кореляції* за тих самих умов. Елементи матриці S = corrcoef(A) пов'язані з елементами матриці коваріацій C=cov(A) таким співвідношенням:

$$S(k,l) = \frac{C(k,l)}{\sqrt{C(k,k)\cdot C(l,l)}}$$

Приклад:

```
» corrcoef(A)
ans =
   1. 0000   0. 6228   0. 7210
   0. 6228   1. 0000   0. 6518
   0.7210    0. 6518   1. 0000
```

4.4. Поелементне перетворення матриць

Для поелементного перетворення матриці придатні всі зазначені раніше алгебричні функції. Кожна така функція формує матрицю того самого розміру, що й задана, кожний елемент якої обчислюється як зазначена функція від відповідного елемента заданої матриці. Крім цього, у Matlab визначені операції ***поелементного множення*** матриць однакового розміру (сполученням ".*", що записується між іменами матриць, що перемножуються), ***поелементного ділення*** (сполучення "./" і ".\"), ***поелементного піднесення до степеня*** (сполучення ".^"), коли кожний елемент першої матриці підноситься до степеня, який дорівнює значенню відповідного елемента другої матриці.

Наведемо кілька прикладів:

```
» A = [1,2,3,4,5; -2, 3, 1, 4, 0]
A =
    1    2    3    4    5
   -2    3    1    4    0
» B = [-1,3,5,-2,1; 1,8,-3,-1,2]
B =
   -1    3    5   -2    1
    1    8   -3   -1    2
» sin(A)
ans =
   0. 8415   0. 9093   0. 1411  -0. 7568  -0. 9589
  -0. 9093   0. 1411   0. 8415  -0. 7568         0
» A . * B
ans =
   -1    6   15   -8    5
   -2   24   -3   -4    0
» A . / B
ans =
  -1. 0000   0. 6667   0. 6000  -2. 0000   5. 0000
```

```
  -2. 0000    0. 3750   -0. 3333   -4. 0000         0
» A . \ B
Warning: Divide by zero
ans =
  -1. 0000    1. 5000    1. 6667   -0. 5000    0. 2000
  -0. 5000    2. 6667   -3. 0000   -0. 2500       Inf
» A . ^ B
ans =
  1. 0e+003 *
   0. 0010    0. 0080    0. 2430    0. 0001    0. 0050
  -0. 0020    6. 5610    0. 0010    0. 0002         0
```

Оригінальною в мові Matlab є операція *додавання до матриці числа*. Вона записується в такий спосіб: A + x, або x + A (A – матриця, а x – число). Такої операції немає в математиці. У Matlab вона є еквівалентною до сукупності операцій

$$A + x * E,$$

де E – позначення матриці, що складається саме з одиниць, тих самих розмірів, що і матриця A. Наприклад:

```
» A = [ 1 2 3 4 5; 6 7 8 9 11 ]
A =
    1    2    3    4    5
    6    7    8    9   11
» A + 2
ans =
    3    4    5    6    7
    8    9   10   11   13
» 2 + A
ans =
    3    4    5    6    7
    8    9   10   11   13
```

4.5. Матричні дії над матрицями

До матричних дій над матрицями відносять такі операції, які використовуються в матричному зчисленні в математиці і не суперечать йому.

Базові дії з матрицями – ***додавання, віднімання, транспонування, множення матриці на число, множення матриці на матрицю, піднесення матриці до цілого степеня*** – здійснюються в мові Matlab за допомогою звичайних знаків арифметичних операцій. При використовуванні цих операцій *важливо пам'ятати* умови, за яких ці операції є можливими:

при додаванні або відніманні матриць вони повинні мати однакові розміри;

при множенні матриць кількість стовпчиків першої матриці повинна збігатися з кількістю рядків другої матриці.

Невиконання цих умов призведе до появи в командному вікні повідомлення про помилку. Наведемо кілька прикладів.

Приклад додавання й віднімання:

```
» A = [ 1 2 3 4 5; 6 7 8 9 11 ]
A =
   1    2    3    4    5
   6    7    8    9   11
» B = [ 0 -1 -2 -3 -4; 5 6 7 8 9 ]
```

```
B =
    0   -1   -2   -3   -4
    5    6    7    8    9
» A + B
ans =
    1     1     1     1     1
   11    13    15    17    20
» A – B
ans =
    1    3    5    7    9
    1    1    1    1    2.
```

Приклад множення на число:

```
» 5*A
ans =
    5   10   15   20   25
   30   35   40   45   55
» A*5
ans =
    5   10   15   20   25
    30   35   40   45   55.
```

Приклад транспонування матриці:

```
» A'
ans =
    1   6
    2   7
    3   8
    4   9
    5   11.
```

Приклад множення матриці на матрицю:

```
» A' * B
ans =
   30   35   40   45   50
   35   40   45   50   55
   40   45   50   55   60
   45   50   55   60   65
   55   61   67   73   79
» C = A * B'
C =
  -40   115
  -94   299.
```

Функція ***обернення матриці – inv***(A) – обчисляє матрицю, обернену до заданої матриці А. Початкова матриця А має бути квадратною, а її визначник – не дорівнювати нулю.

Наведемо приклад:

```
» inv©
ans =
 -2. 6000e-001  1. 0000e-001
 -8. 1739e-002  3. 4783e-002
```

Перевіримо слушність виконання операції обернення, застосовуючи її ще раз до отриманого результату:

```
» inv(ans)
ans =
 -4. 0000e+001  1. 1500e+002
 -9. 4000e+001  2. 9900e+002
```

Як бачимо, одержано початкову матрицю С, що є ознакою правильності виконання обернення матриці.

Піднесення матриці до цілого степеня здійснюється в Matlab за допомогою знака " ^ ": A^***n***. При цьому матриця має бути квадратною, а ***n*** має бути цілим (додатним або від'ємним) числом. Ця матрична дія є еквівалентною до множення матриці A на себе ***n*** разів (якщо ***n*** – додатне) або множенню оберненої матриці на себе (при ***n*** від'ємному).

Наведемо приклад:

```
» A^2
ans =
    8    -3   -10
   -5    10    16
   -2     4     9
» A^(-2)
ans =
  1.5385e-001  -7.6923e-002   3.0769e-001
  7. 6923e-002  3. 0769e-001 -4. 6154e-001
  2. 1328e-018 -1. 5385e-001  3. 8462e-001
```

Дуже оригінальними в мові Matlab є дві нові, невідомі в математиці функції ***ділення матриць***. При цьому вводяться поняття ***ділення матриць зліва направо*** і ***ділення матриць справа наліво***. Перша операція записується за допомогою знаку "/" , а друга – "\", які розташовуються між іменами матриць, які діляться.

Операція B/A еквівалентна послідовності дій B****inv***(A), де функція ***inv*** здійснює *обернення матриці.* Її зручно використовувати для роз'язування матричного рівняння:

$$X*A = B.$$

Аналогічно операція A\B рівносильна сукупності операцій ***inv***(A)*B, що є розв'язком матричного рівняння:

$$A*X = B.$$

4.6. Розв'язування систем лінійних алгебричних рівнянь

Система лінійних алгебричних рівнянь (СЛАР) n-го порядку має вигляд:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots a_{2n}x_n = b_2 \\ \ldots \quad \ldots \quad \ldots \quad \ldots = \ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots a_{nn}x_n = b_n \end{cases}$$

Тут позначено: x_i $(i = 1,2,\ldots n)$ – деякі змінні, a_{ij} $(i, j = 1,2,\ldots n)$ - коефіцієнти при змінних, b_i $(i = 1,2,\ldots n)$ – так звані "вільні" члени.

Під розв'язуванням СЛАР розуміється відшукування таких значень змінних x_i, підставляння яких у кожне з n рівнянь, перетворює їх одночасно у тотожності.

Якщо використати матричні позначення:

$$A=\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix};\ B=\begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix};\ X=\begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix},$$

то СЛАР може бути поданою у матричній формі у такий спосіб:

$$AX = B. \qquad (1)$$

У системі Matlab розв'язування рівняння (1) здійснюється вельми просто, з використанням дії зворотного ділення. Для прикладу розглянемо задачу відшукування коренів системи лінійних алгебричних рівнянь:

$$x_1 + 2x_2 + 3x_3 = 14$$
$$2x_1 - x_2 - 5x_3 = -15$$
$$x_1 - x_2 - x_3 = -4$$

Це можна зробити у такий спосіб:

```
» A = [ 1 2 3; 2 -1 -5; 1 -1 -1]
A =
    1    2    3
    2   -1   -5
    1   -1   -1
» B = [ 14;-15;-4]
B =
   14
  -15
   -4
» x = A \ B
x =
    1
    2
    3
```

Розглянутий метод є точним. Точні (або прямі) методи працюють достатньо швидко і широко застосовуються на практиці, якщо є достатніми обсяги пам'яті для їх реалізації.

Але існують й інші шляхи відшукування коренів СЛАР, які відносяться до наближених. Наприклад, різні модифікації *методу ітерацій*. Вони знаходять за скінченну кількість кроків (ітерацій) лише наближені розв'язки із заданою припустимою відносною похибкою.

Метод ітерацій у загальному випадку полягає у тому, що попередньо вихідне рівняння $f(x) = 0$ перетворюється до виду

$$x = \varphi(x). \qquad (2)$$

З області ізоляції $[a, b]$ шуканого кореня обирається деяке певне значення x_0 аргументу, яке приймається за початкове наближення кореня. Наближені значення кореня у наступних наближеннях визначаються з співвідношення

$$x_k = \varphi(x_{k-1}); \qquad (k = 1, 2, \cdots n), \qquad (3)$$

при цьому k має зміст номера ітерації (наближення).

Очевидно, ітераційний процес може приводити до послідовності значень, які наближаються до деякого значення аргумента (шуканого кореня), і тоді цей процес є стійким. Але такий процес може приводити і до послідовності значень

аргумента, які все далі віддаляються від шуканого значення, тобто бути нестійким. Стійкість або нестійкість ітераційного процеса суттєво залежить від виду залежності $\varphi(x)$, яка, як неважко впевнитися, визначається неоднозначно, і тому можна підібрати з можливих її варіантів і такі, які забезпечують стійкість ітерацій. Саме це й є найбільш складним у побудові ітераційного процесу.

Методи ітерацій можуть бути застосовані і для відшукування коренів СЛАР, якщо попередньо систему (1) подати у виді

$$\begin{cases} x_1 = \varphi_1(x_1, x_2, ..., x_n) \\ x_2 = \varphi_2(x_1, x_2, ..., x_n) \\ \quad ..... \\ x_n = \varphi_n(x_1, x_2, ..., x_n) \end{cases} \tag{4}$$

MatLAB має кілька вбудованих процедур, які дозволяють розв'язувати систему лінійних алгебричних рівнянь виду (1) наближеними ітераційними методами. Їх застосовують при обчисленнях тоді, коли матриці коефіцієнтів СЛАР є розрідженими і великими за розмірами. До них відносяться:

bicg – метод біспряжених градієнтів;

bicgstab – стабілізований метод біспряжених градієнтів;

cgs – квадратичний метод спряжених градієнтів;

gmres – узагальнений метод мінімального відхилу;

qmr – метод квазімінімального відхилу;

pcg – передобумовлений метод спряжених градієнтів (застосовується лише для симетричних матриць A).

Загальне звернення до цих процедур має вигляд

x = ***bicg***(A,B,tol,maxit).

Тут A – квадратна матриця розміром $(n \times n)$ коефіцієнтів при аргументах системи рівнянь (1), B – матриця-стовпець розміром $(n \times 1)$ вільних членів, tol – допустима межова відносна похибка визначення коренів, maxit – межова допустима кількість ітерацій, x – вектор отриманих наближених значень коренів рівняння (1).

За початкове наближення обирається вектор x0 з нульових елементів.

Наведемо приклад. Введемо наступну послідовність операторів:

```
A=[1 1 2 3;3 -1 -1 -2;2 3 -1 -1; 1 2 3 -1]
B=[1; -4; -6; -4]
X=A\B;
X1=bicg(A,B);
X2=bicgstab(A,B);
X3=cgs(A,B);
X4=gmres(A,B);
X5=qmr(A,B);
disp(' ')
disp(' ')
disp(['   Точно  ','   BICG  ','  BICGSTAB ','   CGs  ','   GMRES  ','   QMR ']) 
Y=[X X1 X2 X3 X4 X5]
disp(' ')
disp('                П О Х И Б К И')
disp(['    BICG  ','  BICGSTAB ','   CGs  ','   GMRES  ','   QMR ']) 
dY=[X1-X X2-X X3-X X4-X X5-X]
```

Виконавши їх, система MATLAB виведе результат у наступному вигляді:

```
A =
    1    1    2    3
    3   -1   -1   -2
    2    3   -1   -1
    1    2    3   -1
B =
    1
   -4
   -6
   -4

    Точно           BICG         BICGSTAB         CGS          GMRES         QMR
Y =
 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000
 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000
          0    3.5638e-014  3.8956e-011 -5.8953e-014 -1.6653e-016 -4.1078e-015
  1.0000e+000  1.0000e+000  1.0000e+000  1.0000e+000  1.0000e+000  1.0000e+000

                П О Х И Б К И
     BICG          BICGSTAB        CGs           GMRES          QMR
dY =
  1.0436e-014  -4.6485e-012  -7.4163e-014   2.2204e-016  -3.1530e-014
 -1.7319e-014   2.6083e-011   2.5535e-015   4.4409e-016   8.5487e-015
  3.5638e-014   3.8956e-011  -5.8953e-014  -1.6653e-016  -4.1078e-015
  4.8406e-014   2.0103e-011  -2.8422e-014         0      -2.0428e-014
```

4.7. Матричні функції

Окрім вищезазначених, у MatLAB передбачені ще такі матричні функції й операції.

Операція ***транспонування матриці*** здійснюється за допомогою знака апострофа, записаного після ймення матриці, наприклад:

```
X=[1 2 3; 4 5 6]
X =
    1    2    3
    4    5    6
X'
ans =
    1    4
    2    5
    3    6
```

Обчислення ***матричної експоненти*** (e^A) здійснюється за допомогою функцій ***expm, expm1, expm2, expm3.*** Ці функції варто відрізняти від раніше розглянутої функції ***exp***(A), яка формує матрицю, кожний елемент якої дорівнює ***e*** в степені, що дорівнює відповідному елементу матриці A.

Функція ***expm*** є вмонтованою функцією Matlab. Функція ***expm1***(A) є M-файлом, який обчислює матричну експоненту шляхом використання розкладення Паде матриці A. Функція ***expm2***(A) обчислює матричну експоненту, використовуючи розкладення Тейлора матриці A. Функція

expm3(A) обчислює матричну експоненту на основі використання спектрального розкладу A.

Наведемо приклади використання цих функцій:

```
» A = [1,2,3; 0, -1,5;7, -4,1]
A =
    1    2    3
    0   -1    5
    7   -4    1
» expm(A)
ans =
  131.3648    -9.5601    80.6685
   97. 8030   -7. 1768   59. 9309
  123. 0245   -8. 8236   75. 4773

» expm1(A)
ans =
  131.3648    -9.5601    80.6685
   97. 8030   -7. 1768   59. 9309
  123. 0245   -8. 8236   75. 4773

» expm2(A)
ans =
  131.3648    -9.5601    80.6685
   97. 8030   -7. 1768   59. 9309
  123. 0245   -8. 8236   75. 4773

» expm3(A)
ans =
  1.0e+002 *
   1. 3136 + 0. 0000i      -0. 0956 + 0. 0000i       0. 8067 – 0. 0000i
   0. 9780 + 0. 0000i      -0. 0718 – 0. 0000i       0. 5993 – 0. 0000i
    1.2302 + 0. 0000i      -0. 0882 – 0. 0000i       0. 7548 – 0. 0000i
```

Функція ***logm***(A) робить обернену операцію – логарифмування матриці за натуральною основою, наприклад:

```
 A =
    1    2    3
    0    1    5
    7    4    1
» B = expm3(A)
B =
  1.0e+003 *
   0.9378    0.7987    0.9547
   1. 0643   0. 9074   1. 0844
   1. 5182   1. 2932   1. 5459

» logm(B)
ans =
   1. 0000   2. 0000   3. 0000
   0. 0000   1. 0000   5. 0000
   7.0000    4. 0000   1. 0000
```

Функція ***sqrtm***(A) обчислює таку матрицю Y, що Y*Y = A:

```
» Y = sqrtm(A)
Y =
   0.7884 + 0.8806i       0.6717 – 0.1795i       0.8029 – 0.4180i
  0. 8953 + 0. 6508i      0. 7628 + 0. 8620i      0. 9118 – 1. 0066i
```

```
   1.2765 – 1. 4092i        1. 0875 – 0. 5449i        1. 3000 + 1. 2525i

» Y * Y
ans =
   1. 0000 + 0. 0000i        2. 0000 – 0. 0000i        3. 0000 + 0. 0000i
   0. 0000 – 0. 0000i        1. 0000 – 0. 0000i        5. 0000 – 0. 0000i
   7.0000 + 0. 0000i         4. 0000 + 0. 0000i        1. 0000 + 0. 0000i
```

4.8. Функції лінійної алгебри

Традиційно до лінійної алгебри відносять такі задачі, як обернення і псевдообернення матриці, спектральне й сингулярне розкладання матриць, обчислення власних значень і векторів, сингулярних чисел матриць, обчислення функцій від матриць. Коротко ознайомимося з деякими основними функціями Matlab у цій області.

Функція ***k = cond***(A) обчислює й видає число обумовленості матриці стосовно операції обернення, яке дорівнює відношенню максимального сингулярного числа матриці до мінімального.

Функція ***k = norm***(v,p) обчислює p-норму вектора v за формулою:

k = ***sum***(***abs***(v) . ^p)^(1/p),

де p – ціле додатне число. Якщо аргумент p при зверненні до функції не зазначений, обчислюється 2-норма.

Функція ***k = norm***(A,p) обчислює p-норму матриці A за формулою:

k = ***max(sum***(***abs***(A) . ^p))^(1/p),

де p = 1,2, 'fro' або inf. Якщо аргумент p не зазначений, обчислюється 2-норма. При цьому є слушними співвідношення:

norm(A,1) = max(sum(abs(A)));

norm(A,inf) = max(sum(abs(A')));

*norm(A,'fro') = sqrt(sum(diag(A'*A)));*

$norm(A) = norm(A,2) = \sigma_{max}(A)$.

Функція ***rd = rcond***(A) обчислює величину, обернену значенню числа обумовленості матриці A щодо 1-норми. Якщо матриця A добре обумовлена, значення ***rd*** близько до одиниці. Якщо ж вона погано обумовлена, ***rd*** наближається до нуля.

Функція ***r = rank***(A) обчислює ранг матриці, який визначається як кількість сингулярних чисел матриці, що перевищують поріг

*max(size(A))*norm(A)*eps.*

Наведемо приклади застосування цих функцій:

```
A =
    1    2    3
    0    1    5
    7    4    1
» disp(cond(A))
   13. 8032
» disp(norm(A,1))
    9
» disp(norm(A))
   8. 6950
» disp(rcond(A))
```

```
5.4645e-002
» disp(rank(A))
   3
```

Процедура ***d* = *det***(A) обчислює *визначник квадратної матриці* на основі трикутного розкладання методом виключення Гаусса.

Функція ***t* = *trace***(A) обчислює *слід матриці* A, який дорівнює сумі її діагональних елементів.

Q* = *null(A) обчислює ортонормований *базис нуль-простору* матриці A.

Q* = *orth(A) видає *ортонормований базис матриці* A.

Процедура R = ***rref***(A) здійснює *приведення матриці до трикутного виду на основі методу виключення Гаусса* з частковим вибором провідного елемента.

Приклади:

```
» disp(det(A))
   30
» disp(trace(A))
    3
» disp(null(A))
» disp(orth(A))
   0. 3395    0. 4082   -0. 8474
   0. 2793    0. 8165    0. 5053
   0.8982   -0. 4082    0. 1632
» disp(rref(A))
                  1    0    0
                  0    1    0
                  0    0    1
```

Функція R=***chol***(A) здійснює ***розкладання Холецького*** для дійсних симетричних і комплексних ермітових матриць. Наприклад:

```
» A = [ 1 2 3; 2 15 8; 3 8 400]
A =
    1    2    3
    2   15    8
    3    8  400
» disp(chol(A))
   1. 0000    2. 0000    3. 0000
      0       3. 3166    0. 6030
      0          0      19. 7645
```

Функція ***lu***(A) здійснює ***LU-розкладання*** матриці A в виді добутку нижньої трикутної матриці L (можливо, із перестановками) і верхньої трикутної матриці U так, що A = L * U.

Звернення до цієї функції виду

[L, U, P] = ***lu***(A)

дозволяє одержати три складові цього розкладання – нижню трикутну матрицю L, верхню трикутну U і матрицю перестановок P такі, що

P * A =L * U.

Наведемо приклад:

```
A =
    1    2    3
    2   15    8
    3    8  400
» disp(lu(A))
   3.0000     8.0000  400.0000
  -0. 6667    9. 6667 -258. 6667
  -0. 3333    0. 0690 -148. 1724
» [ L, U, P] = lu(A);
```

```
» L
L =
   1.0000        0           0
   0. 6667   1. 0000        0
   0. 3333  -0. 0690   1. 0000
» U
U =
   3.0000  8.0000   400.0000
      0    9. 6667 -258. 6667
      0       0 -148. 1724
» P
P =
    0    0    1
    0    1    0
    1    0    0
```

З нього випливає, що в першому, спрощеному варіанті звернення функція видає комбінацію з матриць L і U.

Обернення матриці здійснюється за допомогою функції ***inv***(A):

```
» disp(inv(A))
   1. 3814   -0. 1806   -0. 0067
  -0. 1806    0. 0910   -0. 0005
  -0. 0067   -0. 0005    0. 0026
```

Процедура ***pinv***(A) знаходить матрицю, *псевдообернену* матриці A, яка має розміри матриці A^T і задовольняє умови

A * P * A = A; P * A * P = P.

Наприклад:

```
A =
    1    2    3    4    5
    5   -1    4    6    0
» P = pinv(A)
P =
  -0.0423    0.0852
   0.0704   -0.0480
   0.0282    0.0372
   0. 0282   0. 0628
   0. 1408  -0. 0704
» A*P*A,                  % перевірка 1
ans =
   1. 0000   2. 0000   3. 0000   4. 0000   5. 0000
   5. 0000  -1. 0000   4. 0000   6. 0000   0. 0000
» P*A*P                   % перевірка 2
ans =
  -0.0423    0.0852
   0.0704   -0.0480
   0. 0282   0. 0372
   0. 0282   0. 0628
   0.1408  -0. 0704
```

Для квадратних матриць ця операція рівнозначна звичайному оберненню.

Процедура [Q, R, P] = ***qr***(A) здійснює розкладення матриці A на три – унітарну матрицю Q, верхню трикутну R із діагональними елементами, що зменшуються за модулем, і матрицю перестановок P такі що

A * P = Q * R.

Наприклад:

```
A =
    1    2    3    4    5
    5   -1    4    6    0
```

```
» [Q,R,P] = qr(A)
Q =
  -0.5547 -0.8321
  -0.8321 0.5547
R =
  -7.2111 -2.7735 -4.9923 -4.7150   -0.2774
       0    -4.1603 -0.2774  1.9415    -2.2188
P =
    0   0     0    1    0
    0    0    0    0    1
    0    0    1    0    0
    1    0    0    0    0
    0    1    0    0    0
```

Визначення характеристичного полінома матриці *A* можна здійснити за допомогою функції ***poly***(A). Звернення до неї виду *p=**poly**(A)* дає можливість знайти вектор-рядок *p* коефіцієнтів характеристичного полінома

$p(s) = det(s*E - A) = p_1*s^n + ... + p_n*s + p_{n+1}$,

де *E* – позначення одиничної матриці розміром (n*n). Наприклад :

```
» A = [1 2 3; 5 6 0; -1 2 3]
A =
    1    2    3
    5    6    0
   -1    2    3
» p = poly(A)
p =
   1. 0000  -10. 0000   20. 0000  -36. 0000
```

Обчислення власних значень і власних векторів матриці здійснює процедура ***eig***(A). Звичайне звернення до неї дозволяє одержати вектор власних значень матриці A, тобто коренів характеристичного полінома матриці. Якщо ж звернення має вид:

[R, D] = ***eig***(A),

то в результаті одержують діагональну матрицю D власних значень і матрицю R правих власних векторів, що задовольняють умові

A*R = R*D.

Ці вектори є внормованими у такий спосіб, що норма кожного з них дорівнює одиниці. Наведемо приклад:

```
A =
    1    2     3
   -1    8    16
   -5  100     3
» disp(eig(A))
   1.2234
  45. 2658
 -34. 4893
» [ R,D] = eig(A)
R =
   0.9979 -0.0798 -0.0590
   0.0492 -0.3915 -0.3530
   0.0416 -0.9167 0.9338
D =
   1. 2234        0           0
      0     45. 2658        0
      0            0       -34. 4893
```

Сингулярне розкладання матриці робить процедура ***svd***(A). Спрощене звернення до неї дозволяє одержати сингулярні числа матриці A. Більш складне звернення виду:

[U, S, V] = ***svd***(A)

дозволяє одержати три матриці – U, яка сформована з ортонормованих власних векторів, які відповідають найбільшим власним значенням матриці $A*A^T$; V – з ортонормованих власних векторів матриці A^T*A і S – діагональну матрицю, яка містить *невід'ємні значення квадратних коренів із власних значень матриці* A^T*A (їх називають *сингулярними числами*). Ці матриці задовольняють співвідношення:

$$A = U * S * VT.$$

Розглянемо приклад:

```
» disp(svd(A))
  100.5617
   15. 9665
    1. 1896
» [U,S,V] = svd(A)
U =
  -0.0207 0.1806 -0.9833
  -0.0869 0.9795 0.1817
  -0.9960 -0.0892 0.0045
S =
  100.5617 0 0
        0 15.9665 0
        0 0 1.1896
V =
   0. 0502   -0. 0221   -0. 9985
  -0. 9978   -0. 0453   -0. 0491
  -0. 0442    0. 9987   -0. 0243
```

Приведення матриці до форми Гессенберга здійснюється процедурою ***hess***(A). Наприклад:

```
A =
    1    2    3
   -1    8   16
   -5  100    3
» disp(hess(A))
   1. 0000   -3. 3340   -1. 3728
   5. 0990   25. 5000   96. 5000
        0    12. 5000  -14. 5000
```

Більш розгорнуте звернення [P,H] = ***hess***(A) дає можливість одержати, окрім матриці H в верхній формі Гессенберга, також унітарну матрицю перетворень P, яка задовольняє умови:

A = P * H * P; P' * P = eye(size(A)).

Приклад:

```
» [P,H] = hess(A)
P =
   1.0000        0         0
        0   -0.1961   -0.9806
        0   -0.9806    0.1961
H =
   1.0000    -3.3340    -1.3728
   5. 0990   25. 5000   96. 5000
        0    12. 5000  -14. 5000
```

Процедура ***schur*** (A) призначена для ***приведення матриці до форми Шура.*** Спрощене звернення до неї призводить до одержання матриці у формі Шура.

Комплексна форма Шура *– це верхня трикутна матриця із власними значеннями на діагоналі*. ***Дійсна форма Шура*** *зберігає на діагоналі тільки дійсні власні значення, а комплексні зображуються у виді блоків (2*2)*, частково займаючи нижню піддіагональ.

Звернення [U,T] = ***schur***(A) дозволяє, крім матриці T Шура, одержати також унітарну матрицю U, що задовольняє умовам:

A = U * H * U'; U' * U = eye(size(A)).

Якщо початкова матриця A є дійсною, то результатом буде *дійсна форма Шура*, якщо ж комплексною, то результат видається у виді *комплексної форми Шура.*

Наведемо приклад:

```
» disp(schur(A))
   1.2234 -6.0905    -4.4758
        0   45. 2658   84. 0944
        0    0. 0000  -34. 4893
» [U,T] = hess(A)
U =
    1.0000      0        0
        0    -0.1961  -0.9806
        0    -0.9806   0.1961
T =
   1. 0000   -3. 3340   -1. 3728
   5. 0990   25. 5000   96. 5000
        0    12. 5000  -14. 5000
```

Функція [U,T] = ***rsf2csf***(U,T) перетворює дійсну квазитрикутну форму Шура в комплексну трикутну:

```
» [U,T] = rsf2csf(U,T)
U =
  -0.9934 -0.1147    0
  -0.0449  0.3892   -0.9201
  -0.1055  0.9140    0.3917
T =
   1. 4091   -8. 6427    10. 2938
        0    45. 1689   -83. 3695
        0         0     -34. 5780
```

Процедура [AA, BB, Q, Z, V] = ***qz***(A,B) приводить ***пару матриць*** A і B до ***узагальненої форми Шура***. При цьому AA й BB є комплексними верхніми трикутними матрицями, Q, Z – матрицями приведення, а V – вектором узагальнених власних векторів такими, що

Q*A*Z = AA; Q*B*Z = BB.

Узагальнені власні значення можуть бути знайдені, виходячи з такої умови:

A*V*diag(BB) = B*V*diag(AA).

Необхідність в одночасному приведенні пари матриць до форми Шура виникає в багатьох задачах лінійної алгебри – розв’язуванні матричних рівнянь Сильвестра і Ріккаті, змішаних систем диференціальних і лінійних алгебричних рівнянь.

Приклад.

Нехай задана система звичайних диференціальних рівнянь у неявній формі Коші з одним входом u і одним виходом y такого вигляду:

$$\boldsymbol{Q}\cdot\dot{\boldsymbol{x}}+\boldsymbol{R}\cdot\boldsymbol{x}=\boldsymbol{b}\cdot u;$$

$$y=\boldsymbol{c}\cdot x+d\cdot u$$

причому матриці $\boldsymbol{Q}$, $\boldsymbol{R}$ і вектори $\boldsymbol{b}$, $\boldsymbol{c}$ і $\boldsymbol{d}$ дорівнюють відповідно

```
Q =
   1. 0000        0
     0.1920     1. 0000
R =
   1. 1190   -1. 0000
     36.4800   1. 5380
b =
  31. 0960
   0. 1284
c =
         0.6299       0
d =   -0. 0723
```

Необхідно обчислити значення полюсів і нулів відповідної передатної функції.

Ця задача зводиться до відшукування власних значень λ, що задовольняють матричні рівняння:

$$R\cdot r=-\lambda\cdot Q\cdot r;$$

$$\begin{bmatrix}-R & b\\ c & d\end{bmatrix}\cdot r=\lambda\cdot\begin{bmatrix}Q & 0\\ 0 & 0\end{bmatrix}\cdot r.$$

Розв'язання першого рівняння дозволяє ***обчислити полюси передатної функції***, а другого – ***нулі.***

Нижче наведено сукупність операторів, яка призводить до *розрахунку полюсів:*

```
» [AA, BB] = qz(R,-Q)   % Приведення матриць до форми Шура
AA =
   5.5039 + 2.7975i      24.8121 -25.3646i
   0.0000 – 0.0000i        5.5158 – 2.8036i
BB =
  -0. 6457 + 0. 7622i  -0. 1337 + 0. 1378i
            0                -0. 6471 – 0. 7638i
» diag(AA) . /diag(BB)  % Розрахунок полюсів
ans =
  -1. 4245 – 6. 0143i
  -1. 4245 + 6. 0143i
```

Розрахунок нулів здійснюється в такий спосіб:

```
» A = [-R                  b                        % Формування
        c        d]                          % першої матриці
A =
                   -1.1190      1.0000      0.1284
                  -36. 4800   -1. 5380   31. 0960
       0.6299          0       -0. 0723
» B = [ -Q                 zeros(size(b))          % Формування
        zeros(size®)       0                ]      % другої матриці
B =
   -1.0000         0            0
   -0. 1920   -1. 0000         0
        0            0          0

» [AA,BB] = qz(A,B)                    % Приведення матриць   до форми Шура
AA =
```

```
   31.0963 -0.7169 -36.5109
    0.0000   1.0647   0.9229
         0   0.0000   0.5119
BB =
         0   0.9860   -0.2574
         0   0. 0657   0. 9964
         0         0   -0. 0354
» diag(AA) . /diag(BB)                  % Обчислення нулів
ans =
       Inf
    16. 2009
   -14. 4706
```

Обчислення ***власних значень матричного полінома*** здійснює процедура ***polyeig.*** Звернення

[R, d] = ***polyeig***(A0, A1,... , Ap)

дозволяє розв'язати повну проблему власних значень для матричного полінома ступеня p виду

(A0 + λ*A1 + ... + λ^p*Ap)*r = 0.

Вхідними змінними цієї процедури є p+1 квадратні матриці A0, A1, ... Ap порядку n. Вихідними змінними – матриця власних векторів R розміром (n*(n*p)) і вектор d власних значень розміром (n*p).

Функція ***polyvalm*** призначена для ***обчислення матричного полінома*** виду

$$Y(X) = p_n \cdot X^n + p_{n-1} \cdot X^{n-1} + \ldots + p_2 \cdot X^2 + p_1 \cdot X + p_0$$

за заданим значенням матриці X и вектора p = [pn, pn-1, ... , p0] коефіцієнтів полінома. Для цього достатньо звернутися до цієї процедури за схемою:

Y = ***polyvalm***(p, X).

Приклад:

```
p =    1    8   31   80   94   20
» X
X =
    1    2    3
    0   -1    3
    2    2   -1
» disp(polyvalm(p,X))
      2196       2214       2880
       882        864       1116
      1332       1332       1746
```

Примітка. *Слід розрізнювати процедури* ***polyval*** *і* ***polyvalm****. Перша обчислює значення поліному для кожного з елементів матриці аргументу, а друга при обчисленні полінома підносить до відповідного степеня всю матрицю аргументу.*

Процедура ***subspace***(A,У) обчислює кут між двома підпросторами, які "натягнуті на стовпчики" матриць A і B. Якщо аргументами є не матриці, а вектори A і B, обчислюється кут між цими векторами.

5. ПАКЕТ ПРОГРАМ ВІЗУАЛЬНОГО ПРОГРАМУВАННЯ SIMULINK

Пакет SimuLink дозволяє здійснювати дослідження (моделювання у часі) поводження динамічних нелінійних систем. Утворення чисельної моделі досліджуваної системи здійснюється *шляхом графічного складання* у спеціальному вікні *схеми з'єднань елементарних* візуальних блоків, що містяться в бібліотеках SimuLink. Кожний блок фактично являє собою математичну програму. Лінії з'єднання блоків перетворюються на зв'язки між цими програмами, які дозволяють визначити послідовність виклику програм і пересилання інформації. У результаті такого складання утворюється програмна модель, яку надалі називатимемо S-моделлю і яка зберігається у файлі з розширенням ***.mdl***. Такий процес утворення обчислювальних програм прийнято називати *візуальним програмуванням*.

Створення моделей у пакеті SimuLink ґрунтується на використанні технології Drag-and-Drop (*Перетягни й Залиш)*. Як "цеглинки" при побудові S-моделі використовуються модулі (блоки), що зберігаються в бібліотеці SimuLink. S-модель може мати ієрархічну структуру, тобто складатися з моделей більш низького рівня, причому кількість рівнів ієрархії є практично необмеженою. Протягом моделювання є можливість спостерігати за процесами, що відбуваються в системі. Для цього використовуються спеціальні блоки "оглядові вікна", що входять до складу бібліотеки SimuLink. Склад бібліотеки SimuLink може бути поповнений користувачем за рахунок розробки власних блоків.

Використання SimuLink є особливо зручним при моделюванні систем, які складаються із з'єднаних певним чином окремих функціональних пристроїв, поведінка яких описується відомими залежностями. Тоді схема з'єднань візуальних блоків у вікні блок-схеми S-моделі збігається з реальними зв'язками між цими пристроями. Ця обставина суттєво спрощує програмний аналіз і синтез систем автоматичного керування.

Однією з найпривабливіших особливостей системи MatLAB є наявність в її складі найбільш наочного і ефективного засобу складання програмних моделей – пакета візуального програмування ***Simulink***.

Пакет ***Simulink*** дозволяє здійснювати досліджування (моделювання у часі) поводження динамічних лінійних і нелінійних систем, причому складання «програми» і введення характеристик досліджування систем здійснювати в діалоговому режимі, *шляхом графічного складання на екрані схеми з'єднань елементарних* (стандартних або користувацьких) *графічних блоков*. В результаті такого складання виходить модель досліджуваної системи, яку надалі називатимемо S-моделлю і яка зберігається у файлі з розширенням ***.mdl***. Такий процес утворення обчислювальних програм прийнято називати *візуальним програмуванням*.

Утворення моделей у пакеті ***Simulink*** грунтується на використанні технології Drag-and-Drop (*Перетягни і залиш)*. У якості «кирпичиків» при побудові S-моделі використовуються візуальні блоки (модулі), які зберігаються у бібліотеках ***Simulink***. S-модель може мати ієрархичну структуру, тобто складатися з моделей більш низького рівня, причому кількість рівнів ієрархії практично не обмежена. У процесі моделювання є можливість сростерігати за процесами, що відбуваються у системі. Для цього використовуються спеціальні блоки («оглядові вікна»), що входять до складу бібліотеки ***Simulink***. Біблиотека ***Simulink*** може бути поповнена користувачем за рахунок розроблення власних блоків.

До переваг користування ***Simulink***-моделями відносяться:

- вельми зручний, наочний і ефективний спосіб утворення програм моделювания навіть доволі складних динамічних систем – *візуальне* програмування, – шляхом складання на екрані блок-схеми системи зі стандартних готових блоків;
- доволі зручні і наочні засоби втручання в готову блок-схему системи з метою її перетворення або отримання додаткової інформації про змінювання проміжних процесів;
- широкий набір ефективних програм розв'язувачів (Solvers, методів чисельного інтегрування) диференціальних рівнянь (з фіксованим кроком інтегрування, зі змінним кроком, а також розв'язувачів так званих «жорстких» систем диференціальних рівнянь);
- відсутність необхідності в спеціальній організації процесу чисельного інтегрування;
- унікальні можливості інтегрування нелінійних систем з «суттєвими» нелінійностями (коли нелінійна залежність має стрибкоподібний характер);
- вельми швидке і зручне отримування графічної інформації про змінювання модельованих величин з часом.

Більш докладно можливості пакету ***Simulink***, зміст бібліотеки ***Simulink*** викладені у главі 3 цього навчального посібника. Там же наведені приклади утворення S-моделей і роботи з ними.

5.1. Запуск SimuLink

Запуск SimuLink складається з двох процедур:

1) виклику створеної раніше S-моделі шляхом введення у командному рядку командного вікна MatLAB ймення відповідного MDL-файла або обравши команду **New > Model** у меню **File**, якщо S-модель ще тільки потрібно утворити; у першому випадку при цьому виникне нове вікно з блок-схемою, а у другому – порожнє вікно **untitled** (вікно, де має бути утворена блок-схема нової S-моделі, MDL-файлу, рис. 5.1);
2) виклику браузера бібліотеки SimuLink шляхом натиснення відповідної піктограми у лінійці інструментів командного вікна;

виникне вікно **Simulink Library Browser** (рис. 5.2), яке містить перелік основних поділів бібліотеки SimuLink; більш зручним є користування вікном **Library: simulink** (рис. 5.3), яке викликається за допомогою контекстного меню. В ньому представлені графічні позначення поділів бібліотеки

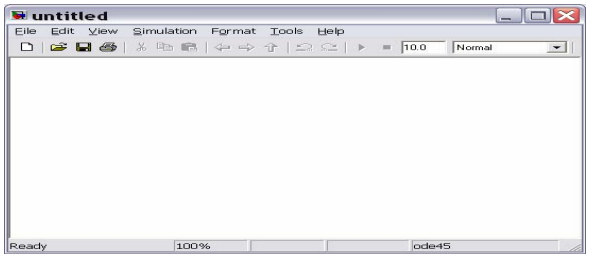


Рис. 5.1. Вікно блок-схеми S-моделі

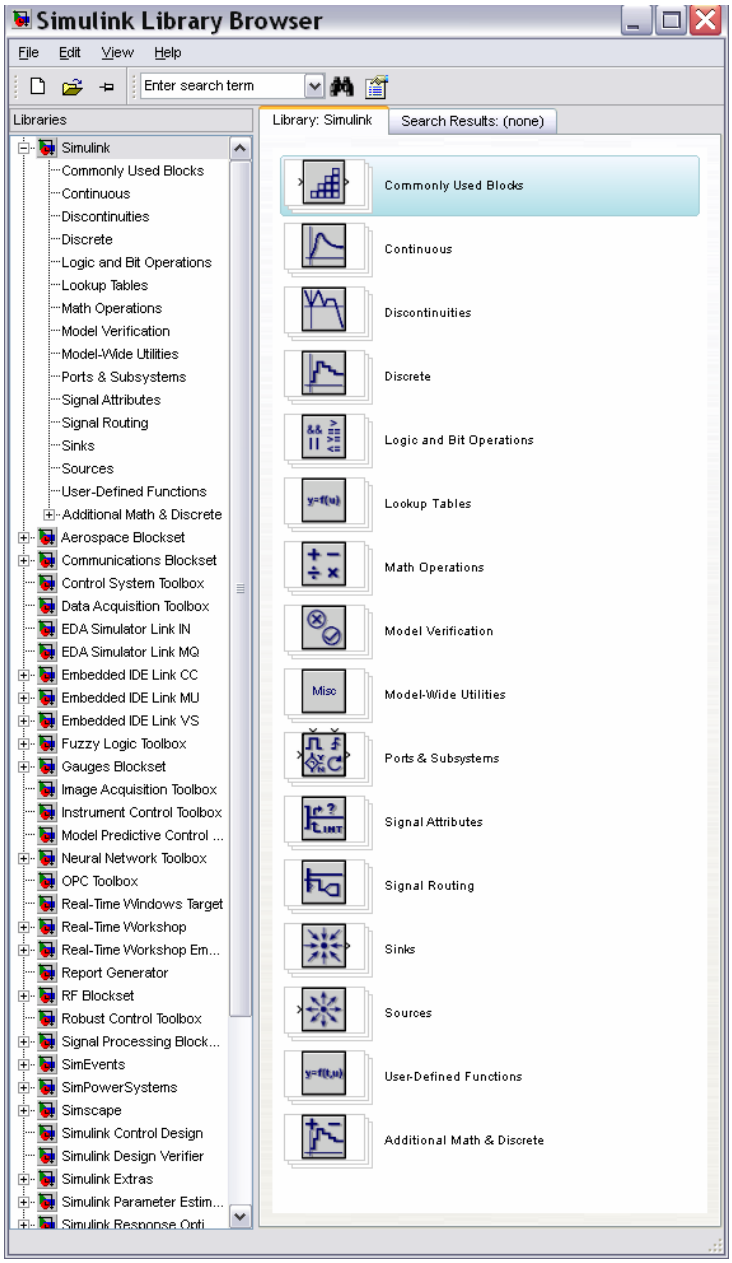


Рис. 5.2. Вікно браузера бібліотеки Simulink

Усі вікна мають подібну структуру і містять рядок меню і робоче поле.

Меню **File** (Файл) містить команди роботи з MDL-файлами, меню **Edit** (редагування) – команди редагування блок-схеми й роботи з бібліотекою, а меню **View** (Подання) – команди змінювання зовнішнього вигляду вікна.

У меню **Simulation** (Моделювання) містяться команди керування моделюванням, а в меню **Format** (Формат) – команди редагування формату (тобто зовнішнього зображення) блоків схеми й блок-схеми в цілому.

Якщо до складу робочої конфігурації MatLAB включений додаток *Real-Time-Workshop,* то меню доповнюється поділом **Tools** (Інструменти), що містить засоби роботи з ним.

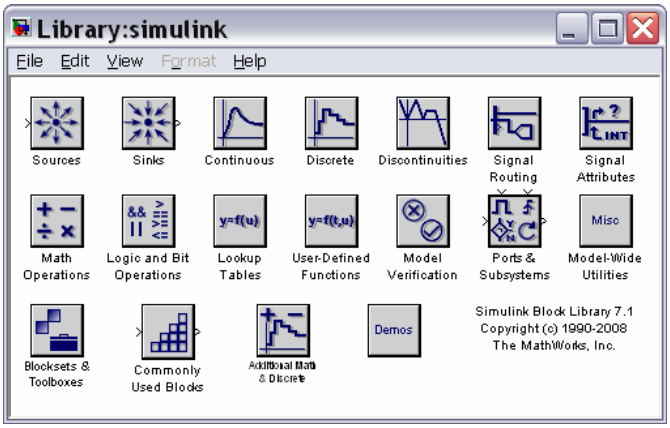


Рис. 5.3. Вікно Library: simulink

5.2. Бібліотека SimuLink

В основі блок-схем S-моделей лежать елементарні блоки, які дозволяють зв’язати блок схему з середовищем Matlab і забезпечити функціонування в ньому S-моделі як програми. Ці блоки містяться у головній бібліотеці пакету ***Simulink***, яка має ту саму назву.Бібліотека блоків SimuLink є набором візуальних об'єктів, використовуючи які, можна, з’єднуючи окремі модулі між собою лініями функціонального зв'язку, складати функціональну блок-схему будь-якого устрою.

Бібліотека блоків складається з 17 поділів. Десять з них є головними і не можуть змінюватися користувачем:

- ***Sources*** (Джерела)*;*
- ***Sinks*** (Приймачі)**;**
- ***Continuous*** (Лінійні неперервні елементи)**;**
- ***Discrete*** (Дискретні елементи)*;*
- ***Discontinuities*** (Розривні елементи)
- ***Signal Routing*** (Пересилання сигналів)**;**
- ***Math Operations*** (Математичні операції)**;**
- ***Logic & Bit Operations*** (Логічні та бітові операції);

■ ***User-defined Functions*** (Функції, що визначаються користувачем);

■ ***Ports & Subsystems*** (Порти та підсистеми).

Блоки, що входять у поділ ***Sources*** (Джерела), призначені для *формування сигналів*, які забезпечують керування роботою S-моделі в цілому або окремих її частин. Усі *блоки-джерела мають по одному виходу і не мають входів.*

Блоки, зібрані в поділі ***Sinks*** (Приймачі)*, мають тільки входи і не мають виходів*. Умовно їх можна поділити на 3 види:

■ блоки, які використовуються як оглядові вікна при моделюванні;

■ блоки, що забезпечують зберігання проміжних і вихідних результатів моделювання;

■ блок керування моделюванням, який дозволяє переривати моделювання при виконанні тих або інших умов.

Поділ ***Continuous*** (Лінійні неперервні елементи) містить блоки, які можна умовно поділити на дві групи:

■ блоки лінійних ланок неперервного часу;

■ блоки лінійних стаціонарних ланок із затримкою.

У поділ ***Discrete*** (Дискретні елементи) входять блоки, за допомогою яких у моделі може бути описане поводження дискретних систем. Розрізняють два основних типи таких систем: *системи з дискретним часом* і *системи з дискретними станами.* Блоки, що входять у поділ ***Discrete*** забезпечують моделювання як тих, так і інших.

Поділ ***Discontinuities*** (Розривні елементи) містить 12 блоків, які релізують кусково-лінійні залежності.

У поділі ***Signal Routing*** (Пересилання сигналів) розташовані блоки, що здійснюють об'єднання, роз'єднання сигналів, їх переключення, пересилання тощо.

Блоки поділу ***Math Operations*** (Математичні операції) реалізують перетворення сигналів, що подаються на входи, за різними типовими математичними залежностями, векторно-матричні операції і перетворення комплексних векторів.

У поділі ***Logic & Bit Operations*** (Логічні та бітові операції) зосереджені блоки, що здійснюють логічні і бітові операції з сигналами, які поступають ло їхнього входу.

Блоки, що входять у склад поділу ***User-defined Functions*** (Функції, що визначаються користувачем) призначені для утворення користувацьких блоків.

Нерешті, у поділ ***Ports & Subsystems*** (Порти та підсистеми) входять блоки, яки здійснюють зв'язок між моделями різних рівнів ієрархії, а також дозволяють утворити підсистеми, тобто моделі більш низького рівня ієрархії.

Щоб перейти у вікно відповідного поділу бібліотеки, у якому розташовані графічні зображення блоків, достатньо подвійно клацнути мишкою на піктограмі цього поділу

Складання схеми S-моделі полягає в тому, що графічні зображення обраних блоків за допомогою миші перетягуються з вікна поділу бібліотеки у вікно

складання схеми, а потім виходи одних блоків у вікні складання з'єднуються із входами інших блоків також за допомогою мишки.

Технологія перетягування зображення блока така: курсор мишки потрібно встановити на зображенні обраного блока у вікні поділу бібліотеки, потім натиснути ліву клавішу мишки і, не відпускаючи її, пересунути курсор на поле складання схеми, після чого відпустити клавішу. Аналогічно здійснюються з'єднання у схемі лініями виходів одних блоків із входами інших блоків: курсор мишки підводять до потрібного виходу певного блока (при цьому курсор має набути форму хрестика), натискують ліву клавішу і, не відпускаючи її, курсор переміщують до потрібного входу іншого блока, а потім відпускають клавішу. Якщо з'єднання зроблено вірно, на вході останнього блоку виникне зображення чорної затушованої стрілки.

5.2.1. Поділ Sinks (Приймачі)

Після переходу до поділу ***Sinks*** на екрані з'являється вікно цього поділу, зображене на рис. 5.4. З його розгляду випливає, що в цьому поділі розміщено три групи блоків, які не мають виходів, а мають тільки входи:

1) блоки, які при моделюванні відіграють роль оглядових вікон; до них відносяться:

- блок ***Scope*** з одним входом, який виводить графік залежності величини, яка подається до його входу, від модельного часу;
- блок ***XYGraph*** із двома входами, який забезпечує побудову графіка залежності однієї модельованої величини (другий зверху вхід) від іншої (перший вхід);
- блок ***Display*** з одним входом, призначений для відображення чисельних значень вхідної величини;

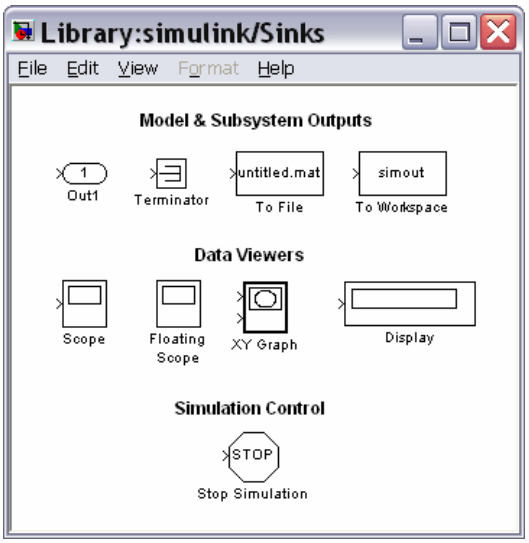


Рис. 5.4. Вміст поділу Sinks

2) блоки для пересилання результатів:

- блок ***To File***, який забезпечує зберігання результатів моделювання на диску в MAT-файлі (із розширенням .mat);
- блок ***To Workspace***, який зберігає результати в робочому просторі;
- блок- заглушка ***Terminator***;
- блок порт виходу ***Out***;

3) блок керування моделюванням – ***Stop Simulation***, який дозволяє переривати моделювання при виконанні тих або інших умов; блок спрацьовує в тому випадку, коли на його вхід надходить ненульовий сигнал.

Блок ***Scope***

Цей блок дозволяє в процесі моделювання спостерігати на графіку процеси, що цікавлять дослідника. Він має один вход, на який подається сигнал, графік залежності якого від часу потрібно вивести у вікні.

Для настроювання параметрів цього блока потрібно *після встановлення зображення блока у вікно блок-схеми двічі клацнути мишкою на його зображенні*. У результаті на екрані з'явиться вікно **Scope** (рис. 5.5). Розмір і пропорції вікна можна змінювати довільно, користуючись мишкою. По горизонтальній осі відкладаються значення модельного часу, а по вертикальній – значення вхідної величини, які відповідають цим моментам часу. Якщо вхідна величина блока ***Scope*** є вектором, у вікні будуються графіки змінювання всіх елементів цього вектора, тобто стільки кривих, скільки елементів у вхідному векторі, причому кожна – свого кольору. Одночасно у вікні може відображатися до 30 кривих.

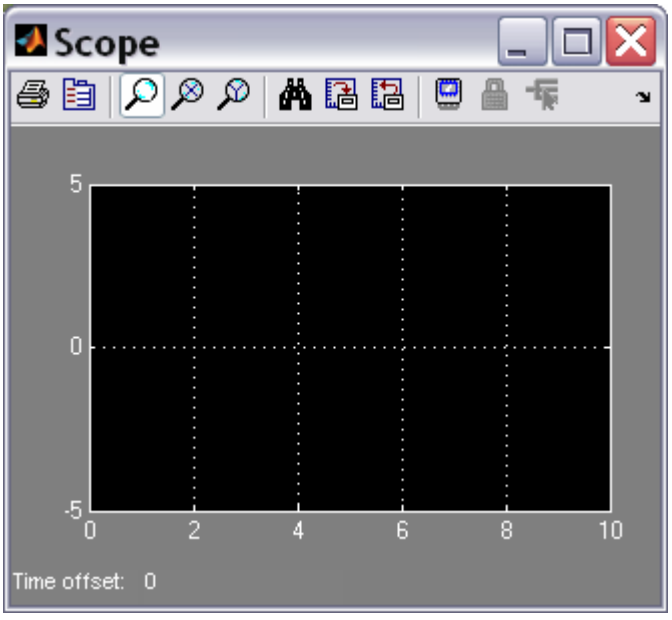


Рис. 5.5. Оглядове вікно Scope

Для керування параметрами вікна в ньому є панель інструментів, що містить 11 піктограм із таким призначенням (зліва праворуч):

- виведення вмісту вікна на прінтер
- виклик вікна настроювання параметрів блоку

- змінювання масштабу одночасно по обох осях графіка;
- змінювання масштабу по горизонтальній осі;
- змінювання масштабу по вертикальній осі;
- автоматичне встановлення оптимального масштабу осей (повний огляд, автошкалювання);
- зберігання встановленого масштабу осей;
- відновлення установок параметрів осей;
- включення холостого під'єднання блоку;
- шлюз селектору сигналів
- селектор сигналів.

Піктограми змінювання масштабу є альтернативними, тобто в кожний момент часу може бути натиснута лише одна з них. Піктограми не активні доти, поки немає графіка у вікні ***Scope.*** Активними із самого початку є лише перші дві, шоста, сьома і девята піктограми. Щиглик на другій піктограмі призводить до появи діалогового вікна настроювання параметрів *'Scope' parameters* (рис. 5.6).

Це вікно має дві вкладинки:

- *General* (Загальні), яка дозволяє встановити параметри осей;
- *Data history* (Установлювання даних), яку призначено для визначення параметрів подання даних блока ***Scope.***

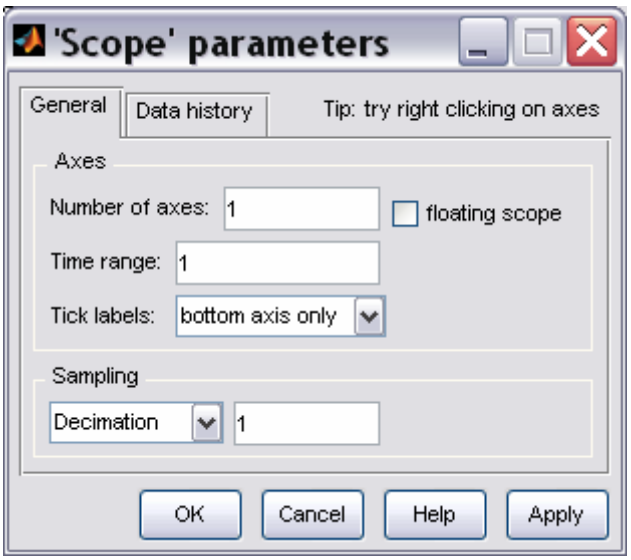


Рис. 5.6. Вікно настроювання блоку Scope

У нижній частині вікна розташовані кнопки: *Apply (Застосувати), Help (Виклик довідки), OK (Підтвердити установку), Cancel (Повернутися назад).*

В області *Axes* (Осі) вкладинки General містять поля введення *Number of axes (Кількість осей)* і *Time range (Інтервал часу)*, а також список *Tick labels (Мітки осей)*.

У першому полі задається кількість графічних полів у вікні *Scope* (одночасно змінюється кількість входів блоку ***Scope***).

У другому полі встановлюється верхня межа модельного часу, що відкладається по осі абсцис. При цьому слід мати на увазі таке. *Якщо розмір заданого інтервалу моделювання ($T_м$) не перевищує встановленого у цьому полі значення (тобто весь процес уміщується у вікні Scope), під графіком у рядку Time offset (Зсув за часом) виводиться значення 0. У випадку ж, коли інтервал моделювання перевищує встановлене значення, у вікні Scope відображається тільки графік, що відповідає останньому відрізку часу, меншому за розміром, ніж Time range і рівному $T_м$-n*Time range, де n – ціле число. При цьому в рядку Time offset виводиться розмір "схованого" інтервалу часу – n*Time range.* Наприклад, якщо значення *Time range* дорівнює 3, а тривалість інтервалу моделювання встановлена 17, то у вікні ***Scope*** буде виведений графік модельованого процесу за останні 2 одиниці часу, а рядок під графіком буде мати вид: *Time offset: 15*.

За допомогою списку *Tick labels (Мітки осей)* можна задати вид оформлення осей координат в графіках вікна ***Scope***. Зазначений список містить три пункти: *all* (все), *bottom axis only* (лише нижньої осі), *none* (нет). В результаті обрання першого з них поділки по осях будуть нанесені вздовж кожної з осей усіх графіків. Обрання другого означає, що поділки по горизонтальних осях графічних полів (якщо їх декілька) за винятком нижньої будуть відсутні. нарешті, якщо обрати третій пункт, то зникнуть поділки по осях графіків і написи на них, графік займе усе поле вікна і останнє прийме вид, показаний на рис. 5.7.

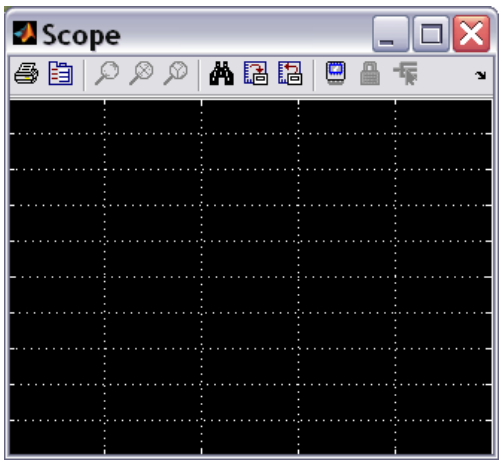


Рис.5.7. Вікно Scope при встановленні NONE

Якщо в області *Axes* (Осі) встановити прапорець *Floating scope,* то входи у блок ***Scope*** будуть відімкнені. У цьому випадку блок відображується як такий, що не має входу, і якщо від був зв'язаний з іншими блоками, ці зв'язки обриваються. Той самий ефект справляє клацання на кнопці з тою самою назвою, що міститься на панелі інструментів блоку.

В області *Sampling* (Дискретизація) міститься список, в якому обраний елемент *Decimation* (Проріджування), і поле, де можна ввести ціле додатне число, яке визначає, через яку кількість проміжків часу (дискретів часу) одержані дані використовуватимуться для побудови графіків в окні *Scope.*

Вкладинка *Data History* (Історія даних) вікна *'Scope' parameters* (рис. 5.8) дозволяє задати максимальну кількість (починаючи з кінця) елементів масивів даних, що використовуються для побудови графіків у вікні *Scope* (поле поряд з прапорцем *Limit data points to last* (Максимальна кількість точок даних).

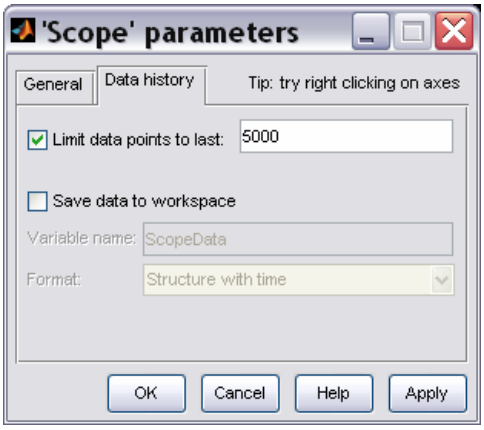


Рис. 5.8. Вкладинка Data History

Якщо встановити прапорець *Save data to workspace* (Записати дані у робочий простір), виникне можливість записати у робочий простір дані, які виводяться на графікі вікна *Scope*. При цьому становляться досяжними поле *Variable name* (Ім'я змінної) і список *Format* (Формат). В поле можна ввести ймення змінної, під яким зберігатимуться дані у робочому просторі (за замовчуванням ці дані будуть записані під ім'ям *ScopeData*), а у спискуможна обрати один з трьох форматов запису даних: *Array* (Масив, матриця), *Structure* (Структура) або *Structure with time* (Структура з часом).

Продемонструємо роботу блока на прикладі. Перетягнемо у вікно блок-схеми з вікна поділу *Sources* блок ***Sine Wave***, а з вікна поділу *Sinks* – блок ***Scope*** і з'єднаємо вихід першого блоку зі входом другого. Одержимо схему, показану на рис. 5.9

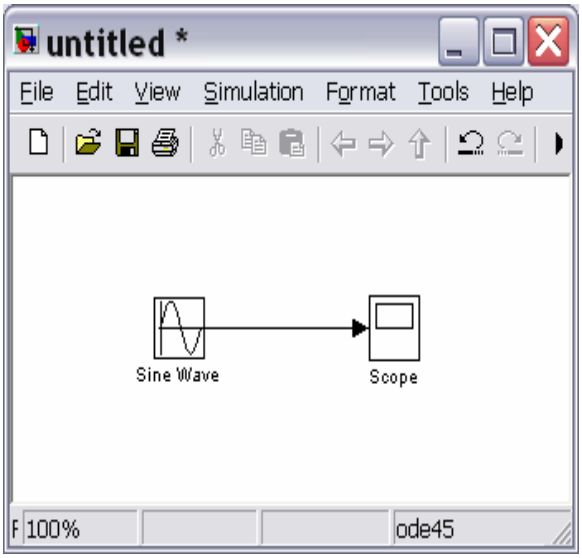


Рис. 5.9. Простіша блок-схема з блоком Scope

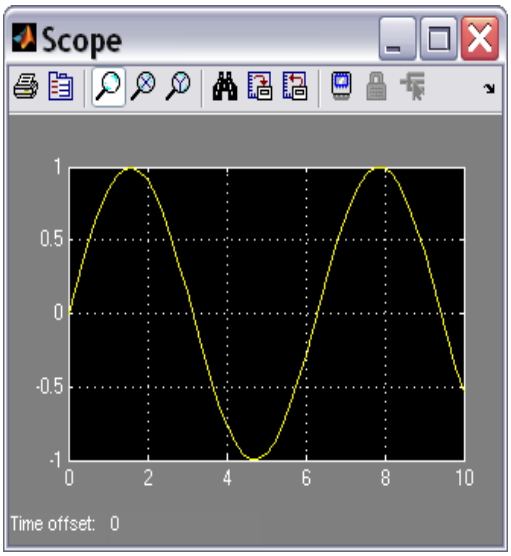


Рис. 5.10. Вікно Scope

Викличемо у вікні цієї блок-схеми команду *Simulation* > *Start* (Моделювання > Почати), потім подвійно клацнемо на зображенні блоку ***Scope***. На екрані виникне вікно *Scope* цього блоку з зображенням графіка змінювання у часі гармонічного сигналу (рис. 5.10).

*Блок **XYGraph***

Цей блок також є оглядовим вікном. На відміну від ***Scope***, він має два входи: на перший (верхній) подається сигнал, значення якого відкладаються по горизонтальній осі графіка, а на другий (нижній) – по вертикальній осі.

Якщо перетягнути цей блок на поле блок-схеми, а потім на зображенні його подвійно клацнути мишкою, то на екрані виникне вікно настроювання блока (рис. 5.11), яке дозволяє встановити межі змінювання обох вхідних величин, в яких буде побудований графік залежності другої величини від першої, а також задати дискрет за часом.

Наведемо приклад використання блока ***XYGraph***. Для цього перетягнемо у вікно блок-схеми зображення цього блока з вікна *Library simulink/Sinks*, а з вікна *Library simulink/Sources* – два блоки-джерела ***Clock*** і ***Sine Wave***. З'єднаємо виходи блоків-джерел із входами блока ***XYGraph.*** Одержимо блок-схему, наведену на рис. 5.12.

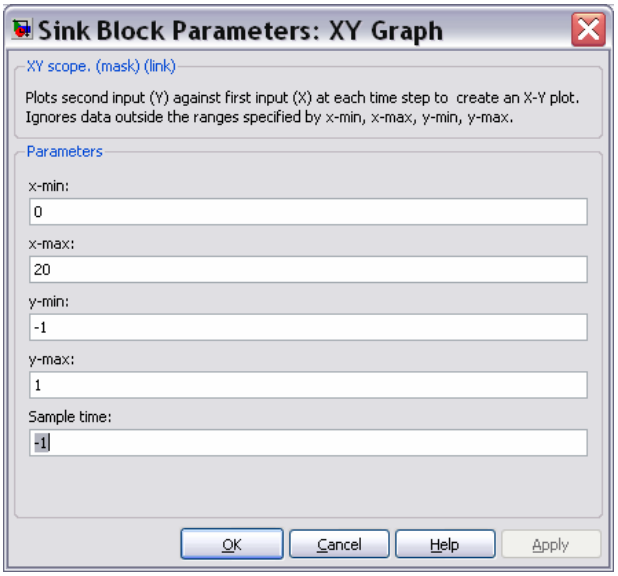


Рис. 5.11. Вікно настроювання блоку XYGraph

Перш ніж запустити процес моделювання цієї схеми, необхідно настроїти блок ***XYGraph***, вводячи у діалоговому вікні *Sink Block Parameters: XY Graph* очікувані діапазони змінювання величин *x* (у розглядуваному випадку – часу) і *y* (синусоїдального сигналу). Вкажено у полях введення x-min, x-max, y-min, y-max відповідно значення 0, 20, -1 і 1, як це відображено на рис. 5.11. Зазначимо, що у випадку використання блоку *Scope* вводити діапазони змінювання величин не потрібно, вони встановлюються автоматично.

Якщо тепер вибрати мишкою меню *Simulation* у рядку меню вікна блок-схеми, а в ньому – команду *Start*, то по закінченні розрахунків у вікні блока ***XYGraph*** з'явиться зображення, подане на рис. 5.13.

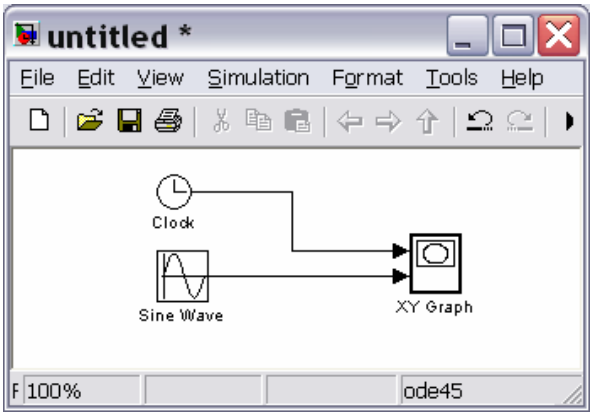


Рис. 5.12. Блок-схема перевірки роботи блоку XYGraph

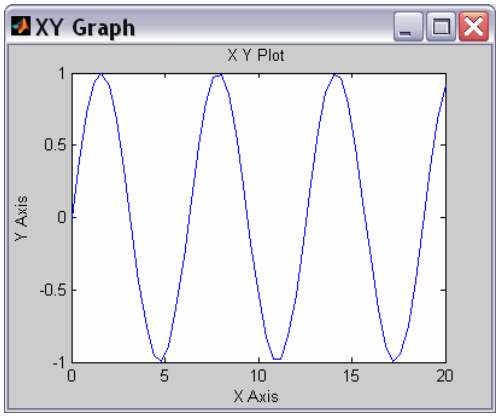


Рис.5.13. Вікно блоку XYGraph з графіком синусоїди

*Блок **Display***

Цей блок призначений для виведення на екран чисельних значень величин, що фігурують у блок-схемі.

Блок має 3 параметри настроювання (рис. 5.14). Список *Format* – задає формат виведення чисел; вид формату обирається за допомогою спадного меню, що містить 5 пунктів: *short, long, short_e, long_e, bank*. Поле введення *Decimation* дозволяє задати періодичність (у дискретах часу) виведення значень у вікні *Display*. Перемикач *Floating display* дозволяє визначати блок ***Display*** як блок без входу, обриваючи його зв'язки.

Блок ***Display*** може використовуватися для виведення як скалярних, так і векторних величин. Якщо відображувана величина є вектором, то вихідне подання блока змінюється автоматично, про що свідчить поява маленького чорного трикутника в правому нижньому рогові блока. Для кожного елемента вектора створюється своє міні-вікно, але щоб вони стали видимими, необхідно розтягнути зображення блока. Для цього слід виділити блок, підвести курсор мишки до одного з його рогів, натиснути ліву клавішу мишки і, не відпускаючи її, розтягнути зображення блока, поки не зникне чорний трикутник.

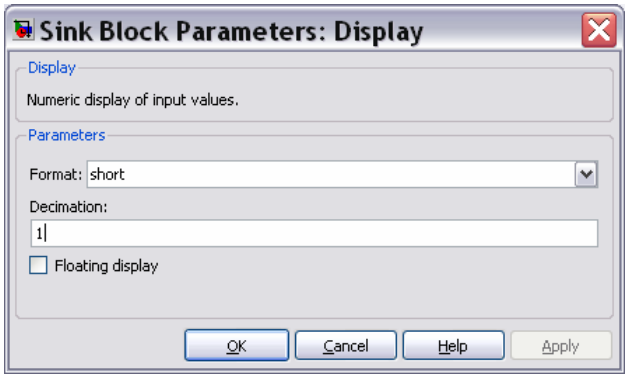


Рис.5.14. Вікно настроювання блоку Display

Для прикладу створимо блок-схему (рис. 5.15) із двох елементів – блока-джерела ***Constant*** і блока-приймача ***Display***.

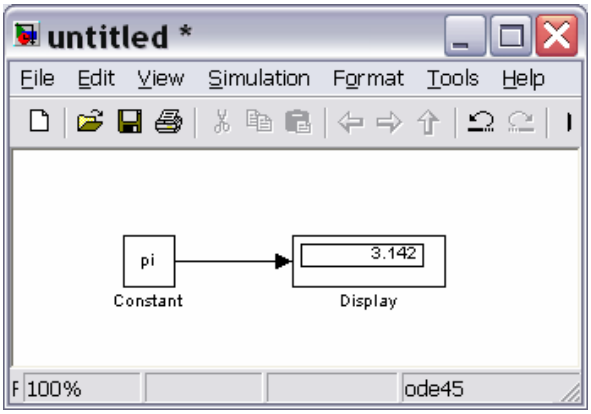


Рис. 5.15. Блок-схема перевірки блоку Display

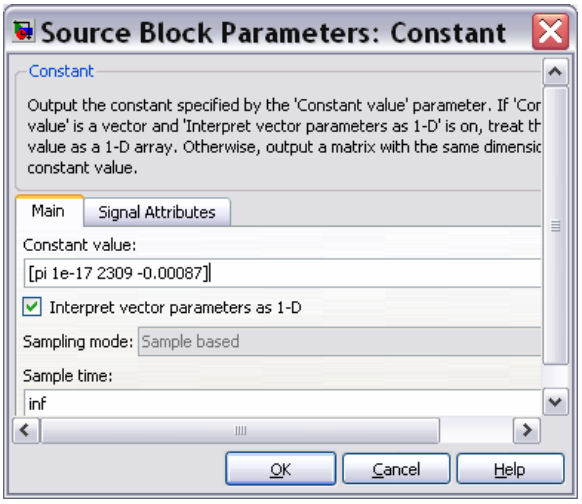


Рис. 5.16. Вікно настроювання блоку Constant

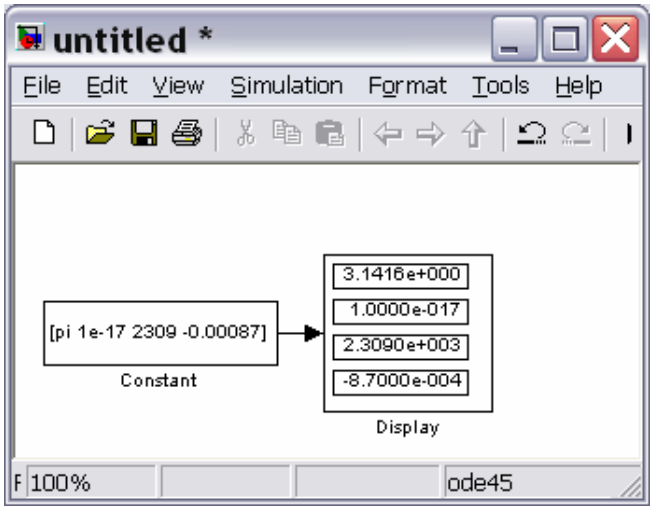


Рис. 5.17. Результат перевірки роботи блоку Display

Викликавши вікно настроювання блока ***Constant*** (рис. 5.16), встановимо в ньому значення константи-вектора, який складається із чотирьох елементів [pi

1e-17 2309 -0.00087]. Викликаючи вікно настроювання блока ***Display***, встановимо з його допомогою формат виведення чисел *short_e*. Після активізації команди *Start* із меню Simulation, розтягуючи зазначеним чином зображення блока ***Display*** на блок-схемі, одержимо картину, подану на рис. 5.17.

*Блок **To File***

Цей блок забезпечує запис значень величини, поданої на його вхід, у MAT-файл даних для використання їх у наступному в інших S-моделях.

Блок має такі параметри настроювання (див. рис. 5.18):

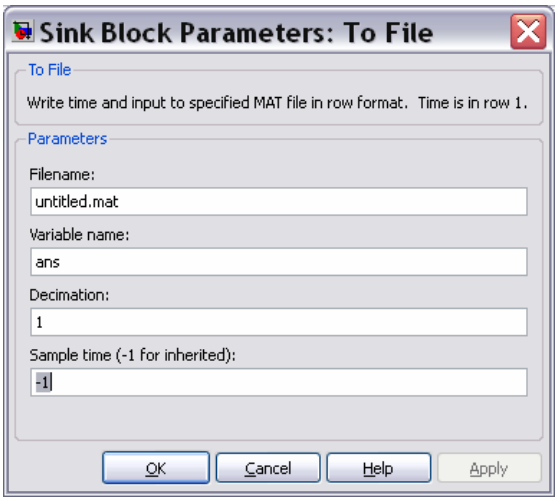


Рис. 5.18. Вікно настроювання блоку To File

Filename – ім'я MAT-файлу, в який записуватимуться значення вхідної величини; за замовчуванням – ***untitled. mat***; ймення файлу виводиться на зображенні блока в блок-схемі;

Variable name – ім'я змінної, за яким можна буде звертатися до даних, записаних у файлі (для того, щоб переглянути або змінити їх у командному вікні MatLAB); за замовчуванням використовується системне ім'я ***ans***;

Decimation – кількість дискретів часу, через яке провадитиметься запис даних у файл;

Sample Time – розмір дискрету часу для даного блока.

Слід зазначити, що значення даних, що подаються до входу блока записуються у вихідну змінну (наприклад, ***ans***) у такий спосіб: перший рядок матриці утворюють значення відповідних моментів часу; другий рядок містить відповідні значення першого елемента вхідного вектора, третій рядок – значення другого елемента і т. д. В результаті записується матриця розміром (k+1)*N, де k – кількість елементів вхідного вектора, а N – кількість точок вимірювання (або кількість моментів часу, у які здійснено вимірювання).

*Блок **To Workspace***

Цей блок призначається для зберігання даних у робочому просторі системи MatLAB. Дані зберігаються у виді матриці розміром (N*k), структура якої відрізняється від структури даних у MAT-файлі тим, що:

- значення величин, що зберігаються, розташовані по стовпцях, а не по рядках;
- не записуються значення модельного часу.

Блок має 4 параметри настроювання (див. рис. 5.19):

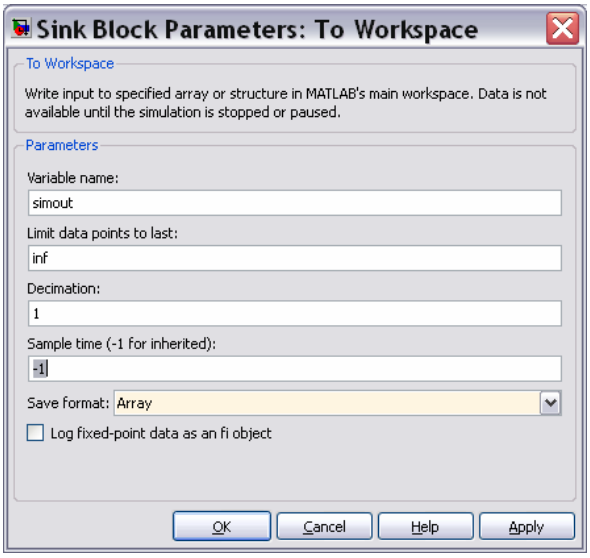


Рис. 5.19. Вікно настроювання блоку To Workspace

Variable name – ім'я, із яким дані зберігаються в робочому просторі (за замовчуванням – *simout*);

Limit data points to last – максимально припустима кількість точок, тобто значень даних, що записуються; за замовчуванням вона задається константою ***inf***, тобто дані реєструються на всьому інтервалі моделювання;

Decimation і *Sample Time* мають той самий зміст, що і раніше.

5.2.2. Поділ Sources (Джерела)

Блоки, що входять до поділу ***Sources*** (Джерела), призначені для формування сигналів (послідовності числових даних), які при моделюванні забезпечують роботу S-моделі у цілому або окремих її частин. Усі блоки мають по одному виходу і не мають входів.

Після вибору поділу ***Sources*** бібліотеки SimuLink на екрані з'явиться додаткове вікно, показане на рис. 5.20.

Як бачимо, поділ містить дві групи блоків: ***Model &Subsystem Inputs*** (Входи моделей і підсистем) та ***Signal Generators*** (Генератори сигналів).

Як джерела сигналів передбачені такі блоки:

- ***Constant*** – формує постійну величину (скаляр, вектор або матрицю);
- ***Signal Generator*** – створює (генерує) неперервний коливальний сигнал однієї із хвильових форм на вибір – синусоїдальний, прямокутний, трикутний чи випадковий;
- ***Pulse Generator*** - генератор неперервних прямокутних імпульсів;
- ***Signal Builder*** – побудувач сигналів,

- ***Ramp*** – створює лінійно висхідний (або спадний) сигнал (сходинку за швидкістю);

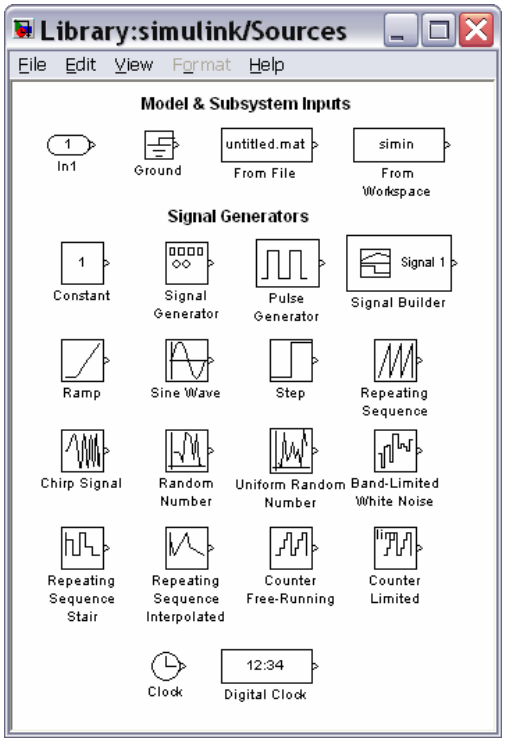


Рис. 5.20. Вміст поділу Sources

- ***Sine Wave*** – генерує гармонічний сигнал;
- ***Step*** – генерує сигнал у виді поодинокої сходинки (східчастий сигнал) із заданими параметрами (початку сходинки і її висоти);
- ***Repeating Sequence*** – генерує періодичну послідовність;
- ***Chirp Signal*** – генератор гармонічних коливань із частотою, яка лінійно змінюється з часом;
- ***Random Number –*** джерело дискретного сигналу, значення якого є випадковою величиною, розподіленою за нормальним (гауссовим) законом;
- ***Uniform Random Number*** – джерело дискретного сигналу, значення якого є випадковою рівномірно розподіленою величиною;
- ***Band-Limited White Noise*** – генератор білого шуму з обмеженою смугою частот;
- ***Clock*** (Годинник) – джерело неперервного сигналу, пропорційного до модельного часу;
- ***Digital clock*** (Цифровий годинник) – формує дискретний сигнал, пропорційний часу.

У групі блоків ***Model &Subsystem Inputs*** передбачені 4 блоки, які забезпечують використовування в моделі даних, отриманих раніше. Перший з них – ***In1*** є вхідним портом, завдяки якому можна завести у модель сигнал, отриманий в іншій моделі. Завдяки блоку ***Ground*** (Земля) можна ввести порожній (нульовий) сигнал у модель. Блок ***From File*** призначений для введення в S-модель

даних, що зберігаються на диску в MAT-файлі. Нарешті блок ***From Workspace*** забезпечує введення в модель даних безпосередньо з робочого простору MatLAB. Нагадаємо, що структура даних у MAT-файлі є багатовимірним масивом із змінною кількістю рядків, яка визначається кількістю змінних, що реєструються. Елементи першого рядка містять послідовні значення модельного часу, елементи в інших рядках – відповідні окремим елементам записаного вектора значення змінних.

Як і інші блоки бібліотеки SimuLink, блоки-джерела можуть настроюватися користувачем, за винятком блока ***Clock***, робота якого ґрунтується на використанні апаратного таймера комп'ютера.

Блок ***Constant***

Блок призначений для генерування процесів, що є незмінними у часі, тобто мають стале значення. Він має один параметр настроювання (рис. 5.16) – *Constant value*, який може бути введений і як вектор-рядок з кількох елементів по загальних правилах MatLAB. Приклад його використання наведений раніше при розгляді блока ***Display***.

Блок ***Signal Generator***

Вікно настроювання цього блока виглядає так, як показано на рис. 5.21. Як видно, у параметри настроювання входять:

- *Wave form* – форма хвилі – дозволяє обрати одну з таких форм періодичного процесу
  1) *Sine* – синусоїдальні хвилі;
  2) *Square* – прямокутні хвилі;
  3) *Sawtooth* – трикутні хвилі;
  4) *Random* – випадкові коливання;

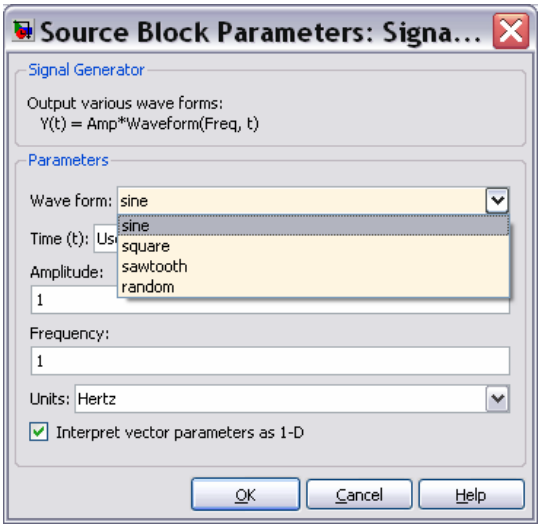


Рис. 5.21. Вікно настроювання блоку Signal Generator

- *Amplitude* – визначає значення амплітуди коливань, що генеруються;
- *Frequency* – задає частоту коливань;

■ *Units* – дозволяє обрати одну з одиниць виміру частоти за допомогою в спадного меню – *Hertz* (у герцах) і *Rad/Sec* (у радіанах у секунду).

На рис. 5.22 показано найпростішу блок-схему S-моделі, що складається з блоку ***Signal Generator*** і оглядового блоку ***XYGraph***.

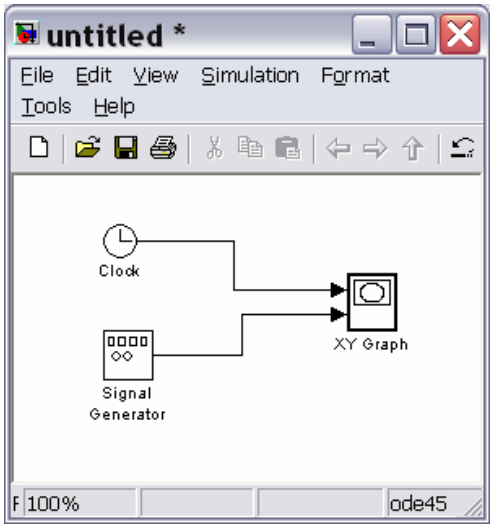
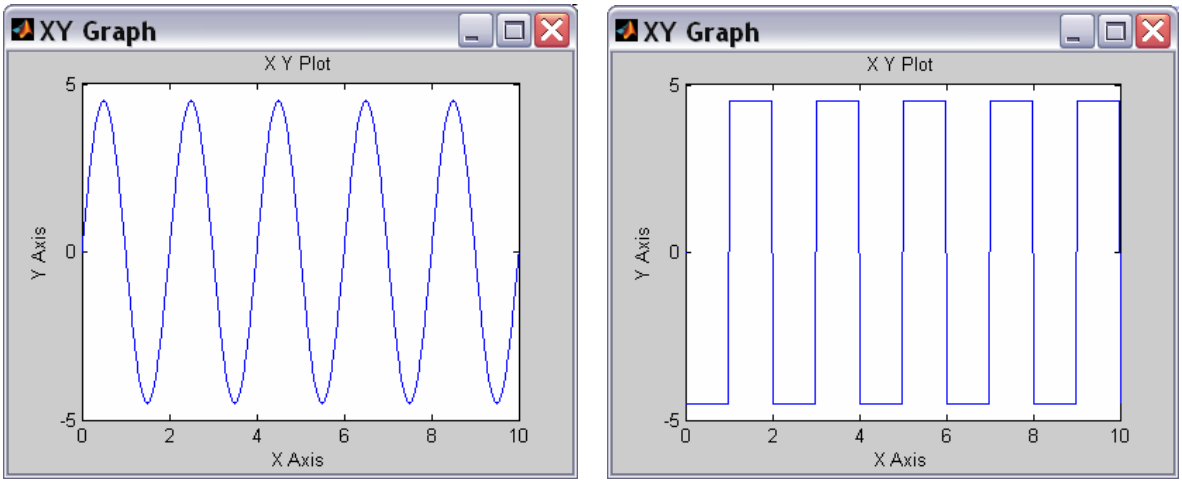


Рис. 5.22. Блок-схема перевірки функціонування блоку Signal Generator

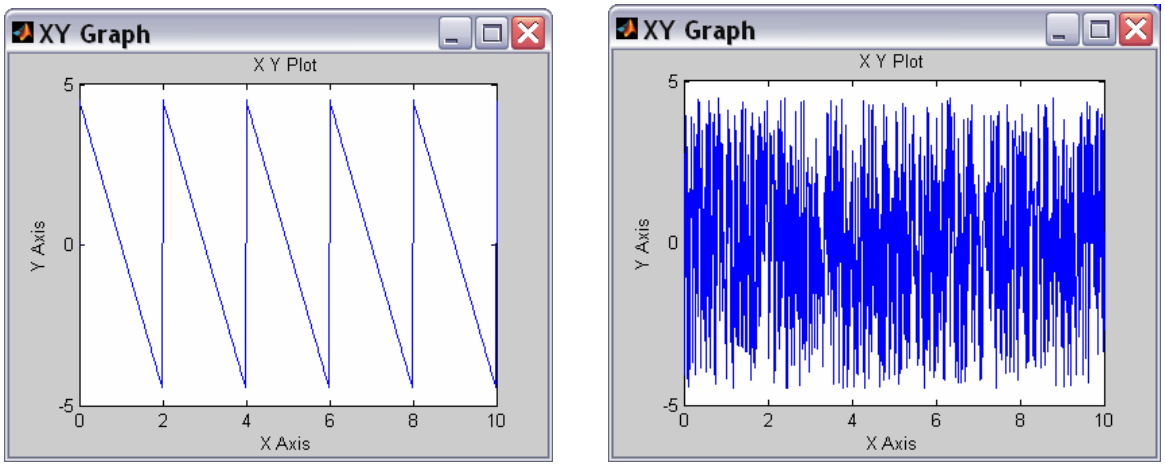
На наступному рис. 5.23 а) відображений вміст блоку ***XY Graph*** після проведення моделювання при таких параметрах настроювання: вид коливань – *Sine*; амплітуда – 4,5; частота – 0,5 Гц. Рис. 5.23б відбиває результат генерування прямокутної хвилі *Square*.



а) б)

Рис. 5.23. Результат генерування хвиль Sine (а) і Square (б)

На рис. 5.24 подані результати, відображені у вікні ***XYGraph*** у випадку обрання відповідно трикутних і випадкових хвиль за решти тих самих параметрів настроювання.



а) б)

Рис. 5.24. Результат генерування хвиль Sawtooth (а) і Random (б)

При обранні пункту *Random* у списку *Wave Form* генерується послідовність даних (сигнал), значення яких рівномірно випадково розподілені у діапазоні, вказаному у параметрі *Amplitude*, а значення моментів часу, у які здійснюються стрибкоподібні змінювання сигналу, відділені один від одного на величину кроку моделювання, встановлюваного командою Simulation > Configuration Parameters.

Блок ***Pulse Generator***

Блок генерує послідовності прямокутних імпульсів. У число параметрів цього блока, що настроюються, входять (рис. 5.25):

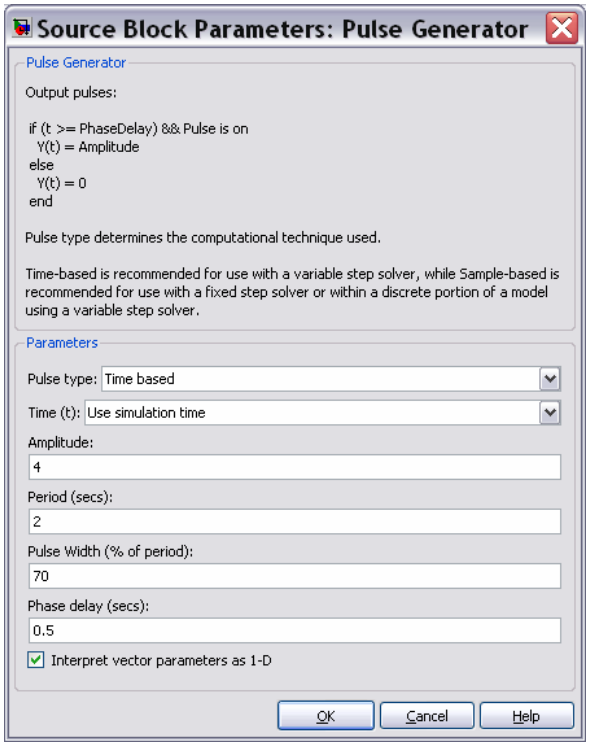


Рис. 5.25. Вікно настроювання блоку Pulse Generator

- тип імпульсів (*Pulse Type*); *Time based* (для неперевного сигналу, аргумент – час), *Sample based* (для дискретного часу, аргумент – кількість дискретів часу);
- амплітуда сигналу (*Amplitude*), тобто висота прямокутного імпульсу;
- розмір періоду сигналу (*Period*) у секундах;
- ширина імпульсу (*Pulse width*), у відсотках до періоду;
- розмір затримки імпульсу щодо *t*=0 (*Phase delay*) – у секундах.

Приклад застосування цього блока при значеннях параметрів, зазначених на рис. 5.25, наведений на рис. 5.26.

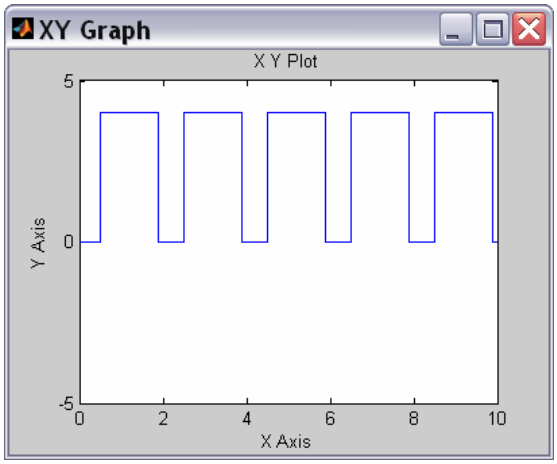


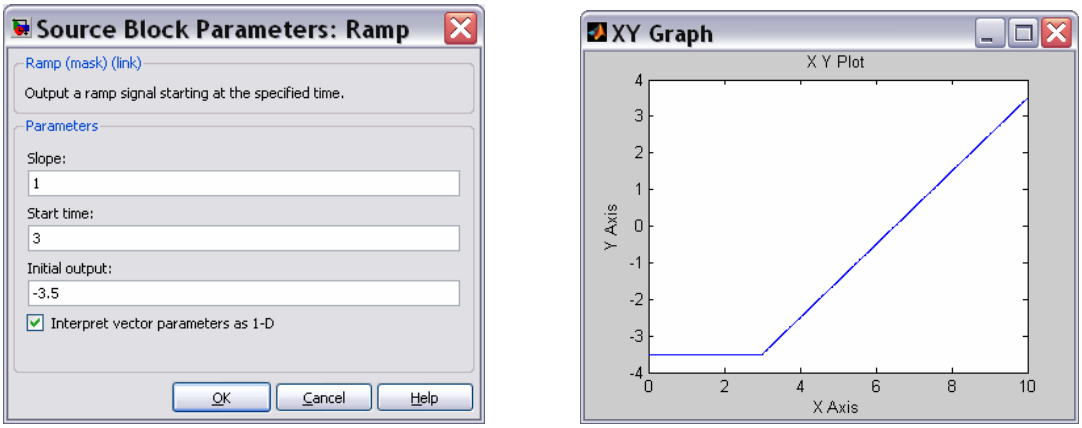
Рис. 5.26. Результат генерування послідовності прямокутних імпульсів блоком Pulse Generator

Блок ***Signal Builder***

Блок призначений для утворення поодинокого сигналу довільної форми, яка складається з відрізків прямих ліній.

Блок ***Ramp***

Цей блок формує неперервно висхідний сигнал і має такі параметри настроювання (рис. 5.27а):



а) б)

Рис. 5.27. Вікно настроювання (а) і результат роботи (б) блоку Ramp

■ *Slope* – значення швидкості сходження сигналу (тангенса кута нахилу прямої графіка залежності сигнала від часу);
■ *Start time* –час початку сходження сигналу;
■ *Initial output* – значення сигналу до моменту початку його сходження.

На рис. 5.27б наведений приклад результату застосування блока ***Ramp*** при таких значеннях параметрів: *Slope* = 1; *Start time* = 3; *Initial output* = -3.5.

*Блок **Sine Wave***

Цей блок є генератором гармонічно змінюваного з часом сигнала.

Блок ***Sine Wave*** має такі настройки (рис. 5.28):

■ *Sine Type* – тип синусоїдальної хвилі: *Time based* –для неперевного сигналу (аргумент – час); *Sample based* – для дискретного часу (аргумент – кількість дискретів часу);
■ *Amplitude* – амплітуда синусоїдального сигналу;
■ *Bias* – зміщення (стала складова сигналу, середнє його значення);
■ *Frequensy (rad/sec)* – частота коливань у радіанах у секунду;
■ *Phase (rad)* – початкова фаза в радіанах;
■ *Sample time* – визначає величину дискрету часу для цього блоку.

Результат застосування блока (за значень параметрів настроювання: амплітуда – 2,5; стала складова сигналу – (-1); частота – 1 радіан у секунду і початкова фаза - $\pi/2$ радіан) показаний на рис. 5.29.

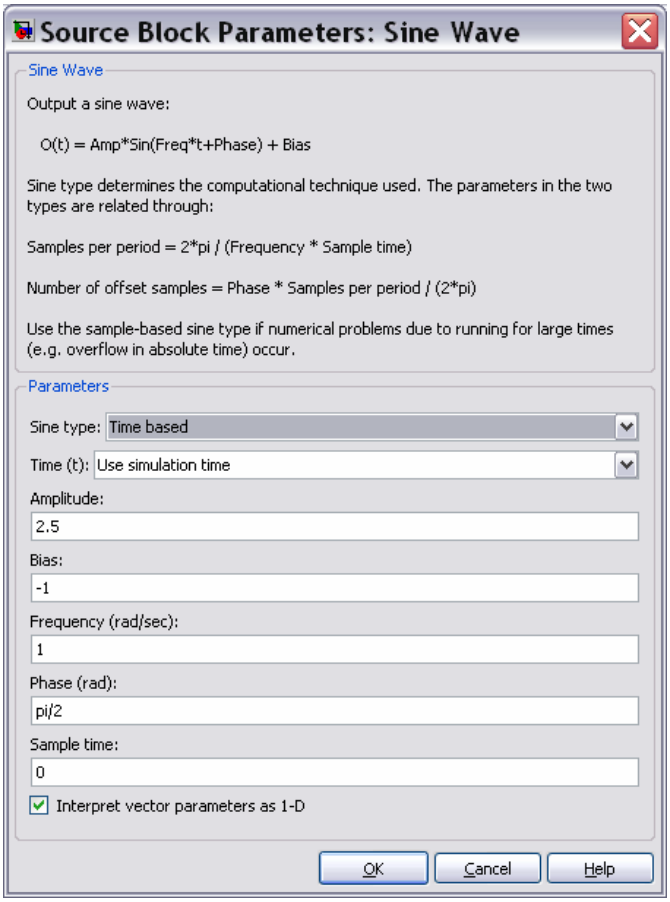


Рис. 5.28. Вікно настроювання блоку Sine Wave

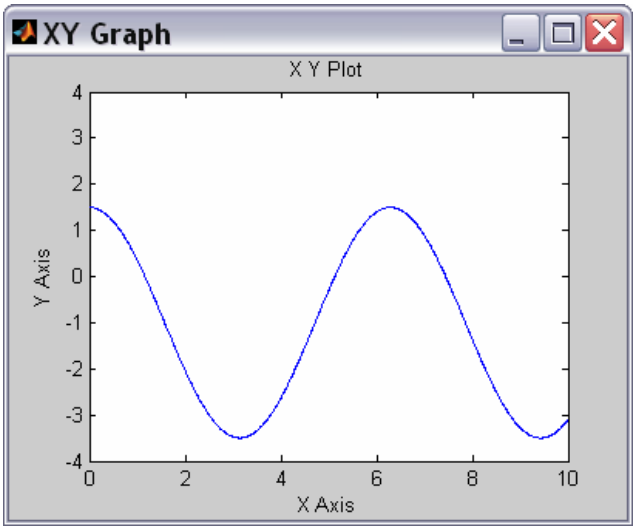


Рис. 5.29. Результат роботи блоку Sine Wave

*Блок **Step***

Блок забезпечує формування керувального сигналу у формі сходинки (або, як говорять, східчастого сигналу).

Блок має 3 параметри настроювання (рис. 5.30):

- *Step time* – час початку сходинки (момент стрибка сигналу) – визначає момент часу, у який відбувається стрибкоподібне змінювання сигналу; за замовчуванням приймається рівним 1;
- *Initial value* – початкове значення – задає рівень сигналу до стрибка; значення за замовчуванням – 0;
- *Final value* – кінцеве значення – задає рівень сигналу після стрибка; значення його за замовчуванням – 1.

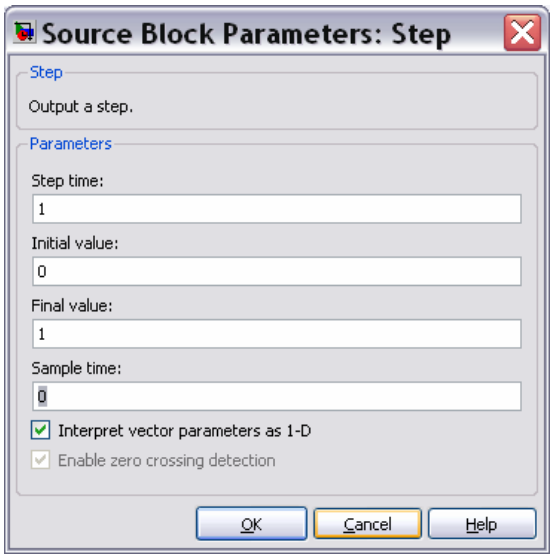


Рис. 5.30. Вікно настроювання блоку Step

Якщо встановити такі параметри настроювання блока: *Step time* – 3,5, *Initial value* – (-2), *Final value* – 3, то після активізації моделювання (*Simulation/Start)* у вікні ***Scope*** виходить картина, поданп на рис. 5.31.

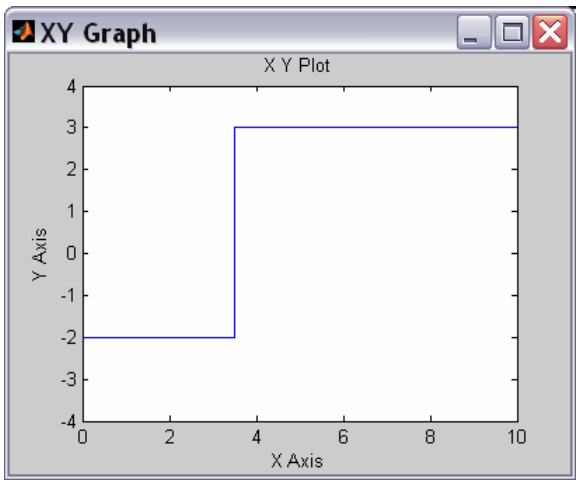


Рис. 5.31. Результат генерування східчастого сигналу

*Блок **Repeating Sequence***

Цей блок являє собою генератор періодичних коливань, хвиля яких складається з відрізків прямих. У вікнімістить дві настройки (рис. 5.32):

- *Time values* – вектор значень часу, в які задані значення вихідної величини;
- *Output values* – вектор значень вихідної величини, які вона повинна прийняти в зазначені в першому векторі відповідні моменти часу.

Блок забезпечує генерування коливань із періодом, рівним різниці між останнім значенням вектора *Time values* і значенням першого його елемента. Форма хвилі усередині періоду є ламаною, що проходить через точки із зазначеними у векторах *Time values* і *Output values* координатами.

Як приклад на рис. 5.33 наведене зображення процесу, згенерованого блоком ***Repeating Sequence*** при параметрах настроювання, зазначених на рис. 5.32.

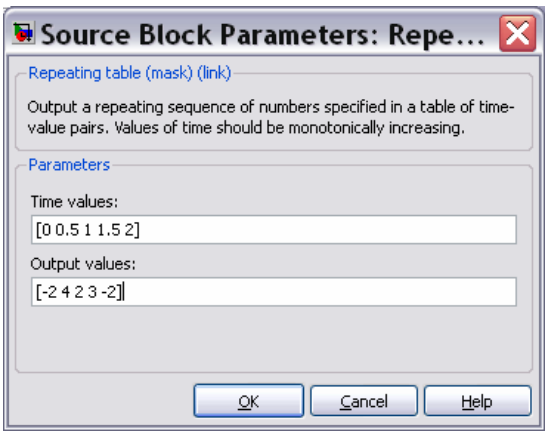


Рис. 5.32. Вікно настроювання блоку Repeating Sequence

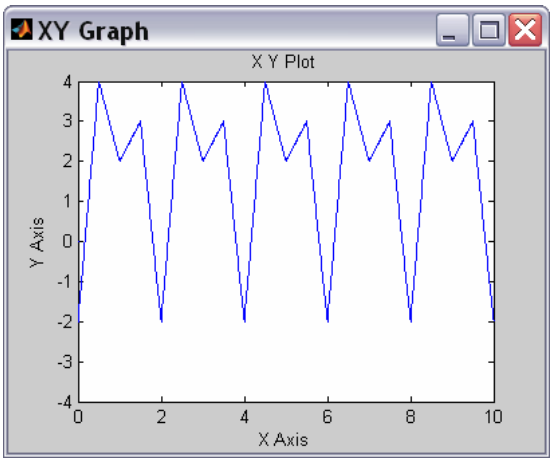


Рис. 5.33. Сигнал, згенерований блоком Repeating Sequence

Блок Chirp Signal

Цей блок генерує синусоїдальний сигнал одиничної амплітуди змінної частоти, причому значення частоти коливань змінюється з часом за лінійним законом. Відповідно до цього в ньому передбачені такі параметри настроювання (рис. 5.34):

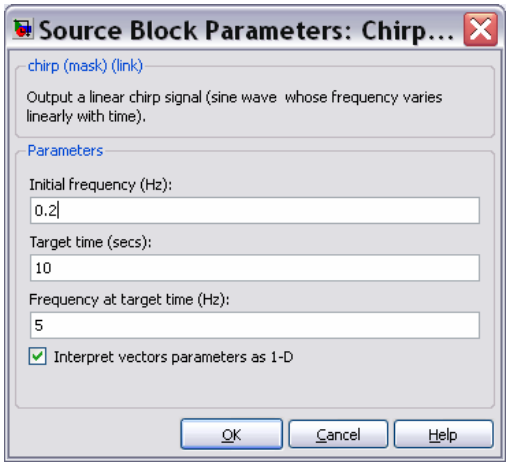


Рис. 5.34. Вікно настроювання блоку Chirp Signal

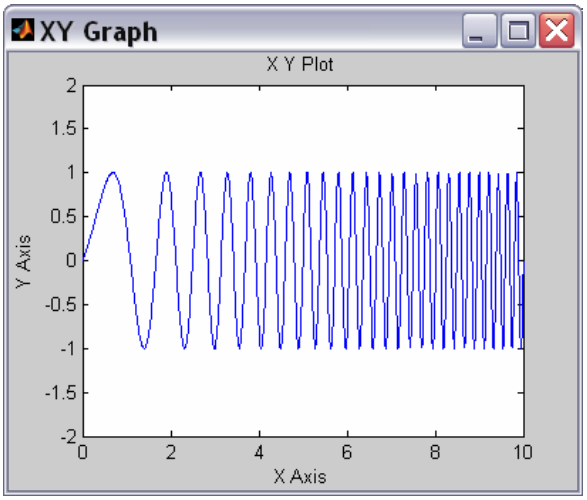


Рис. 5.35. Сигнал, згенерований блоком Chirp Signal

- *Initial frequency (Hz)* – початкове значення (при t=0) частоти в герцах;
- *Target time (secs)* – другий (додатний) момент часу (у секундах);
- *Frequency at target time (Hz)* – значення частоти (у герцах) в цей другий момент часу.

Рис. 5.35 демонструє результат використання блока при параметрах, визначених на рис. 5.34.

Блоки, що генерують випадкові процеси

Блок ***Random Number*** забезпечує формування сигналів, значення яких в окремі моменти часу є випадковою величиною, розподіленою за нормальним (гауссовим) законом із заданими параметрами.

Блок має чотири параметри настроювання (рис. 5.36). Перші два – *Mean* і *Variance* – є параметрами нормального закону (середнє й середньоквадратичне відхилення від цього середнього), третій – *Initial seed* – задає початкове значення бази для ініціалізації генератора послідовності випадкових чисел. При фіксованому значенні цього параметра генератор завжди виробляє ту саму послідовність. Четвертий параметр (*Sample time*), як і раніше, задає величину дискрету часу.

Блок ***Uniform Random Number*** блок формує сигнали, значення яких в окремі моменти часу є випадковою величиною, що рівномірно розподілена в заданому інтервалі. У число параметрів настроювання блока входять:

- *Minimum* – нижня межа випадкової величини;
- *Maximum* – верхня межа;
- *Initial seed* – початкове значення бази генератора випадкових чисел;
- *Sample time* – дискрет часу.

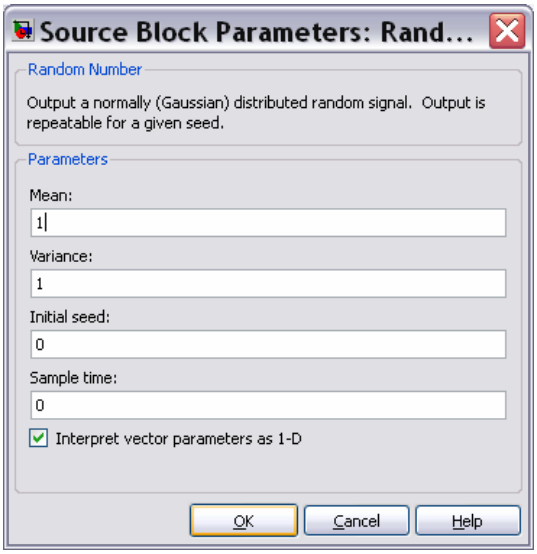


Рис. 5.36. Вікно настроювання блоку Random Number

Блок ***Band-Limited White Noise*** формує процес у виді частотно-обмеженого білого шуму. Параметри настроювання в нього такі:

- *Noise power* – значення потужності білого шуму;

■ *Sample time* – значення дискрету часу (визначає верхнє значення частоти процесу);

■ *Seed* – початкове значення бази генератора випадкової величини.

Сформуємо блок-схему, яка ілюструє роботи блоків. Для того, щоб у вікні *Scope* можна було розмістити три графіки, кожен з яких відбивав би окремий процес, необхідно викликати команду *Scope > 'Scope' parameters > General* і встановити у полі *Number of ax*es кількість осей – 3. Вид блока ***Scope*** на блок-схемі при цьому зміниться, на ньому виникнуть зображення трьох входів. У списку *Tick Labels* встановимо значення *all*. Після цього над кожним з трьох графіків, які виводяться, можна буде проставити заголовки. Тексти заголовків розміщуються на лініях, що ведуть до входів блоку ***Scope*** (див. рис. 5.37)

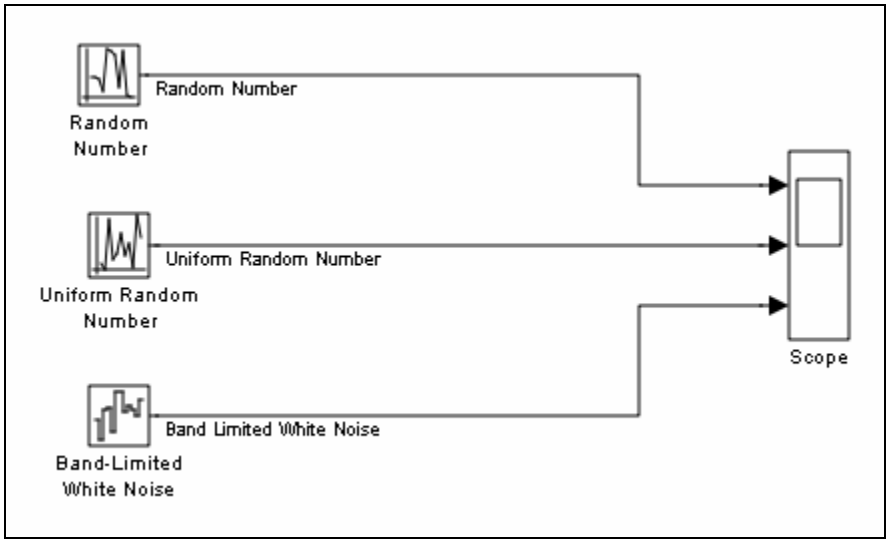


Рис. 5.37. Блок-схема перевірки роботи генераторів випадкових процесів

Перед запуском моделі у вікні блок-схеми викличемо команду *Simulation > Configuration parameters > Solver*, у віконці якого зі спадним списком оберемо значення discrete (no continuous state). У тому самому вікні у віконці з написом *Fixed step size* (Розмір фіксованого кроку) встановимо 0,05.

У вікнах настроювання усіх трьох блоків встановимо дискрет часу (*Sample time*), рівним 0,1.

Здійснюючи тепер моделювання, одержимо процеси, подані на рис. 5.38.

Як бачимо, модельовані процеси зберігають незмінні значення всередині інтервалу часу *Sample time* (Дискрета часу). Змінювання значень процеса здійснюється стрибкоподібно на межі сусідніх дискретів часу.

Примітка. Слід відрізняти крок змінювання модельного часу, який встановлюється при проведенні моделювання у вікні *Configuration parameters* і визначає інтервали часу, через які здійснюються обчислення окремих станів системи, що моделюється, і параметр *Sample time* (Дискрет часу), який визначає інтервал часу, всередині якого вихідна величина блоку не змінює свого значення.

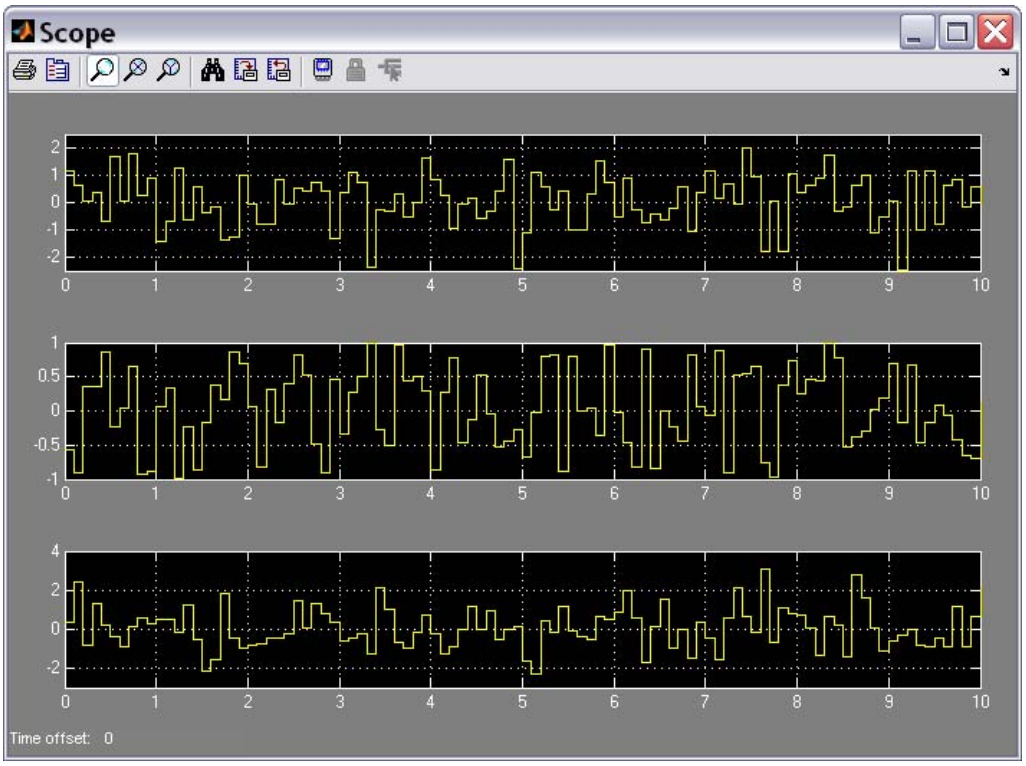


Рис. 5.38. Результати роботи генераторів випадкових процесів при Sample time=0,1

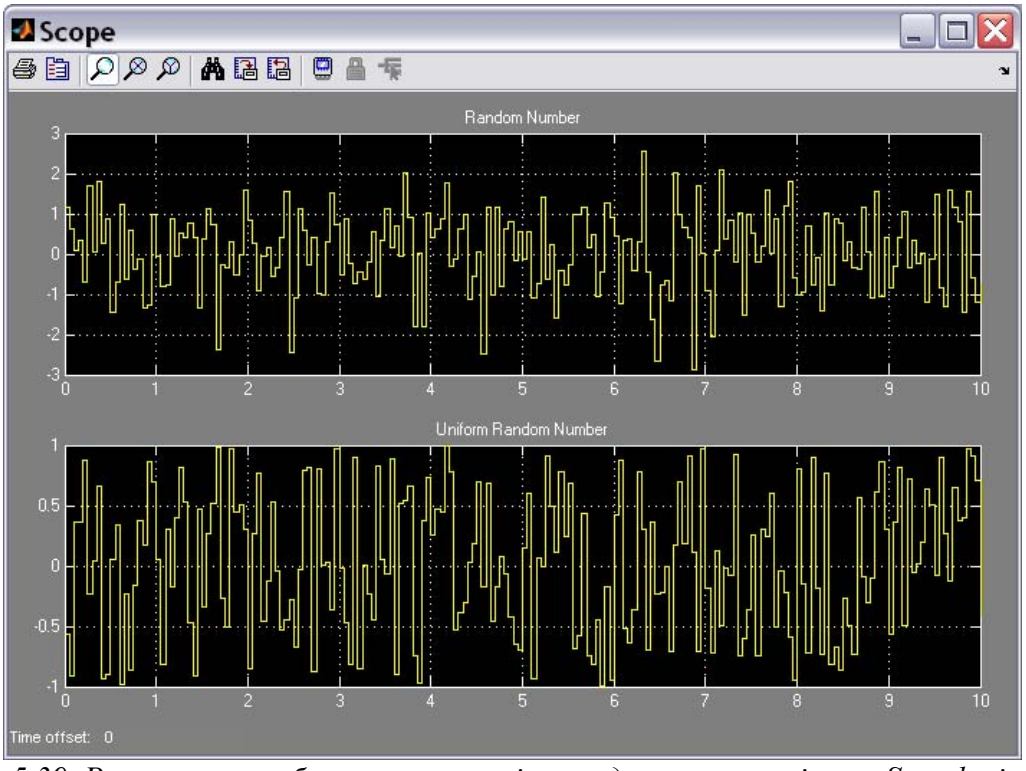


Рис. 5.39. Результати роботи генераторів випадкових процесів при Sample time=0

Варто зазначити, що блок ***Band-Limited White Noise*** суттєво відрізняється від перших двох блоків-генераторів. Для останніх встановлення значення параметра *Sample time*, відмінного від нуля, не є обов'язковим. У блоці ***Band-Limited White Noise*** цей параметр визначає найбільшу частоту у спектрі вихід-

ного сигналу, вона дорівнює величині (у герцах), обернену значенню параметра *Sample time*, тому останній не може бути рівний нулю.

На рис. 5.39 показаний результат моделювання перших двох блоків при значенні параметра *Sample time,* рівним нулю. У цьому випадку змінювання величини випадкового процесу здійснюється вже на стику кроків моделювання (0,05), величина яких встановлюється за допомогою команди *Simulation > Configuration parameters* у віконці з написом *Fixed step size* (Розмір фіксованого кроку).

5.2.3. Поділ Continuous (Неперервні елементи)

Цей поділ бібліотеки містить наступні групи блоків (рис. 5.40):

– ***Continuous-Time Linear Systems*** (Лінійні системи неперервного часу);

– ***Continuous-Time Delays*** (Затримки неперервного часу).

Перша група складається з блоків:

- ***Integrator*** – ідеальна інтегрувальна ланка (інтегратор);
- ***Derivative*** – ідеальна диференціювальна ланка;
- ***State-Space*** – визначення лінійної ланки через завдання чотирьох матриць її простору станів;
- ***Transfer Fcn*** – визначення лінійної ланки через завдання її передатної функції;
- ***Zero-Pole*** – завдання ланки через указівку векторів значень його полюсів і нулів, а також значення коефіцієнта передачі;

Використання більшості блоків-ланок є досить прозорим для тих, хто знайомий з основами теорії автоматичного керування.

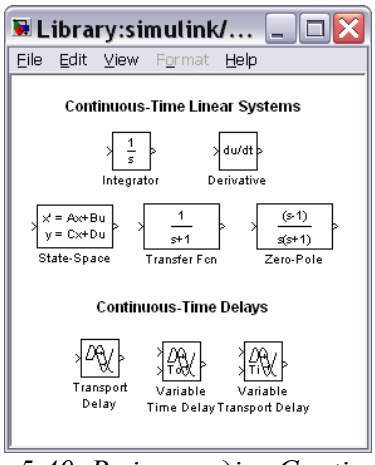


Рис. 5.40. Вміст поділу Continuous

Блок Integrator

Блок здійснює інтегрування в неперервному часі вхідної величини. Він має наступні параметри настроювання (рис. 5.41):

- підключення додаткового керувального сигналу (*External reset*);
- визначення джерела (внутрішнє чи зовнішнє) встановлювання початкового значення вихідного сигналу (*Initial condition source*);

- початкове значення вихідної величини (*Initial condition*); значення вводиться в рядку редагування або як числова константа, або у вигляді виразу, що обчислюється;
- прапорець *Limit output* (*Обмеження вихідного значення)* визначає, чи використовуватимуться наступні 2 параметри настроювання;

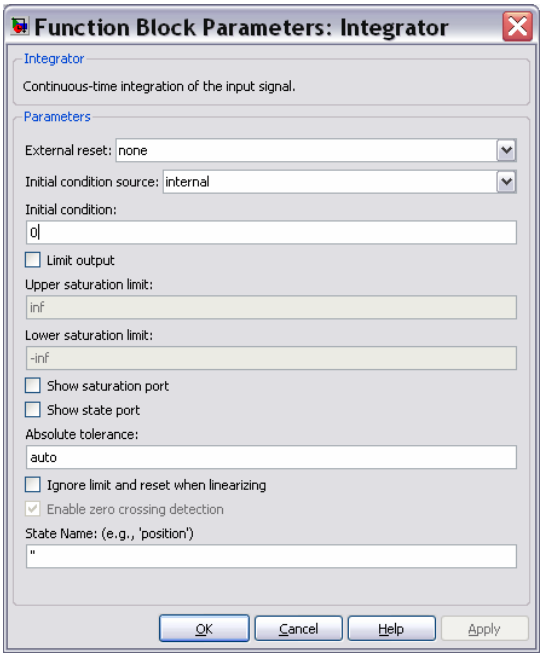


Рис. 5.41. Вікно настроювання блоку Integrator

- верхнє граничне значення вихідної величини (*Upper saturation limit*); за замовчуванням – не обмежене (*inf*);
- нижнє граничне значення вихідної величини (*Lower saturation limit*); за замовчуванням параметр має значення (*-inf*);
- прапорець *Показати порт насичення* (*Show saturation port*);
- прапорець *Показати порт стану* (*Show state port*);
- припустима гранична величина абсолютної похибки (*Absolute tolerance*).

Параметр *External reset* може приймати такі значення (див. його спадне меню): *none* – додатковий керувальний сигнал не використовується; *rising* – для керування використовується висхідний сигнал; *falling* – для керування використовується спадний сигнал; *either* – для керування використовується і висхідний і спадний сигнал.

Параметр *Initial condition source* набуває одного з двох значень:

internal – використовується внутрішнє встановлювання початкового значення вихідної величини;

external – встановлення початкових умов здійснюватиметься ззовні.

Якщо обрані користувачем значення цих двох параметрів припускають наявність додаткових вхідних сигналів, то на графічному зображенні блока виникають додаткові вхідні порти (після натискання кнопки *Apply* у вікні настроювань блока). Якщо прапорець *Limit output* встановлений, то при переході вихі-

дного значення інтегратора через верхню або нижню межу на додатковому виході блока (*saturation port*) формується одиничний сигнал. Щоб цей сигнал можна було використовувати для керування роботою S-моделі, прапорець *Show saturation port* має бути включений. При цьому на графічному зображенні блока виникає позначення нового вихідного порту на правій стороні зображення блоку-інтегратора.

Встановлення прапорця *Show state port* також приводить до появи додаткового виходу *state port* блока (він виникає зазвичай на нижній стороні зображення блока). Сигнал, який подається на цей порт, збігається з головним вихідним сигналом, але, на відміну від нього, може бути використаний тільки для переривання алгебричного циклу або для узгодження стану підсистем моделі.

*Блок **State-Space** (простір станів)*

Блок ***State-Space*** забезпечує введення лінійної стаціонарної ланки перетворення вхідного сигналу у виді матриць у просторі станів. Його вікно настроювання (рис. 5.42) містить 7 параметрів настроювання:

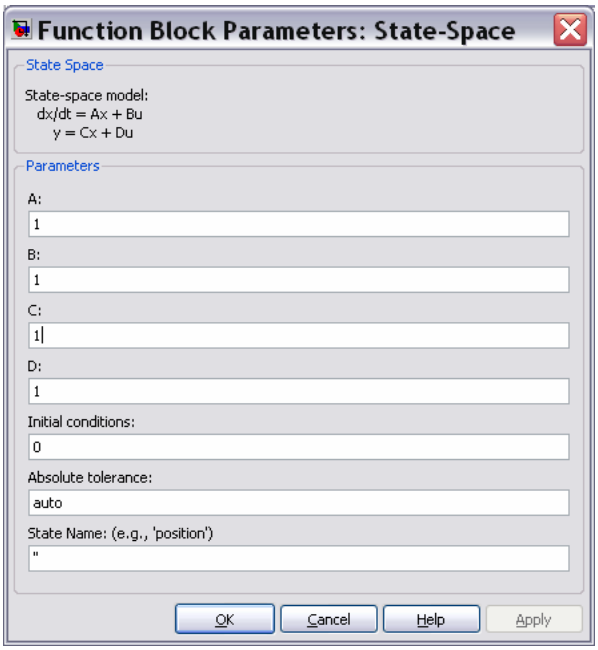


Рис. 5.42. Вікно настроювання блоку State-Space

– матриця A системи зв’язку похідних від змінних стану з самими змінними стану;
– матриця B зв’язку похідних від змінних стану з вхідним сигналом блоку;
– матриця C зв’язку вихідного сигналу зі змінними стану;
– матриця D зв’язку вихідного сигналу з вхідним сигналом;
– вектор *Initial conditions* початкових значень вектору змінних стану;
– вектор Absolute tolerance максимально припустимих похибок обчислення кожної зі змінних стану; за замовчуванням встановлюється автоматично;

–State Name – символьний рядок, що містить ймення стану системи.

*Блок **Transfer Fcn** (передатна функція)*

Блок дозволяє ввести лінійну ланку через завдання її передатної функції. Вікно настроювання показане на рис. 5.43 і містить два основних параметри:

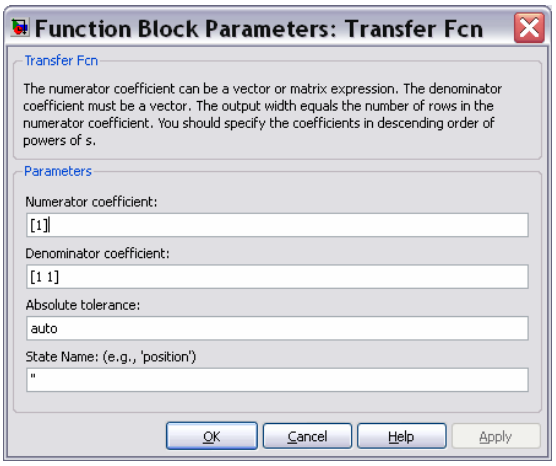


Рис. 5.43. Вікно настроювання блоку Transfer Fcn

- *Numerator coefficients* – вектор значень коефіцієнтів чисельника передатної функції;
- *Denominator coefficients* – вектор значень коефіцієнтів знаменника передатної функції.

*Блок **Zero-Pole** (нулі – полюси)*

Блок ***Zero-Pole*** дозволяє ввести лінійну ланку через завдання нулів і полюсів її передатної функції.

Головні параметри настроювання блоку є такими (рис. 5.44):

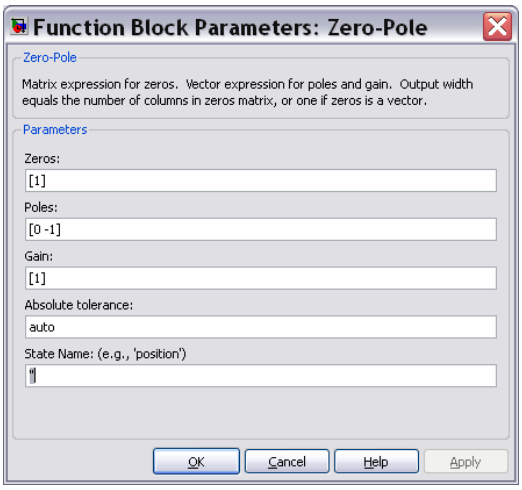


Рис. 5.44. Вікно настроювання блоку Zero-Pole

– *Zeros* – вектор нулів передатної функції (коренів поліному її чисельника);

– *Poles* – вектор полюсів передатної функції (коренів поліному її знаменника);

– *Gain* – вектор коефіцієнтів підсилення.

Друга група блоків здійснює затримку неперервних сигналів, що поступають на їхній вхід.

Блок ***Transport Delay*** забезпечує затримку сигналу на задану кількість кроків модельного часу, причому необов'язково ціле. Настроювання блока відбувається по трьох параметрах:

Time delay (Час затримки) – кількість кроків модельного часу, на який слід затримати сигнал; може вводитися або в числовій формі, або у формі виразу, що обчислюється;

Initial input (Початкове значення входу) – за замовчуванням дорівнює 0;

Initial buffer size (Початковий розмір буфера) – обсяг пам'яті (у байтах), що виділяється в робочому просторі MatLAB для збереження параметрів затриманого сигналу; має бути кратним до 8 (за замовчуванням – 1024).

Блок ***Variable Transport Delay*** дозволяє задавати керовану ззовні величину затримки. З цією метою блок має додатковий вхід. Подаваний на нього сигнал визначає тривалість затримки.

5.2.4. Поділ Discrete (Дискретні елементи)

Раніше розглянуті розділи бібліотеки дозволяють формувати неперервну динамічну систему. Розділ ***Discrete*** містить елементи (блоки), властиві тільки дискретним системам, а також такі, що перетворюють неперервну систему в дискретну (рис. 5.45). Ці блоки поділені на дві групи

– ***Discrete-Time Linear Systems*** (Лінійні системи дискретного часу);

– ***Sample & Hold Delays*** (Затримки дискретного часу).

У першу групу входять блоки:

- ***Unit Delay, Integer Delay, Tapped Delay*** – блоки, яки використовуються для затримки сигналу;
- ***Discrete-Time Integrator*** – дискретний інтегратор;
- ***Discrete Transfer Fcn*** –блок завдання лінійної дискретної ланки через дискретну передатну дробово-раціональну функцію щодо z;
- ***Discrete Filter*** – блок завдання дискретної ланки через дискретну передатну дробово-раціональну функцію щодо 1/z;
- ***Discrete Zero-Pole*** – блок завдання дискретної ланки через задання значень нулів і полюсів дискретної передатної функції щодо 1/z.
- ***Discrete Derivative*** – формує дискретну похідну вхідного сигналу

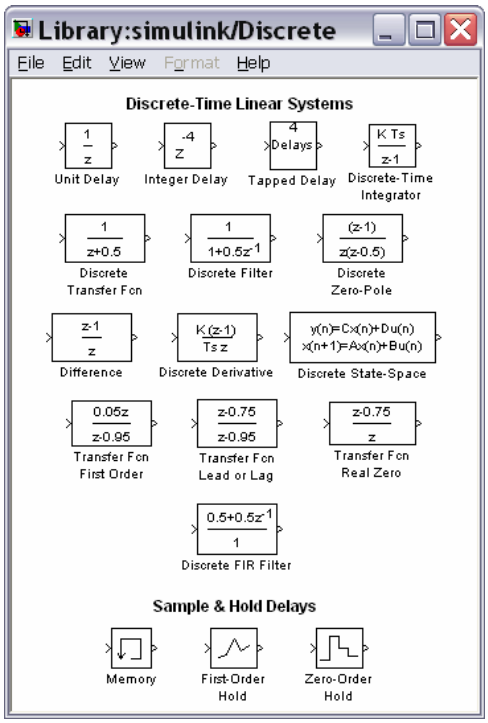


Рис. 5.45. Вміст поділу Discrete

- ***Discrete State-Space*** – блок завдання дискретної лінійної ланки матрицями її стану;
- ***Transfer Fcn First Order, Transfer Fcn Lead or Lag, Transfer Fcn Real Zero*** – блоки, що формують дискретні ланки з спеціальними передатними функціями;
- ***Discrete FIR Filter*** – формує дискретний КІХ –фільтр.

Другу групу складають блоки:

- ***Memory*** – використовується для затримки сигнала на один крок модельного часу;
- ***First-Order Hold*** – екстраполятор першого порядку;
- ***Zero-Order Hold*** – екстраполятор нульового порядку;

Блок ***Unit Delay*** забезпечує затримку вхідного сигналу на задане число кроків модельного часу. Параметрами настроювання для цього блока є: початкове значення сигналу (*Initial condition)* і час затримки *(Sample time)*, який задається кількістю кроків модельного часу.

Блок ***Discrete-Time Integrator*** виконує чисельне інтегрування вхідного сигналу. Більшість параметрів настроювання цього блока збігаються з параметрами блока ***Integrator*** розділу ***Continuous***. Відмінності полягають у наступному. У блоці дискретного інтегратора є додатковий параметр – метод чисельного інтегрування (*Integrator method*). За допомогою спадного меню можна вибрати один із трьох методів: прямий метод Ейлера (лівих прямокутників); зворотний метод Ейлера (правих прямокутників); метод трапецій. Друга відмінність – замість параметра *Absolute tolerance* уведений параметр *Sample time,* який задає крок інтегрування в одиницях кроків модельного часу.

Блок ***Memory*** *(Пам'ять)* виконує затримку сигналу тільки на один крок модельного часу. Блок має два параметри настроювання: *Initial condition (Початкова умова)* задає значення вхідного сигналу в початковий момент часу; прапорець *Inherit sample time (Спадкування кроку часу)* дозволяє вибрати величину проміжку часу, на який буде здійснюватися затримка сигналу:

- якщо прапорець знятий, то використовується мінімальна затримка, рівна 0,1 одиниці модельного часу;
- якщо прапорець встановлений, то величина затримки дорівнює значенню дискрету часу блока, що передує блоку ***Memory.***

Блоки ***First-Order Hold*** і ***Zero-Order Hold*** прислуговуються задля перетворення неперервного сигналу у дискретний.

На рис. 5.46. графічно поданий результат проходження неперервного синусоїдального сигналу через ці два блоки-екстраполятори.

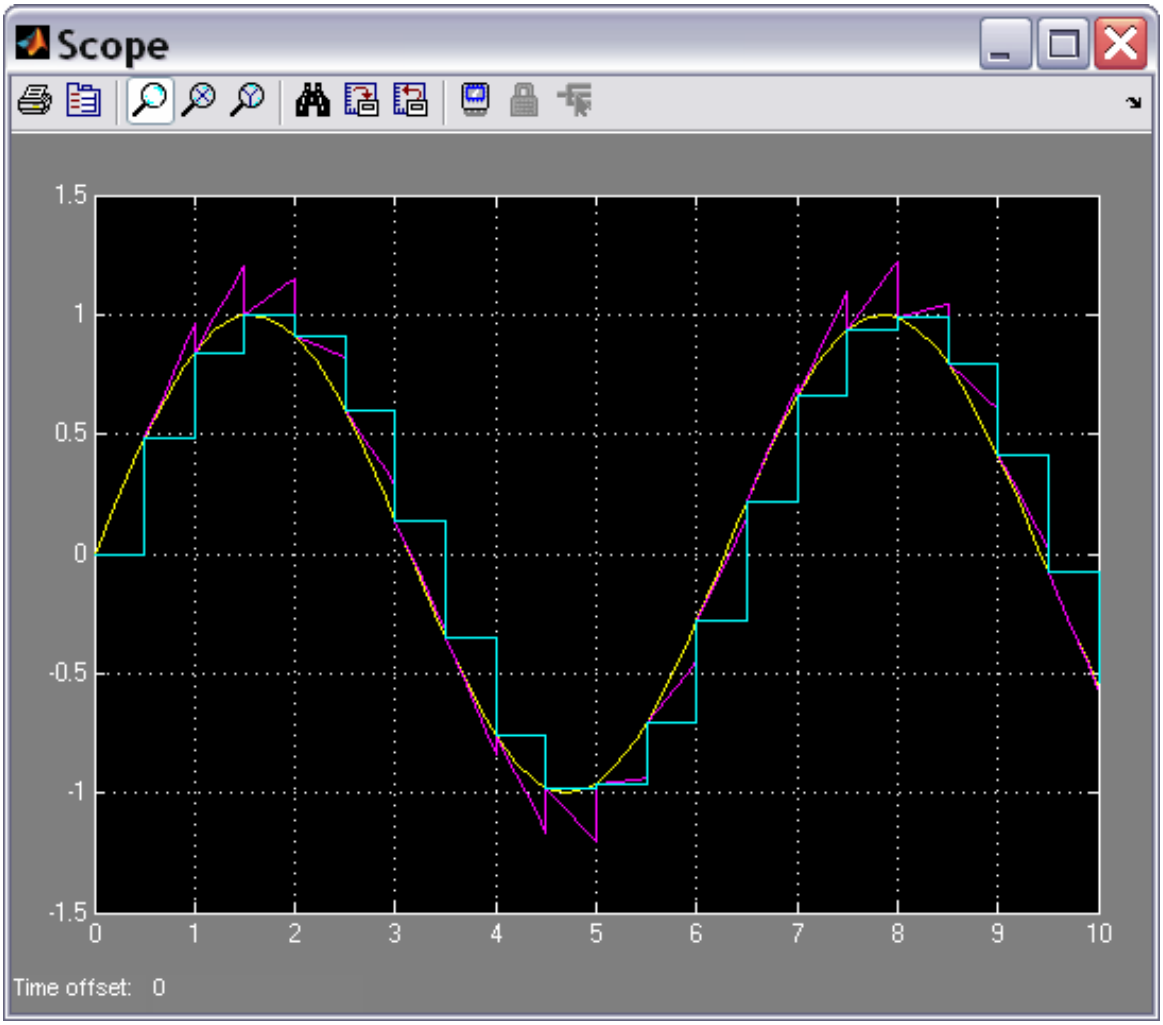


Рис. 5.46. Проходження синусоїдального сигналу через екстраполятори

5.2.5. Поділ Discontinuities (Розривні елементи)

Це, мабуть, найбільш корисний розділ бібліотеки ***SimuLink***. Він включає 12 блоків (рис. 5.47).

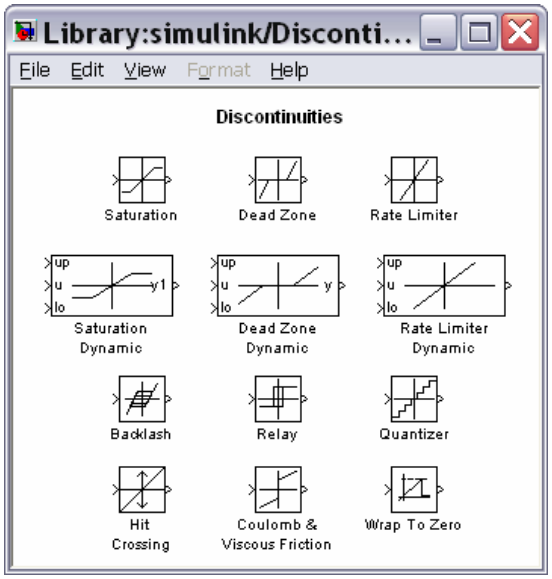


Рис. 5.47. Вміст поділу Discontinuities

Блок ***Saturation*** (*Насичення*) реалізує лінійну залежність із насиченням (обмеженням). Вихідна величина цього блока збігається із вхідною, якщо остання знаходиться усередині зазначеного діапазону. Якщо ж вхідна величина виходить за рамки діапазону, то вихідний сигнал приймає значення найближчої з меж. Значення меж діапазону встановлюються у вікні настроювання блока.

Блок ***Dead Zone*** *(Мертва зона)* реалізує нелінійність типу зони нечутливості. Параметрів настроювання тут два – початок і кінець зони нечутливості.

Блок ***Rate Limiter*** (*Обмежувач швидкості*) забезпечує обмеження згори і знизу швидкості змінювання сигналу, що проходить крізь нього. Вікно настроювання блоку містить два головних параметри: *Rising slew rate* і *Falling slew rate*. Блок працює у такий спосіб: спочатку обчислюється швидкість змінювання сигналу, що проходить кріз нього, за формулою

$$rate = \frac{u(i) - y(i-1)}{t(i) - t(i-1)},$$

де $u(i)$ – значення вхідного сигналу у момент часу $t(i)$; $y(i-1)$ – значення вихідного сигнала у момент $t(i-1)$. Далі, якщо обчислене значення *rate* перевищує значення параметра *Rising slew rate* ®, вихідна величина визначається за формулою:

$$y(i) = y(i-1) + R \cdot \Delta t.$$

Якщо *rate* є меншим за значення *Falling slew rate (F)*, вихідна величина розраховується так

$$y(i) = y(i-1) + F \cdot \Delta t.$$

У тому випадку, коли значення *rate* міститься проміж значень *R* і *F*, вихідна величина збігається зі вхідною

$$y(i) = u(i).$$

Блок ***BackLash*** *(Люфт)* реалізує нелінійність типу зазору. У ньому передбачено два параметри настроювання: *Deadband width* – величина люфту і *Initial output* – початкове значення вихідної величини.

Блок ***Relay*** *(Реле)* працює за аналогією зі звичайним реле: якщо вхідний сигнал перевищує деяке граничне значення, то на виході блока формується деякий сталий сигнал. Блок має 4 параметри настроювання:

- *Switch on point (Точка вмикання)* – задає граничне значення, при перевищенні якого відбувається вмикання реле;
- *Switch off point (Точка вимикання)* – визначає рівень вхідного сигналу, при якому реле вимикається;
- *Output when on (Вихід при включеному стані)* – установлює рівень вихідної величини при включеному реле;
- *Output when off (Вихід при виключеному стані)* – визначає рівень вихідного сигналу при виключеному реле.

Блок ***Quantizer*** *(Квантувач)* здійснює дискретизацію вхідного сигналу за його величиною. Єдиним параметром настроювання цього блока є *Quantization interval – Інтервал квантування* – розмір дискрету за рівнем вхідного сигналу.

Блок ***Hit Crossing*** *(Виявити перетинання)* дозволяє зафіксувати стан, коли вхідний сигнал "перетинає" деяке значення. При виникненні такої ситуації на виході блока формується одиничний сигнал. Блок має 3 параметри настроювання (рис. 5.48):

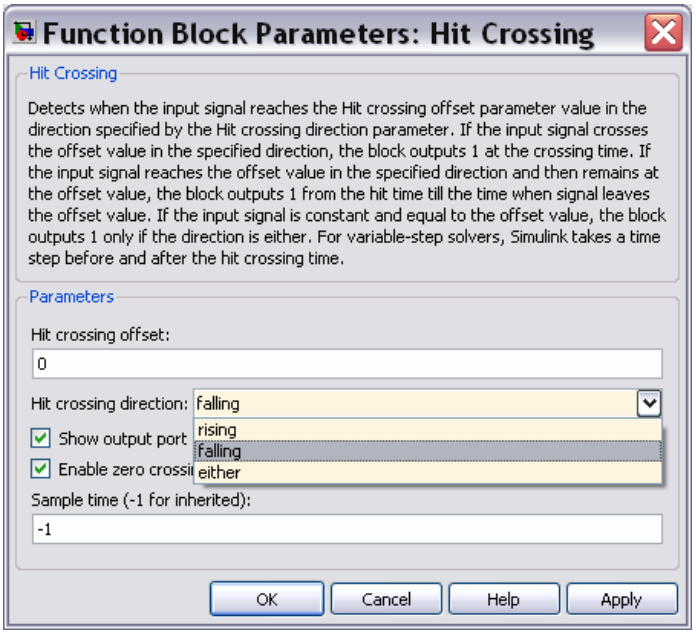


Рис. 5.48. Вікно настроювання блоку Hit crossing

- *Hit crossing offset* – визначає значення, перетинання якого необхідно ідентифікувати;

- *Hit crossing direction* дозволяє зазначити напрямок перетинання, при якому це перетинання повинно виявлятися; значення цього параметра вибирається за допомогою спадного меню, яке містить три альтернативи: *rising (сходження), falling (спадання), either (у будь-якому напрямку);*
- *Show output port (зазначити порт виходу)* – прапорець, за допомогою якого вибирається формат використання блока.

При одночасному виконанні умов, що задаються параметрами *Hit crossing offset* і *Hit crossing direction*, на виході блока формується одиничний сигнал. Його тривалість визначається значенням дискрету часу (параметр *Sample time*) блока, який передує в моделі блоку ***Hit crossing.*** Якщо цей параметр відсутній, то одиничний сигнал на виході блока існує до його наступного спрацьовування.

Блок ***Coulomb & Viscous Friction*** *(Кулонове і в'язке тертя)* реалізує нелінійну залежність типу "лінійна з натягом". Якщо вхід додатний, то вихід пропорційний входові з коефіцієнтом пропорційності – "коефіцієнтом в'язкого тертя" – і збільшений на величину "натягу" ("кулонове, сухе тертя"). Якщо вхід є від'ємним, то вихід також є пропорційним входові (із тим же коефіцієнтом пропорційності) за відрахуванням величини "натягу". При вході рівному нулю вихід теж дорівнює нулю. У параметри настроювання блока входять величини "кулонова тертя" ("натягу") і коефіцієнта "в'язкого тертя".

Блок ***Wrap To Zero*** (Скидання у нуль) забезпечує рівність нулеві вихідного сигналу, якщо значення вхідного сигналу перевищує встановлене значення єдиного параметра настроювання *Threshold (Поріг)*. Якщо ж вхідний сигнал менше за *Threshold* вихідний сигнал дорівнює вхідному.

5.2.6. Поділ Signal Routing (Пересилання сигналів)

Поділ бібліотеки ***Signal Routing*** призначений для побудови складних S-моделей, що складаються з інших моделей більш низького рівня. Склад блоків наведений на рис. 5.49.

Блок ***Mux*** *(Мультіплексор)* виконує об'єднання вхідних величин у єдиний вихідний вектор. При цьому вхідні величини можуть бути як скалярними, так і векторними. Довжина результуючого вектора дорівнює сумі довжин усіх векторів. Порядок елементів у векторі виходу визначається порядком входів (зверху униз) і порядком розташування елементів усередині кожного входу. Блок має один параметр настроювання – *Number of inputs (Кількість входів).*

Блок ***Demux*** *(Подільник, Демультіплексор)* виконує зворотну функцію – *поділяє вхідний вектор на задану кількість компонентів*. Він також має єдиний параметр настроювання *Number of outputs (Кількість виходів).* У випадку, коли зазначене число виходів (N) задається меншим довжини вхідного вектора (M), блок формує вихідні вектори в такий спосіб. Перші (N-1) виходів будуть векторами однакової довжини, рівної цілій частині відношення M/(N-1). Останній вихід буде мати довжину, рівну залишку від ділення.

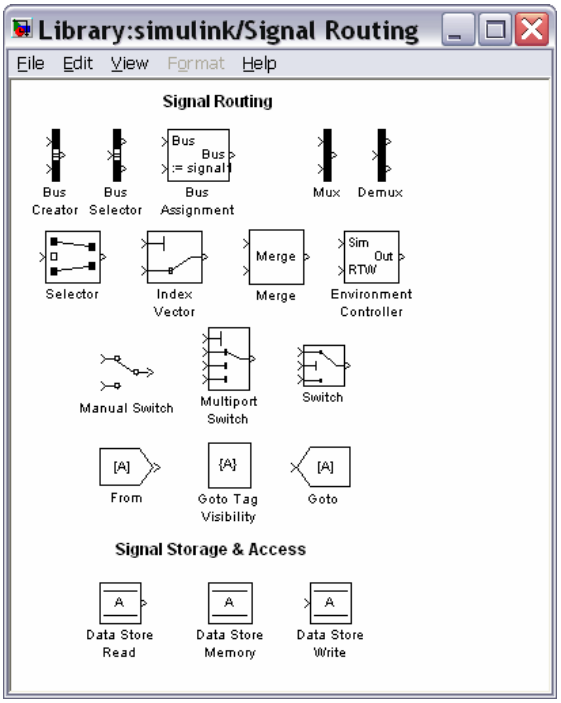


Рис. 5.49. Вміст поділу Signal Routing

Блоки ***Bus Creator*** (Утворювач шини) і ***Bus Selector*** (Роздільник шини) взагалі виконують ті самі функції, що й відповідно блоки ***Mux*** і ***Demux***, але мають більші можливості щодо перерозподілу сигналів всередині шини.

Блоки ***From*** *(Прийняти від)*, ***Goto Tag Visibility*** *(Ознака видимості)* і ***Goto*** *(Передати до)* використовуються спільно і призначені для обміну даними між різноманітними частинами S-моделі з урахуванням досяжності (видимості) цих даних.

Блоки ***Data Store Read*** *(Читання даних),* ***Data Store Memory*** *(Запам'ятовування даних)* і ***Data Store Write*** *(Запис даних)* також використовуються спільно і забезпечують не тільки передачу даних, але і їхнє збереження на інтервалі моделювання.

Блок ***Merge*** *(Злиття)* виконує об'єднання сигналів, що надходять до його входів, у єдиний.

Група блоків-перемикачів, що керують напрямком передачі сигналу, складається з трьох блоків: ***Switch*** *(Перемикач)****, Manual Switch*** *(Ручний перемикач)* і ***Multiport Switch*** *(Багатовходовий перемикач).*

Блок ***Switch*** має три входи: два (1-й і 3-й) інформаційні й один (2-й) – керувальний – і один вихід. Якщо величина керувального сигналу, що надходить до другого входу, не менше за деяке задане межове значення (параметр *Threshold – Поріг*), то на вихід блока передається сигнал з 1-го входу, у противному разі – сигнал із 3-го входу.

Блок ***Manual Switch*** не має параметрів настроювання. У нього два входи й один вихід. На зображенні блока показано перемичкою, який саме із двох входів залучений до виходу. Блок дозволяє "вручну" перемикати входи. Для цього необхідно двічі клацнути мишкою на зображенні блока. При цьому змі-

ниться і зображення блока – на ньому вихід уже буде сполучений перемичкою з іншим входом.

Блок ***Multiport Switch*** має не менше трьох входів. Перший (горішній) з них є керувальним, інші – інформаційними. Блок має один параметр настроювання *Number of inputs (Кількість входів)*, який визначає кількість інформаційних входів. Номер входу, що з'єднується з виходом, дорівнює значенню керувального сигналу, що надходить на верхній вхід. Якщо це значення є дробовим числом, то воно округлюється до цілого по звичайних правилах. Якщо воно менше за одиницю, то воно вважається рівним 1; якщо воно більше кількості інформаційних входів, то воно приймається рівним найбільшому номеру (входи нумеруються зверху униз, крім самого верхнього – керувального).

5.2.7. Поділ Math Operations (Математичні операції)

У поділі ***Math Operations*** *(Математичні операції)* містяться блоки, які реалізують деякі вбудовані математичні функції системи Matlab. Вони об'єднані у три групи (рис. 5.50)

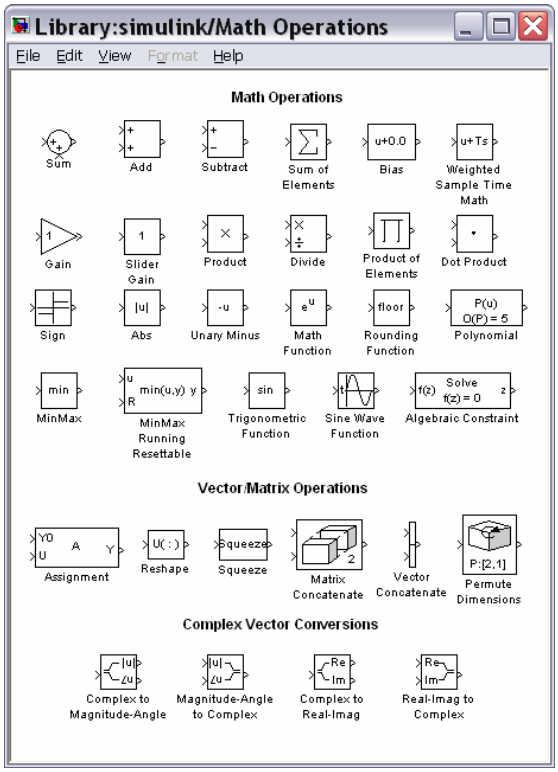


Рис. 5.50. Вміст поділу Math Operations

– ***Math Operations*** *(Математичні операції)*,
– ***Vector/Matrix Operations*** *(Векторно-матричні операції)*
– ***Complex Vector Conversions*** *(Перетворення комплексного вектора)*.

У першу групу ***Math Operations*** входять блоки, які здійснюють математичні перетворення вхідного сигналу у вихідний.

Блоки ***Sum***, ***Add***, ***Subtract*** і ***Sum of Elements*** здійснюють підсумовування сигналів, що надаються до їхніх входів.

Блок ***Sum*** може використовуватися у двох режимах:

- додавання вхідних сигналів (у тому числі з різними знаками);
- підсумовування елементів вектора, що надходить на вхід блока.

Для керування режимами роботи блока використовується два параметри (рис. 5.51):

– *Icon Shape* (Форма зображення),

– *List of signs* (Список знаків).

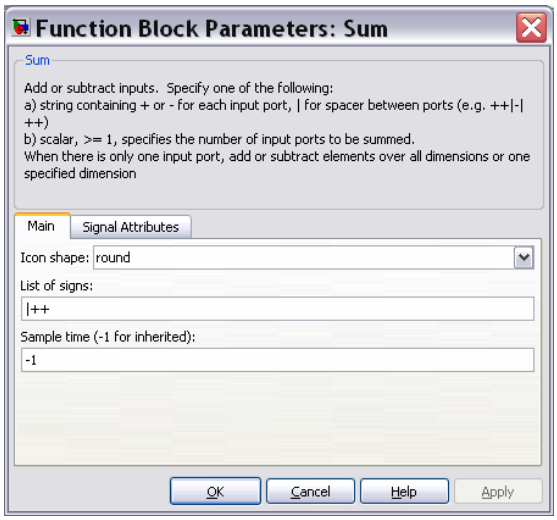


Рис. 5.51. Вікно настроювання блоку Sum

Першій параметр може набувати двох значень: *round* (круглий) і rectangular (прямокутний).

Значення другого параметра можуть задаватися одним із трьох способів:

- у вигляді послідовності знаків "+" або "–"; при цьому *кількість знаків визначає кількість входів блока, а самий знак – полярність відповідного вхідного сигналу*;
- у виді цілої додатної і більше одиниці константи; *значення цієї константи визначає кількість входів блока, а усі входи вважаються додатними*;
- у виді символу "1", який указує, що *блок використовується в другому режимі*

Блок ***Bias*** додає постійну величину (параметр настроювання *Bias*) до вхідного сигналу.

Блок ***Gain*** є лінійною підсилювальною ланкою, тобто здійснює множення вхвдного сигналу на постійне число або вектор, значення якого задається параметром настроювання і вказується на зображенні блоку. Вхідна величина блоку (u) може бути скалярною, векторною або матричною. У випадку, коли вхідний сигнал є вектором довжиною N елементів, коефіцієнт підсилювання має бути вектором тої самої довжини. У списку *Multiplication* (Множення) обирається один з наступних способів множення вхідної величини на вектор K ко-

ефіцієнтів підсилювання: *Element wise(K.*u)* – поелементне множення вхідного вектора на вектор коефіцієнтів підсилювання; *Matrix(K*u)* - матричне множення вектора коефіцієнтів підсилювання на матрицю вхідних величин; *Matrix(u*K)* – матричне множення матриці вхідних величин на вектор коефіцієнтів підсилювання; *Matrix(K*u) (u vector)* – матричне множення векторів K і u.

Блок ***Slider Gain*** є різновидом підсилювальної ланки і одним з елементів взаємодії користувача з моделлю. Він дозволяє в зручній діалоговій формі змінювати значення деякого параметра в процесі моделювання. Щоб відчинити вікно з "повзунковим" регулятором (рис. 5.52), необхідно подвійно клацнути мишкою на зображенні блока.

Рис.5.52. Вікно настроювання блоку Slider Gain

Вікно ***Slider Gain*** має три поля введення інформації:

- для вказівки нижньої межі змінювання параметра (*Low*);
- для вказівки верхньої межі змінювання параметра (*High*);
- для вказівки поточного значення.

Поточне значення має лежати усередині діапазону [*Low, High*]. Його можна вводити, записуючи значення у середнє поле, або переміщуючи мишкою повзунок у верхній частині вікна настроювання. Проте при виборі нового діапазону необхідно спочатку зазначити нове значення параметра, а потім змінити межі діапазону.

Блок ***Product*** виконує множення або ділення кількох вхідних сигналів. У параметри настроювання входять кількість входів блока (*Number of inputs*) й вид виконуваної операції (*Multiplication*). Завдання значень цих параметрів аналогічно настроюванню блока ***Sum.*** Якщо як значення параметра настроювання блока ввести "1", буде обчислюватися добуток елементів вхідного вектора. При цьому на зображенні блока виводиться символ ***P.*** Вхідні сигнали можуть бути векторними або матричними. У списку *Multiplication* обирається спосіб множення вхідних величин: *Element wise* – поелементне множення вхідних векторів або матриць; *Matrix* – матричне множення вхідних векторів або матриць. У випадку, коли результат виконання має містити ділення на деякі вхідні величини, у полі *Number of inputs* (Кількість входів) слід ввести послідовність символів "*" або "/" (за кількістю входів блоку). Якщо обрано матричне множення, то символ "/" означає множення на матрицю, обернену по відношенню до матриці відповідної вхідної величини.

У блоці ***Dot Product*** (Скалярний добуток) є лише два входи. Вхідні сигнали блоку мають бути векторами однакової довжини. Вихідна величина блоку у

кожний момент часу дорівнює сумі добутків відповідних елементів цих двох векторів. Якщо вектори є комплексними, то перед множенням першій вектор (верхній вхідний порт) замінюється комплексно-спряженим.

Блок ***Sign*** реалізує нелінійність типу сигнум-функції. У ньому немає параметрів настроювання. Блок формує вихідний сигнал, що набуває лише трьох можливих значень: "+1" – у випадку, коли вхідний сигнал є додатним, "-1" – при від'ємному вхідному сигналі і "0" при вхідному сигналі, рівному нулю.

Блок ***Abs*** формує абсолютне значення вхідного сигналу. Він не має параметрів настроювання.

Блок ***Trigonometric Function*** забезпечує перетворення вхідного сигналу за допомогою однієї з таких функцій MatLAB: ***sin, cos, tan, asin, acos, atan, atan2, sinh, cosh, tanh.*** Обрання необхідної функції здійснюється у вікні настроювання блоку за допомогою спадного меню.

Блок ***Math Function*** дозволяє обрати для перетворення вхідного сигналу елементарні не тригонометричні і не гіперболічні функції, такі як ***exp***, ***log***, ***10^u***, ***log10***, ***square*** (u^2), ***sqrt***, ***pow***, ***reciprocal*** (1/u), ***hypot***, ***rem***, ***mod***. Потрібна функція обирається за допомогою спадного меню у вікні настроювання.

Блок ***Rounding Function*** містить різноманітні функції округлення, передбачені в MatLAB. Він здійснює округлення значення вхідного сигналу. Вибір конкретного методу округлення здійснюється також за допомогою спадного меню у вікні настроювання.

Для зазначених вище блоків ім'я обраної функції виводиться на графічному зображенні блока.

Блок ***MinMax*** здійснює пошук мінімального або максимального елемента вхідного вектора. Якщо входом є скалярна величина, то вихідна величина збігається із вхідною. Якщо входів декілька, шукається мінімум або максимум серед входів. У число настроювань входить вибір методу й кількість входів блока.

Блоки третьої групи – *Complex Vector Conversions* (Перетворення комплексного вектора) здійснюють перетворення комплексних вхідних сигналів у дійсні і навпаки. Блоки ***Complex to Magnitude-Angle*** та ***Complex to Real-Imag*** здіюснюють перетворення комплексного вхідного сигнала у два дійсних сигнали, що є модулем і аргументом вхідного сигналу у першому випадку і дійсною і уявною частинами – у другому випадку. Блоки ***Magnitude-Angle to Complex*** та ***Real-Imag to Complex*** перетворюють два вхідних дійсних сигнали у єдиний комплексний.

5.2.8. Поділ Logic & Bit Operations (Логічні та бітові операції)

Як вказано у назві поділу, він містить блоки, що забезпечують певну логічну обробку вхідних величин, або перетворення їх двійкового подання (рис. 5.53).

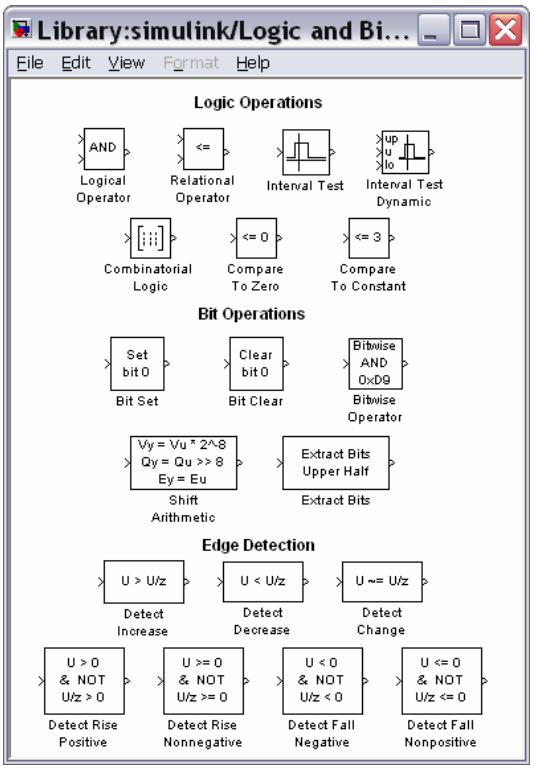


Рис. 5.53. Вміст поділу Logic & Bit Operations

Загальним для всіх блоків групи Logic Operations є те, що вихідна величина в усіх них є бульовою, тобто може приймати лише два значення: "1" ("істина") або "0" ("хибність"). У багатьох з них бульовими мають бути й усі вхідні величини.

Блок ***Relational Operator*** реалізує операції відношення між двома вхідними сигналами >, <, ≤, ≥, ==, ~= (відповідно: більше, менше, менше або дорівнює, більше або дорівнює, тотожно, не дорівнює). Конкретна операція обирається при настроюванні параметрів блока за допомогою спадного меню. Знак операції виводиться на зображенні блока.

Блок ***Logical Operator*** містить набір основних логічних операцій AND, OR, NAND, NOR, XOR, NOT. Вхідні величини мають бути бульовими. обрання необхідної логічної операції здійснюється у вікні настроювання блока за допомогою спадного меню. Другим параметром настроювання є кількість вхідних величин (портів) блока (*Number of input ports*), тобто кількість аргументів логічної операції.

Блок ***Combinatorial Logic*** забезпечує перетворення вхідних бульових величин у вихідну у відповідності із заданою таблицею істинності. Блок має єдиний параметр настроювання – *Truth table* (таблиця істинності).

5.2.9. Поділ User-defined Functions (функції, що визначаються користувачем)

У поділі ***User-defined Functions*** (Користувацькі функції) містяться блоки, призначення яких визначає користувач (рис. 5.54).

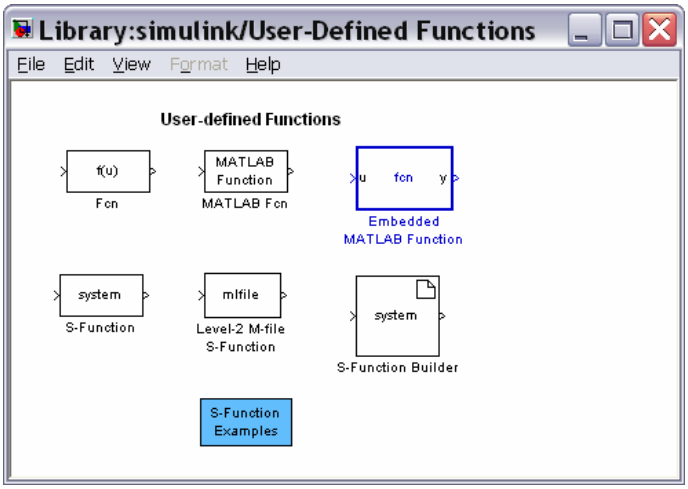


Рис. 5.54. Вміст поділу User-defined Functions

Блок ***Fcn*** дозволяє користувачеві ввести будь-яку скалярну функцію від одного (скалярного або векторного) аргументу, яка виражається через стандартні функції MatLAB. Вираз функції вводиться у вікні настроювання блока згідно правил М-мови. Для позначення вхідного сигналу (аргументу функції) використовується символ ***u***.

Блок ***MATLAB Fcn*** дозволяє застосувати до вхідного сигналу будь-яку підпрограму обробки, реалізовану у виді М-файлу. На відміну від попереднього блока, тут до числа параметрів настроювання доданий параметр *Output width (Ширина вихідного сигналу)*, який визначає кількість елементів вихідного вектора. Окремий *i*-й елемент вихідного вектора у вікні настроювання ***MATLAB Fcn*** задається у виді функції, що записана на М-мові, якій передує запис ***u(i)=.***

За допомогою блоку ***S-function*** користувач має змогу реалізувати у виді візуального блоку Simulink будь-яку програму обробки вхідного сигналу, включаючи створення складних моделей систем, що описані нелінійними диференціальними або кінцево-різницєвими рівняннями, і обробку дискретних у часі сигналів. Більш детально робота з S-функціями описана далі, у главі 4.

Блок ***S-function Builder*** (Побудувач S-функцій) дає можливість користувачеві утворювати S-функцію у діалоговому режимі.

5.2.10. Поділ Ports & Subsystems (Порти та підсистеми)

Більшість блоків поділу ***Ports & Subsystems*** призначені для розробки складних за ієрархією S-моделей, що містять моделі більш низького рівня (підсистеми), а також забезпечують встановлення необхідних зв'язків між кількома S- моделями (рис. 5.55).

Підсистема являє собою S-модель більш низького рівня, у яку можуть бути вкладені підсистеми різних рівнів. підсистеми можуть функціонувати лише у складі основної S-моделі, зв'язок з якою здійснюється через вхідні (***In***) і вихідні (***Out***) порти підсистеми.

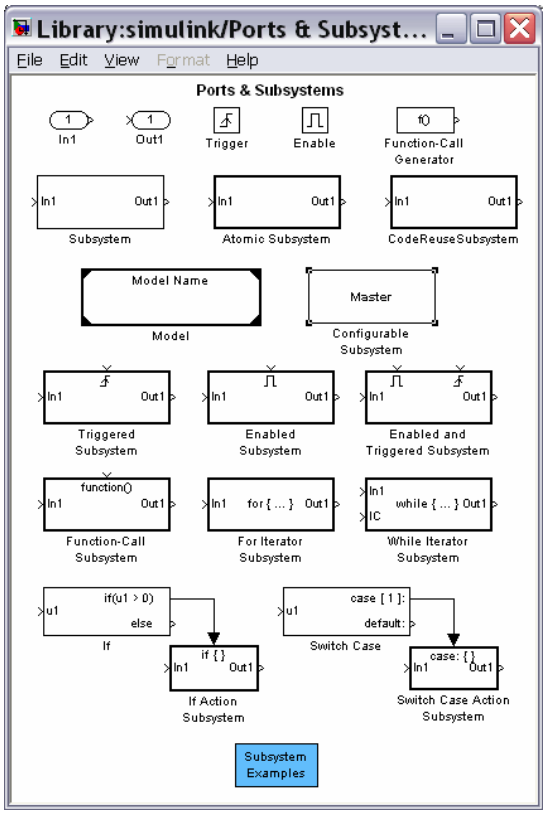


Рис. 5.55. Вміст поділу Ports & Subsystems

Роль підсистеми у S-моделі така сама, що й роль функцій (процедур) в основній М-програмі, яка їх викликає. При цьому вхідні і вихідні величини підсистеми визначаються відповідно її вхідними і вихідними портами. Результати виконання дій у підсистемі (вихідні величини) у подальшому можуть бути використані у S-моделі, яка її викликає (або у підсистемі більш високого рівня).

Застосування підсистем дозволяє звести складання складної S-моделі до утворення сукупності вкладених простих підсистем більш низького рівня, що робить моделювання більш наочним і спрощує налагодження моделі.

Опишемо основні блоки поділу.

Блоки ***In*** (Вхідний порт) і ***Out*** (Вихідний порт) забезпечують інформаційний зв'язок між підситемою і S-моделлю, що її викликає.

Блоки ***Enable*** *(Дозволити)* і ***Trigger*** *(Засувка)* призначені для логічного керування роботою підсистем S-моделі.

Раніше розглянути блоки ***Ground*** (Земля) і ***Terminator*** (Обмежувач) можуть використовуватися як "заглушки" для тих портів, які з якоїсь причини не були з'єднані з іншими блоками S-моделі. при цьому блок ***Ground*** застосовується для вхідних портів, а блок ***Terminator*** – для вихідних.

Блок ***Subsystem*** *(Підсистема)* є "заготівкою" для створення підсистеми. Подвійне клацання на його зображенні приводить до появі на екрані вікна, в якому розміщена блок-схема, що складається лише з одного вхідного порта, який з'єднаний з одним вихідним портом (рис. 5.56). Це є нагадуванням, що утворювана користувачем підситема має обов'язково містити з'єднані між собою (можливо, через інші блоки) вхідні і вихідні порти.

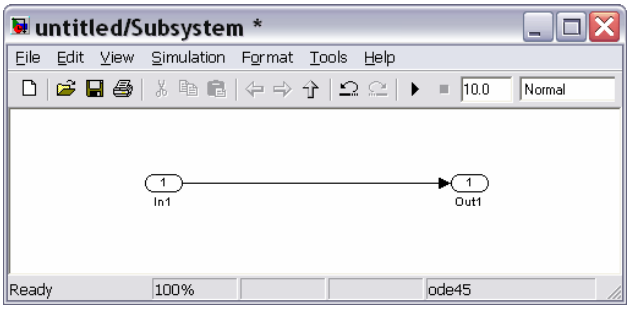


Рис. 5.56. Вікно заготівки блоку Subsystem

У вікні, що відчинилося, користувач будує блок-схему підсистеми за звичайними правилами побудови блок-схем, а потім записує її на диск. Розміщення додаткових вхідних і вихідних портів приведе до появи на зображенні блоку ***Subsystem*** додаткових входів і виходів. При цьому поряд з відповідними входами і виходами блока ***Subsystem*** виникнуть написи, що зроблені на вхідних і вихідних портах підсистеми.

5.3. Побудова блок-схем

Розглянемо операції, за допомогою яких можна формувати блок-схеми складних динамічних систем.

5.3.1. Виділення об'єктів

При створенні й редагуванні S-моделі потрібно виконувати такі операції, як копіювання або вилучення блоків і ліній, для чого необхідно спочатку виділити один чи кілька блоків і ліній (об'єктів).

Щоб ***виділити окремий об'єкт,*** потрібно клацнути на ньому один раз. При цьому з'являються маленькі кола по рогах блока, або на початку й кінці лінії. *При цьому стають невиділеними усі інші попередньо виділені об'єкти*. *Якщо клацнути по блоку другий раз, він стає невиділеним*.

На рис. 5.57 наведений результат виділення лінії, що з'єднує блоки Constant і XYGraph, а на рис. 5.58 – блоку ***Clock***.

Щоб виділити кілька об'єктів, слід, утримуючи натисненою клавішу Shift, клацнути на кожному з них, а потім відпустити клавішу .

Саме у такий спосіб на рис. 5.59 виділені блоки ***Signal Generator***, ***Constant*** і ***XYGraph***.

Кілька об'єктів можна виділити також за допомогою прямокутної рамки. Для цього потрібно клацнути мишкою у точці, яка буде прислуговуватися рогом рамки, а потім, утримуючи кнопку миші натисненою, протягнути курсор у напрямку діагоналі прямокутника. В результаті навколо об'єктів, що виділяються має виникнути пунктирна рамка (рис. 5.60). Коли усі потрібні об'єкти будуть охоплені рамкою, потрібно відпустити кнопку миші.

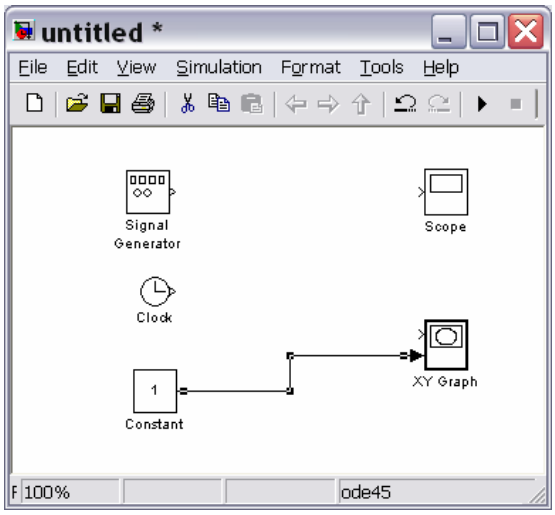


Рис. 5.57. Виділена лінія

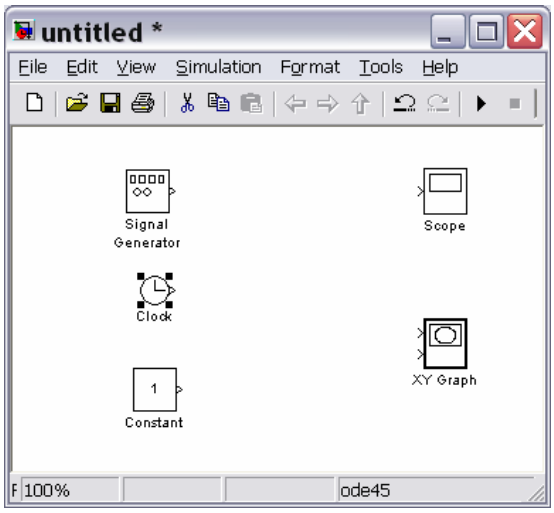


Рис. 5.58. Виділений блок Clock

Виділення усієї моделі, тобто усіх об'єктів в активному вікні блок-схеми, здійснюється одним із двох шляхів:

1) обранням команди *Select All* у меню *Edit* вікна блок-схеми;
2) натисканням сукупності клавіш <Ctrl>+<A>.

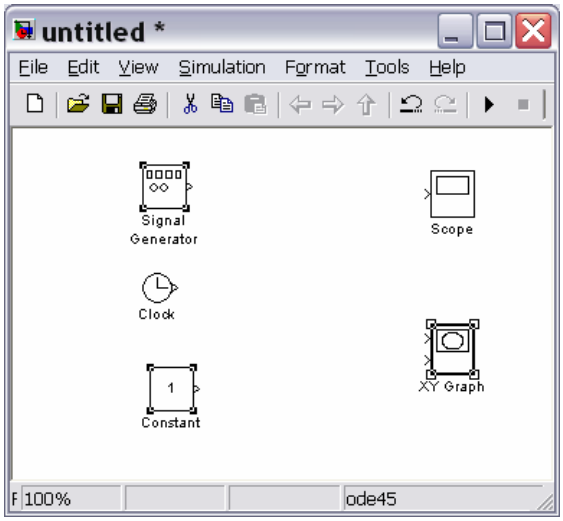


Рис. 5.59. Виділення кількох блоків

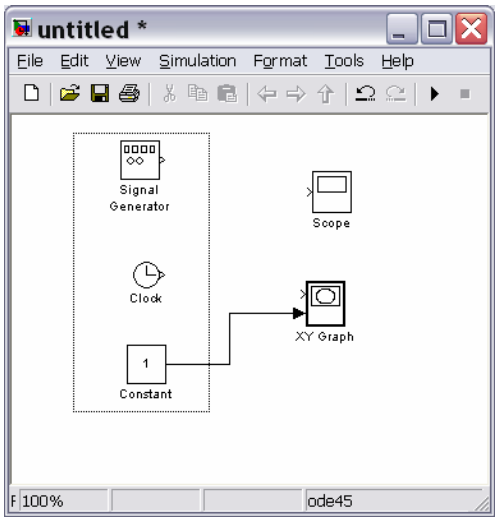


Рис. 5.60. Виділення за допомогою рамки

5.3.2. Операції з блоками

Копіювання блоків з одного вікна у інше

Під час створення й редагування моделі потрібно копіювати блоки з бібліотеки або іншої моделі у поточну модель. Для цього достатньо:

- відчинити потрібний розділ бібліотеки чи вікно моделі – прототипу;
- перетягнути мишкою потрібний блок у вікно утворюваної (редагованої) моделі.

Інший спосіб є таким:

1) виділити блок, який потрібно скопіювати;
2) обрати команду *Copy (Копіювати)* з меню *Edit (Редагування)*;

3) зробити активним вікно, в яке потрібно скопіювати блок;

4) обрати в ньому команду *Paste (Встановити)* з меню *Edit*.

Кожному зі скопійованих блоків автоматично присвоюється ім'я. *Першій скопійований блок матиме те саме ім'я, що й блок у бібліотеці*. *Кожний наступний блок того ж типу матиме те саме ім'я з додаванням порядкового номера*. Користувач може перейменувати блок (див. далі).

При копіюванні блок набуває тих самих значень настроюваних параметрів, що й блок – оригінал.

Переміщення блоків у моделі

Щоб перемістити блок усередині моделі з одного місця у інше, достатньо перетягнути його у це положення за допомогою мишки. При цьому будуть автоматично перерисовані лінії зв'язків інших блоків з тим, якого переставлено.

Переставити кілька блоків одночасно, включаючи з'єднувальні лінії можна у такий спосіб:

- виділити блоки й лінії (див попередній розділ);
- перетягнути мишкою один із виділених блоків на нове місце; решта блоків, зберігаючи всі відносні відстані, посядуть нові місця.

На рис. 5.61 показаний результат таких дій з блоками, виділеними на рис. 5.59.

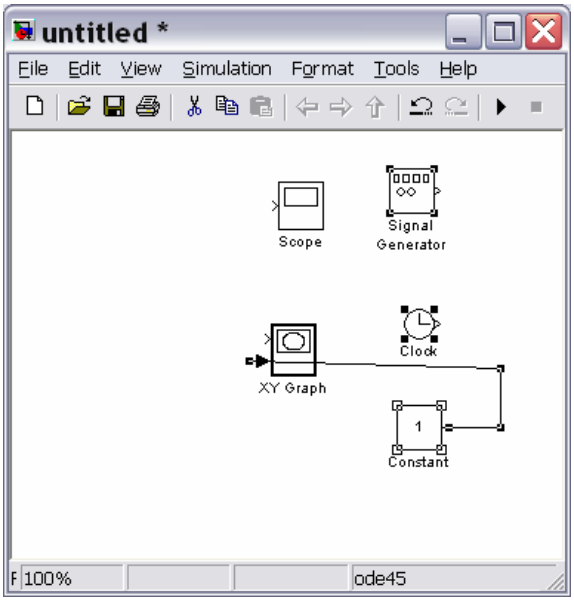


Рис. 5.61. Результат переміщення блоків, виділених на рис. 3.60

Дублювання блоків усередині моделі

Щоб ***скопіювати блоки усередині моделі*** потрібно зробити наступне:

1) натиснути клавішу <Ctrl>;

2) не відпускаючи клавішу <Ctrl>, встановити курсор на блок, який необхідно скопіювати й перетягнути його у нове положення.

Того самого результату можна досягти, якщо просто *перетягти* мишкою блок у нове положення, але за допомогою *правої* клавіші мишки.

На рис. 5.62 подано результат копіювання блоків ***Scope*** і ***XYGraph***.

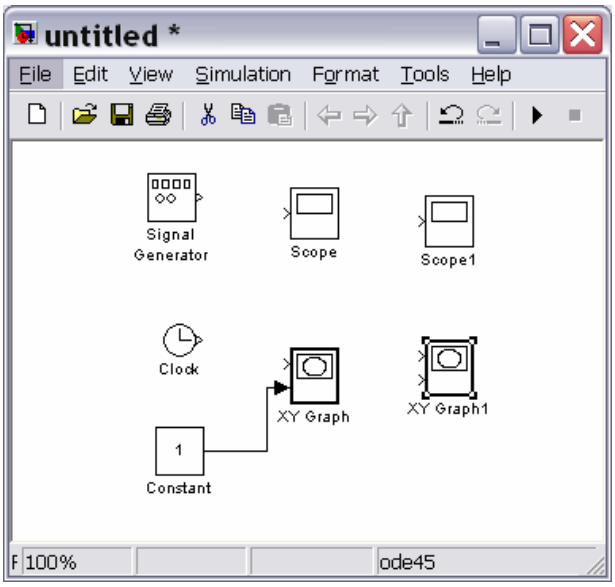


Рис. 5.62. Результат копіювання блоків

Встановлення параметрів блока

Функції, які виконує блок, залежать від значень параметрів блока. Встановлення цих значень здійснюється у вікні настроювання блока, яке виникає, якщо подвійно клацнути на зображенні блока у блок-схемі.

Вилучення блоків

Для ***вилучення непотрібних блоків*** із блок-схеми достатньо виділити ці блоки так, як було зазначено раніше, і натиснути клавішу <Delete> або <Backspace>. Можна також використати команду *Clear* або *Cut* із розділу *Edit* меню вікна блок-схеми. Якщо використано команду *Cut*, то у подальшому вилучені блоки можна скопіювати зворотно у модель, якщо скористатися командою *Paste* того ж розділу меню вікна схеми.

Від'єднання блока

Для ***від'єднання блоку від з'єднувальних ліній*** достатньо натиснути клавішу <Shift> і, не відпускаючи її, перетягнути блок у деяке інше місце.

Змінювання кутової орієнтації блока

У звичайному зображенні сигнал проходить крізь блок зліва направо (ліворуч містяться входи блока, а праворуч – виходи). Щоб ***змінити кутову орієнтацію блока*** потрібно:

- виділити блок, який потрібно повернути;
- обрати меню *Format* у вікні блок-схеми;
- у додатковому меню, яке з'явиться на екрані обрати команду *Flip Block* – поворот блока на 180 градусів, або *Rotate Block* – поворот блока за годинниковою стрілкою на 90 градусів.

На рис. 5.63 показаний результат застосування команди *Rotate Block* до блоку ***Constant*** і команд *Rotate Block* і *Flip Block* - до блока ***Signal Generator***.

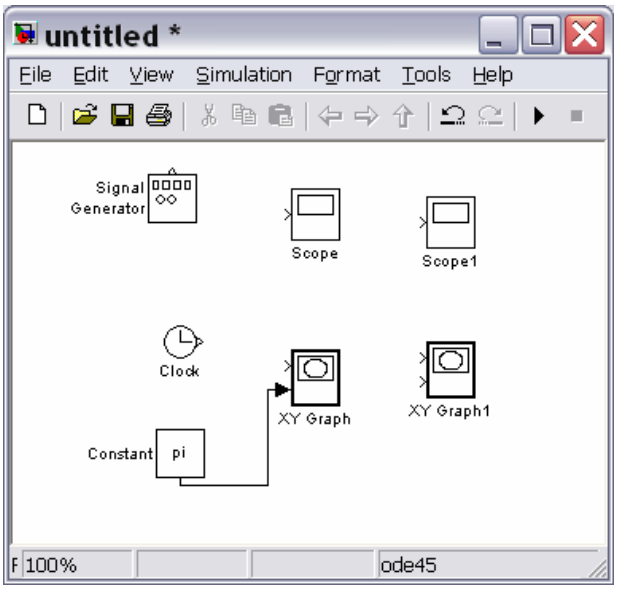


Рис. 3.63. Результат зміненяя орієнтації блоків

Змінювання розмірів блока

Щоб змінити розміри блока, потрібно зробити наступне:

- виділити блок, розміри якого треба змінити;
- навести курсор мишки на одну з рогових міток блоку; при цьому на екрані у цієї мітки має виникнути новий курсор у вигляді обопільної стрілки під нахилом 45 градусів;
- захопити цю мітку мишкою і перетягнути у нове положення; при цьому протилежна мітка цього блоку залишиться нерухомою.

На рис. 5.64 показаний результат процесу збільшення розмірів блока ***XYGraph1***.

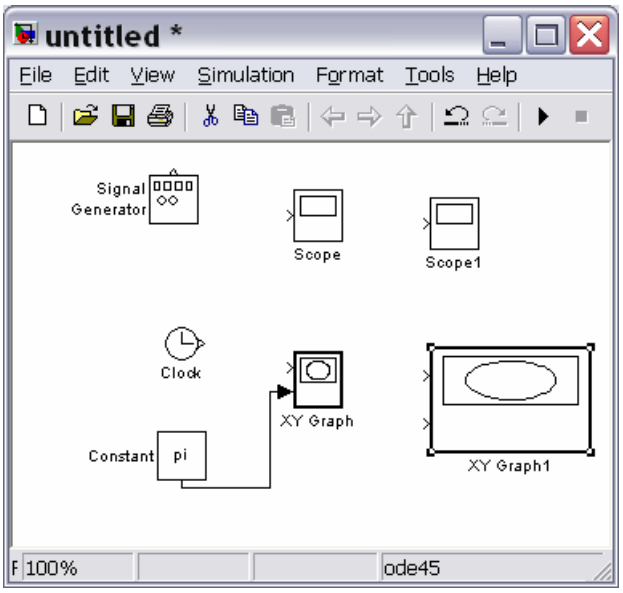


Рис. 5.64. Результат розтягування блоку XYGraph1

Змінювання імен блоків та маніпулювання з ними

Усі імена блоків у моделі мають бути унікальними і мати, як мінімум один символ. Якщо блок орієнтований зліва направо, то ім'я, за замовчуванням, міститься під блоком, якщо справа наліво – понад блоком, якщо ж зверху униз або знизу уверх – праворуч блоку (див. рис. 5.64).

Змінювання ймення блоку здійснюється у такий спосіб: треба клацнути на існуючому імені блока, потім, використовуючи клавіші звичайного редагування тексту, змінити це ім'я на потрібне.

Для ***змінювання шрифту*** слід виділити блок, потім обрати команду *Font* (Шрифт) із меню *Format* вікна моделі і обрати потрібний шрифт із поданого переліку.

Щоб ***змінити місце розташування ймення виділеного блока,*** існують два шляхи:

- перетягнути ім'я на протилежний бік мишкою;
- скористатися командою *Flip Name* (Розвернути ім'я) із поділу *Format* меню вікна моделі – вона теж переносить ім'я на протилежний бік.

Вилучити ім'я блоку можна,використовуючи команду *Hide Name* (Сховати ім'я) з меню *Format* вікна моделі. Щоб ***відновити*** потім ***відображення імені*** поряд із зображенням блоку, слід скористатися командою *Show Name* (Показати ім'я) того самого меню.

5.3.3. Проведення з'єднувальних ліній

Сигнали у моделі передаються по лініях. Кожна лінія може передавати або скалярний, або векторний сигнал. Лінія з'єднує вихідний порт одного блоку із вхідним портом іншого блоку. Лінія може також з'єднувати вихідний порт одного блоку із вхідними портами кількох блоків через розгалужування лінії.

Створення лінії між блоками

Для сполучення вихідного порту одного блока із вхідним портом іншого блоку слід виконати таку послідовність дій:

- встановити курсор на вихідний порт першого блока; при цьому *курсор має перетворитися на перехрестя*;
- утримуючи натисненою ліву кнопку миші, пересунути перехрестя до вхідного порту другого блоку;
- відпустити кнопку миші; Simulink замінить символи портів з'єднувальною лінією з поданням напрямку передавання сигналу.

Саме у такий спосіб з'єднано на рис. 5.60 вихід блока ***Constant*** із входом блоку ***XYGraph***.

Лінії можна рисувати як от вхідного порту до вихідного, так і навпаки.

Simulink малює з‘єднувальні лінії, використовуючи лише горизонтальні й вертикальні сегменти *Для утворення діагональної лінії натисніть і утримуйте клавішу <Shift> протягом рисування.*

Створення розгалуження лінії

Лінія, що розгалужується, починається з існуючої і передає її сигнал до вхідного порту іншого блоку. Як існуюча, так і відгалужена лінія передають той самий сигнал. Розгалужена лінія дає можливість передати той самий сигнал до кількох блоків.

Щоб ***утворити відгалуження*** від існуючої лінії, потрібно:

- установити курсор на точку лінії, від якої має відгалужуватися інша лінія;
- натиснувши й утримуючи клавішу <Ctrl>, натиснути й утримувати ліву кнопку миші;
- провести лінію до вхідного порту потрібного блоку; відпустити клавішу <Ctrl> і ЛКМ (див. рис. 5.65).

Те саме можна зробити, використовуючи праву кнопку миші, при цьому не потрібно користатися клавишою <Ctrl>.

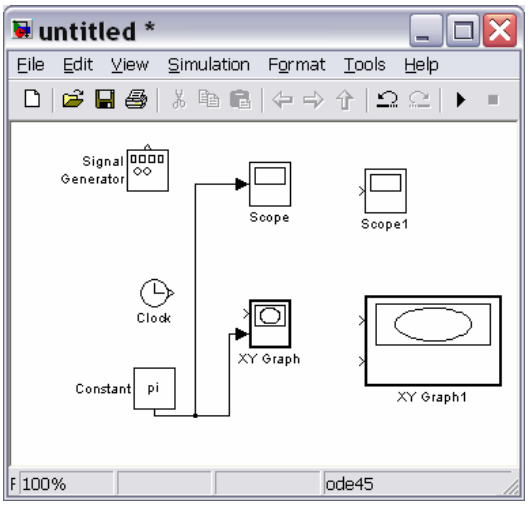


Рис. 5.65. Утворення розгалуження лінії

Створення сегмента лінії

Лінії можуть бути нарисовані по сегментах. У цьому разі для створення наступного сегмента слід установити курсор у кінець попереднього сегмента і нарисувати (за допомогою миші) наступний сегмент. У такий спосіб, наприклад, з'єднані на рис. 5.66 блоки ***Clock*** з ***XYGraph1*** та ***Signal Generatorr*** з ***Scope1***.

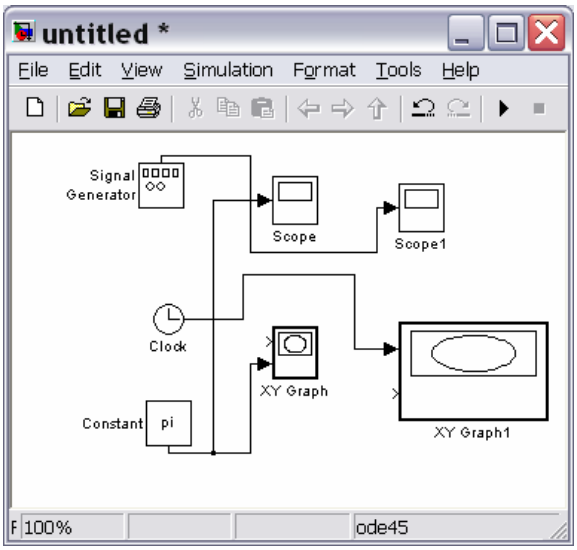


Рис. 5.66. Лінії, нарисовані по сегментах

Пересування сегмента лінії

Щоб ***пересунути окремий сегмент*** лінії, необхідно виконати наступне:

- установити курсор на сегмент, який потрібно пересунути;
- натиснути й утримувати ліву кнопку миші; при цьому курсор має перетворитися на "хрест";
- пересунути "хрест" до нового положення сегмента;
- відпустити кнопку миші.

На рис. 5.67 показаний результат пересування вертикального сегменту лінії, що з'єднує блоки ***Random Number*** з ***XYGraph1***.

Не можна пересунути сегмент, який безпосередньо прилягає до порту блока.

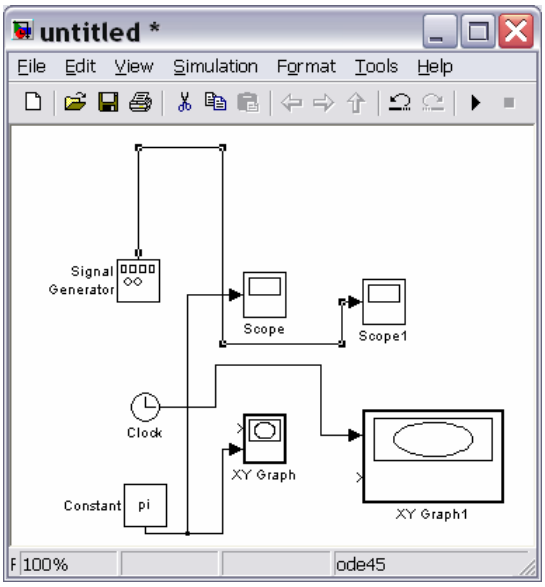


Рис. 5.67. Сегмент лінії переміщений угору

Поділення лінії на сегменти

Щоб ***поділити лінію на два сегменти,*** потрібно:

- виділити лінію;
- установити курсор у ту точку виділеної лінії, у якій лінія має бути поділеною на два сегменти;
- утримуючи натисненою клавішу <Shift>, натиснути й утримувати ліву кнопку миші; курсор перетвориться на маленьке коло; на лінії утвориться злам;
- пересунути курсор у нове положення зламу;
- відпустити кнопку миші і клавішу <Shift>.

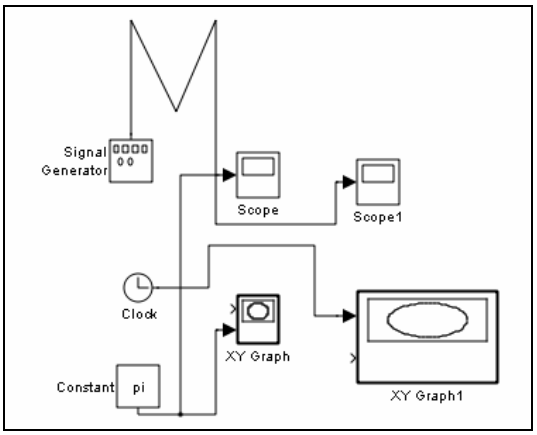


Рис. 5.68. Верхня горизонтальна лінія поділена на два сегменти

Приклад проведення цих дій подано на рис. 5.68, де лінія, яка з'єднує блоки ***Sine Wave*** і ***XYGraph1***, поділена на два сегменти.

Пересування зламу лінії

Для ***пересування зламу*** лінії достатньо мишкою перетягнути точку цього зламу у нове положення на блок-схемі.

5.3.4. Мітки сигналів і коментарів

Задля наочності оформлення блок-схеми й зручності користування нею можна супроводжувати лінії мітками сигналів, що прямують по ній. Мітка розміщується під або над горизонтальною лінією, ліворуч або праворуч вертикальної лінії. Мітка може бути розташована на початку, у кінці або посередині лінії.

Створення й маніпулювання мітками сигналів

Щоб ***створити мітку сигналу,*** треба подвійно клацнути на сегменті лінії і ввести мітку (рис. 5.69). *При створенні мітки сигналу необхідно подвійно клацнути саме точно на лінії, бо інакше буде створений коментар до моделі.*

Для ***пересування мітки*** треба її просто перетягнути на нове місце мишкою.

Щоб ***скопіювати мітку,*** слід натиснути й утримувати клавішу <Ctrl> і перетягнути мітку до нового положення на лінії, або обрати інший сегмент лі-

нії, на якому потрібно встановити копію мітки і подвійно клацнути по цьому сегменту лінії.

Відредагувати мітку можна клацнувши на ній і здійснюючи потім відповідні змінювання як у звичайному текстовому редакторі.

Щоб ***вилучити мітку***, натисніть і утримуйте клавішу <Shift>, виділить мітку і знищить її, використовуючи клавіші <Delete> або <Backspace>. При цьому будуть вилучені усі мітки цієї лінії.

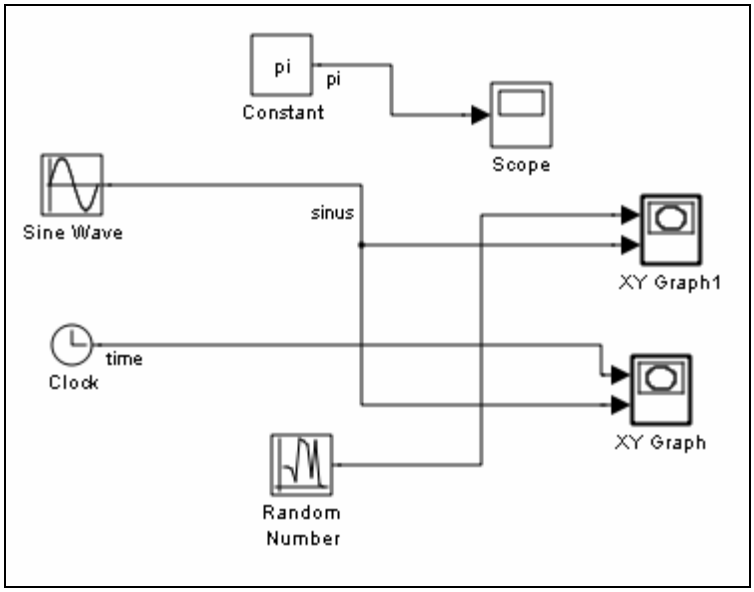


Рис. 5.69. Створені мітки pi, sinus і time

Розповсюдження міток лінії

Розповсюдження міток лінії – це процес автоматичного переносу імені мітки до сегментів однієї лінії, що розірвані блоками ***From/Goto***, ***Mux***.

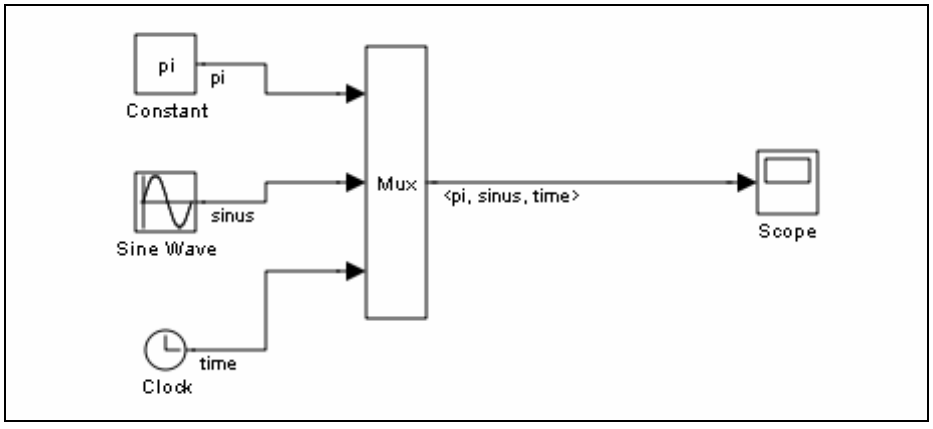


Рис.5.70. Розповсюдження міток

Щоб ***розповсюдити мітки,*** створіть у другому й наступних сегментах лінії мітки з ім'ям '<'. Після повернення до вікна блок-схеми у цих сегментах автоматично будуть проставлені мітки (див. рис. 5.70)

Створення коментаря й маніпулювання ним

Коментарі дають можливість опоряджувати блок-схеми текстовою інформацією про модель та окремі її складові. Коментарі можна проставляти у будь-якому вільному місці блок-схеми (див. рис. 5.71).

Для ***створення коментаря*** двічі клацніть у будь-якому вільному місці блок-схеми, а потім уведіть коментар у прямокутнику, що виник.

Для ***переміщення коментарю*** у інше місце його потрібно перетягнути на це місце за допомогою миші.

Щоб ***скопіювати коментар,*** достатньо натиснути клавішу <Ctrl> і, утримуючи її у цьому положенні, перетягти коментар у нове місце.

Для ***редагування коментарю*** треба клацнути на ньому, а потім внести потрібні зміни як у звичайному текстовому редакторі. Щоб ***змінити*** при цьому ***шрифт***, його розмір або стиль, слід виділити текст коментарю, який потрібно змінити, а потім обрати команду *Font* із меню *Format* вікна блок-схеми, вибрати у вікні, що виникне, назву шрифту, його розмір, атрибути й стиль і натиснути кнопку <Ok> у цьому вікні.

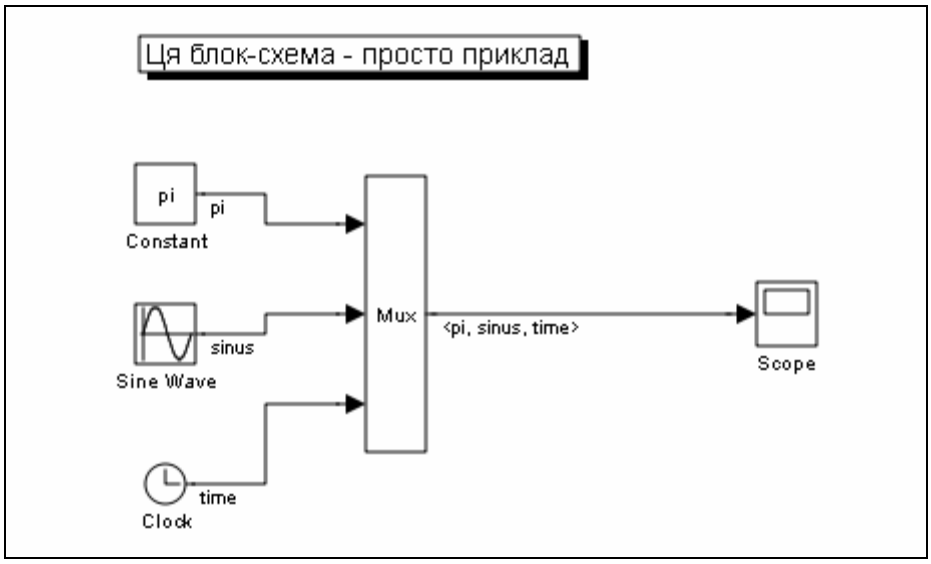


Рис. 5.71. Приклад розміщення коментаря на блок-схемі

Щоб ***вилучити коментар,*** натисніть і утримуйте клавішу <Shift>, виділить коментар і натисніть клавішу <Delete> або <Backspace>.

5.3.5. Створення підсистем

Якщо складність і розміри блок-схеми моделі стають надто великими, її можна суттєво спростити, групуючи блоки у підсистеми. Використання підсистем дає наступні переваги:

- скорочення кількості блоків, що виводяться у вікні моделі;
- об'єднання у єдину групу (підсистему) функціонально пов'язаних блоків;

- можливість створення ієрархічних блок-схем.

Існують три можливості створення підсистем:

- додати блок ***Subsystem*** у модель, потім увійти у цей блок і створити підсистему у вікні підсистеми, яке при цьому виникне;
- виділити частину блок-схеми моделі й об'єднати її у підсистему.

Створення підсистеми через додавання блока ***Subsystem***

У цьому разі слід зробити наступне:

- скопіювати блок ***Subsystem*** у вікно моделі, перетягнувши його з поділу ***Ports&Subsystems***;
- розкрити вікно блока ***Subsystem***, подвійно клацнувши на зображенні блока у блок-схемі;
- у порожньому вікні блокн ***Subsystem*** створити підсистему, використовуючи блоки ***In*** і ***Out*** для створення входів і виходів підсистеми.

Створення підсистеми через групування існуючих блоків

Якщо блок-схема вже містить блоки, які потрібно об'єднати у підсистему, то останню можна зробити у такий спосіб:

- виділити рамкою блоки і з'єднувальні лінії, які потрібно включити у склад підсистеми (див. рис. 5.72);
- обрати команду *Create Subsystem* із меню *Edit* вікна моделі; при цьому SimuLINK замінить виділені блоки одним блоком ***Subsystem*** (див. рис. 3.73)

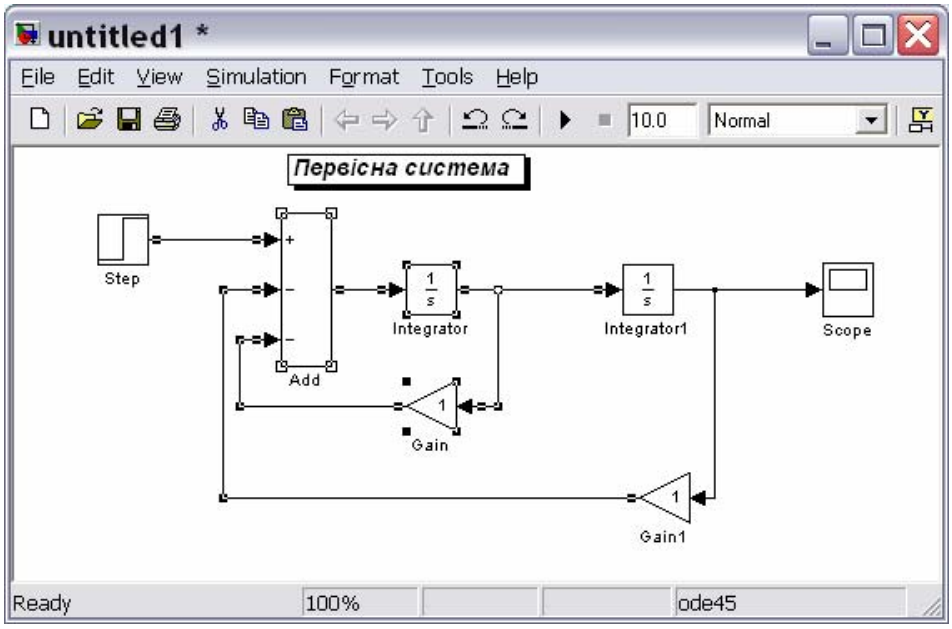


Рис. 5.72. Первісна система з виділеними блоками

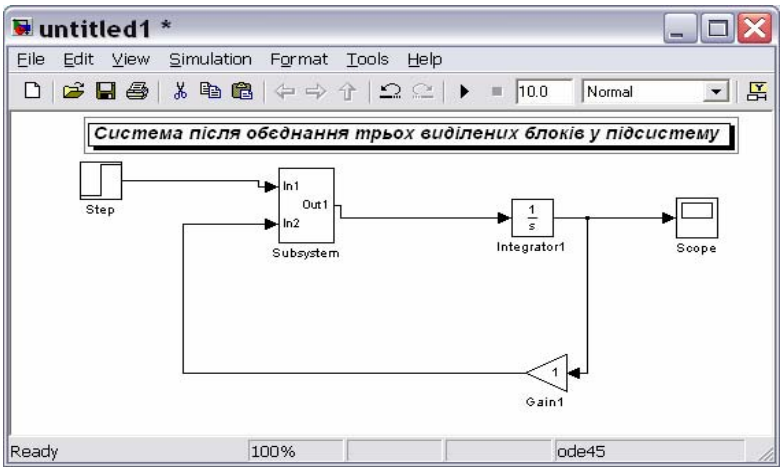


Рис. 5.73. Система після об'єднання виділених блоків у підсистем

Якщо розкрити вікно блоку ***Subsystem***, подвійно клацнувши на ньому, то Simulink відобразить блок-схему створеної підсистеми (рис. 5.74). Як видно, Simulink додав блоки ***In*** і ***Out*** для відображування входів і виходів у систему вищого рівня.

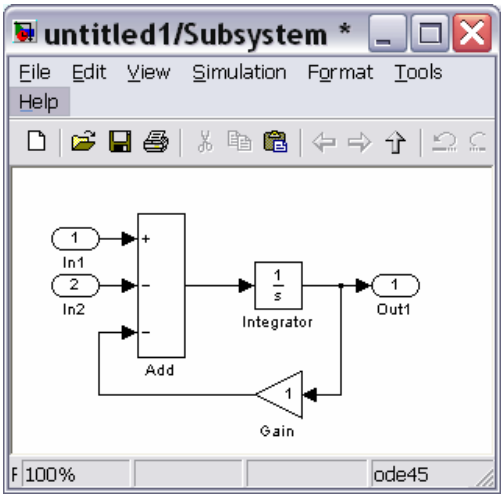


Рис. 5.74. Утворена підсистема

5.3.6. Зберігання і виведення до друку блок-схеми

Для запису моделі (блок-схеми) на диск потрібно викликати команду *Save* (Зберегти) або *Save As* (Зберегти як) з меню *File* (Файл) вікна моделі. Simulink записує у вказану директорію файл з заданим (введеним з клавіатури) ім'ям, присвоюючи йому розширення ***.mdl***. При цьому у цей файл автоматично записуються усі вложені підсистеми моделі.

Щоб роздрукувати блок-схему на прінтері, потрібно скористуватися командою *File* > *Print* (Файл > Друк) вікна моделі.

Доволі цікавою і важливою є можливість вставлення блок-схеми у документ Word. Для цього слід використати команду *Copy model to clipboard* (Копіювати модель у буфер) меню *Edit* (Правка) вікна моделі, яка поміщає у буфер обміну вміст вікна моделі. Якщо після цього перейти у вікно текстового редак-

тора і натиснути клавиші Ctrl+V, у відкритому документі редактора виникне зображення блок-схеми моделі. Саме у такий спосіб отримані рис. 5.68-5.71.

ЛІТЕРАТУРА

1. Лазарєв Ю. Ф. Початки програмування у середовищі MatLAB: Навч. посібник. – К.: "Корнійчук", 1999. – 160 с.
2. Лазарев Ю. Ф. MatLAB 5.x. – К.: Издат. группа BHV, 2000. – 384 с.
3. Лазарев Ю. Ф. Моделирование процессов и систем в MATLAB. Учебный курс. – СПб.: Питер; Киев: Издат. группа BHV, 2005. – 512 с.
4. Лазарєв Ю. Ф. Моделювання на ЕОМ. Навч. посібник. – К.: Корнійчук, 2007.-290 с.
5. Мартынов Г. Г., Иванов А. П. MATLAB 5.x, вычисления, визуализация, программирование. - М.: "Кудиц-образ", 2000. - 332 с.
6. Гультяев А. Визуальное моделирование в среде MATLAB: учебный курс. - СПб. : "Питер", 2000. - 430 с.
7. Краснопрошина А. А., Репникова Н. Б., Ильченко А. А. Современный анализ систем управления с применением MATLAB, Simulink, Control System: Учебное пособие. - К.: "Корнійчук", 1999. - 144 с.
8. Мартынов Г. Г., Иванов А. П. MATLAB 5.x, вычисления, визуализация, программирование. - М.: "Кудиц-образ", 2000. - 332 с.
9. Медведев В. С., Потемкин В. Г. Control System Toolbox. MatLAB 5 для студентов. - М.: ДИАЛОГ-МИФИ, 1999. - 287 с.
10. Потемкин В. Г. Система MatLAB: Справ. пособие. - М.: ДИАЛОГ-МИФИ, 1997. - 350 с.
11. Потемкин В. Г. MatLAB 5 для студентов: Справ. пособие. - М.: ДИАЛОГ-МИФИ, 1998. - 314 с.
12. Потемкин В. Г., Рудаков П. И. MatLAB 5 для студентов. - 2-е изд., испр. и дополн. - М.: ДИАЛОГ-МИФИ, 1999. - 448 с.
13. Потемкин В. Г. Система инженерных и научных расчетов MatLAB 5.x: - В 2-х т.Том 1. - М.: ДИАЛОГ-МИФИ, 1999. - 366 с.
14. Потемкин В. Г. Система инженерных и научных расчетов MatLAB 5.x: - В 2-х т. Том 2. - М.: ДИАЛОГ-МИФИ, 1999. - 304 с.
15. Рудаков П. И., Сафонов В. И. Обработка сигналов и изображений. MATLAB 5x. - М.: "ДИАЛОГ-МИФИ", 2000. - 413 с.

Абетковий покажчик функцій і блоків

A

abs 12, 14
Abs 113
acos 11, 113
acosh 11
acot 11
acoth 11
acsc 11
acsch 11
angle 14
ans 8, 10, 14, 85
asec 11
asech 11
asin 11, 113
asinh 11
atan 11, 113
atan2 ..... 11, 113
atanh 11
axis 34

B

BackLash 107
Band-Limited White Noise 87, 97
bar 29
besseli 12
besselj 12
besselk 12
bessely 12
beta 13
betainc 13
betaln 13
Bias 111
bicg 60
bicgstab 60
Bus Creator 109
Bus Selector 109

C

cart2pol 12
cart2sph 12
ceil 12
cgs 60
Chirp Signal 87, 95
chol 64
Clock 82, 87, 88, 118
Combinatorial Logic 115
comet 31
Complex to Magnitude-Angle 113
Complex to Real-Imag 113
cond 63
conj 14
Connections 128
Constant 83, 88, 118
Continuous 75
conv 23
corrcoeff 54
cos 11, 113
cosh 11, 113
cot 11
coth 11
Coulomb & Viscous Friction 108
cov 54
cplxpair 14
cross 21
csc 11
csch 11
cumprod 51, 52
cumsum 51, 52
Constant 86

D

Data Store Memory 109
Data Store Read 109
Data Store Write 109
Dead Zone 106
deconv 23
Demux 108
Derivative 99
det 63
diag 47
diff 52
Digital clock 87
Discontinuities 75
Discrete 75, 103
Discrete Derivative 104
Discrete Filter 103
Discrete FIR Filter 104
Discrete State-Space 104
Discrete Transfer Fcn 103
Discrete Zero-Pole 103
Discrete-Time Integrator 103, 104
disp 14
Display 77, 83, 88
Dot Product 113

E

eig 66
ellipj 13
ellipke 13
Enable 117
end 49
eps 10
erf 13
erfc 13
erfcx 13
erfinv 13
exp 11, 21, 113
expint 13
expm 61
expm1 61
expm2 61
expm3 61
eye 45

F

Fcn 115
fft 40, 42
fftshift 43
figure 35
filter 39
First-Order Hold 104, 105
fix 12
fliplr 46

flipud 46
floor 12
From 109
From File 87
From Workspace 88

G

Gain 111
gamma 13
gammainc 13
gammaln 13
gcd 13
gmres 60
Goto 109
Goto Tag Visibility 109
grid 27
Ground 87, 117
gtext 34

H

hadamard 45
hankel 47
help 12, 17
hess 67
hilb 45
hist 31
Hit Crossing 107
hold off 35
hold on 35
hypot 113

I

i 10
ifft 40, 41
imag 14
In 116, 117, 128
In 1 87
inf 10, 86
Integrator 99, 104
interp1 38
inv 57, 65
invhilb 45

J

j 10

L

lcm 13
legendre 13
log 11, 113
log10 ..... 11, 113
log2 13
Logic & Bit Operations 75
Logical Operator 114
loglog 31, 32, 33
logm 62
logspace 31
lu 64

M

Magnitude-Angle to Complex 114
Manual Switch 109
Math Function 113
Math Operations 110
MATLAB Fcn 115
max 51, 52
mean 51, 52
Memory 104, 105
Merge 109
min 51, 52
MinMax 113
mod 113
Multiport Switch 109
Mux 108

N

NaN 10
Nonlinear 75, 110, 112
norm 63
null 63

O

ones 44
orth 63
Out 77, 116, 117, 128

P

pascal .....
pcg 60
pi 10
pinv 65
plot 25, 33
pol2cart 12
poly 23, 66
polyder 25
polyeig 70
polyfit 36
polyval 24, 32
polyvalm 70
Ports & Subsystems 75, 116
pow 113
pow2 13
prod 51, 52
Product 112
Pulse Generator 86, 90

Q

qmr 60
qr 65
Quantizer 107
qz 68

R

Ramp 86, 91
rand 45, 48
randn 31, 45
Random Number 87, 96
rank 63
rat 13
rats 13
rcond 63
real 14
Real-Imag to Complex 114
realmax 10

realmin 10
reciprocal 113
Relational Operator 114
Relay 107
rem 12, 113
Repeating Sequence 87, 94
reshape 46
roots 23
rot90 46
round 12
Rounding Function 113
rref 63
rsf2csf 68

S

Saturation 106
schur 67
Scope 77, 78, 79, 94, 97
sec 11
sech 11
semilogx 32, 33
semilogy 32, 33
S-function 115
S-function Builder 116
sign 12
Sign 113
Signal Builder 86, 91
Signal Generator 86, 88, 89, 118
Signal Routing 75, 108
sin 11, 21, 113
Sine Wave 81, 82, 87, 92
sinh 11, 113
Sinks 75, 77
size 51, 52
Slider Gain 112
sort 51, 52
Sources 75, 86
sph2cart 12
spline 37
sqrt 12, 14, 113
sqrtm 62
square 113
State-Space 99, 101
std 51, 52
stem 29
Step 87, 93
Stop Simulation 77
subplot 34
subspace 70
Subsystem 117, 128, 129
sum 51, 52
Sum 111, 112
svd 66
Switch 109

T

tan 11, 21, 113
tanh 11, 113
Terminator 77, 117
text 34
title 28
To File 77, 85
To Workspace 77, 85
trace 63
Transfer Fcn 99, 102
Transfer Fcn First Order 104
Transfer Fcn Lead or Lag 104
Transfer Fcn Real Zero 104
Transport Delay 103
trapz 52
Trigger 117
Trigonometric Function 113
tril 47
triu 47

U

Uniform Random Number 87, 96
Unit Delay 103, 104
User-defined Functions 75, 115

V

Variable Transport Delay 103

X

xlabel 28
XYGraph 77, 82, 89, 118

Y

ylabel 28

Z

Zero-Order Hold 104
Zero-Pole 99, 102
zeros 44, 48

ЗМІСТ

ПЕРЕДМОВА **3**

1. КОМАНДНЕ ВІКНО MATLAB **3**

2. ОПЕРАЦІЇ З ЧИСЛАМИ **5**

2.1. Введення дійсних чисел 5
2.2. Найпростіші арифметичні дії 7
2.3. Введення комплексних чисел 9
2.4. Елементарні математичні функції 10
2.4. Спеціальні математичні функції 11
2.5. Елементарні дії з комплексними числами 12
2.6. Функції комплексного аргументу 13
2.7. Знайомство з програмуванням в Matlab 14

3. ОПЕРАЦІЇ З ВЕКТОРАМИ **17**

3.1. Введення векторів і матриць 17
3.2. Дії над векторами 19
3.2.1. Векторні дії над векторами 19
3.2.2. Поелементне перетворення векторів 20
3.2.3. Операції з поліномами 21
3.2.4. Виведення графіків. Процедура plot 24
3.2.5. Спеціальні графіки 28
3.2.6. Додаткові функції графічного вікна 32
3.2.7. Вставлення графіків до документу 34
3.2.8. Апроксимація та інтерполяція даних 35
3.2.9. Векторна фільтрація й спектральний аналіз 38

4. ОПЕРАЦІЇ З МАТРИЦЯМИ **44**

4.1. Формування матриць 44
4.2. Витягання й вставляння частин матриць 47
4.3. Обробка даних вимірів 50
4.4. Поелементне перетворення матриць 54
4.5. Матричні дії над матрицями 55
4.6. Розв'язування систем лінійних алгебричних рівнянь 57
4.7. Матричні функції 60
4.8. Функції лінійної алгебри 62

5. ПАКЕТ ПРОГРАМ ВІЗУАЛЬНОГО ПРОГРАМУВАННЯ SIMULINK **70**

5.1. Запуск SimuLink 71
5.2. Бібліотека SimuLink 73
5.2.1. Поділ Sinks (Приймачі) 75
5.2.2. Поділ Sources (Джерела) 84
5.2.3. Поділ Continuous (Неперервні елементи) 97
5.2.4. Поділ Discrete (Дискретні елементи) 101
5.2.5. Поділ Discontinuities (Розривні елементи) 104
5.2.6. Поділ Signal Routing (Пересилання сигналів) 106
5.2.7. Поділ Math Operations (Математичні операції) 108
5.2.8. Поділ Logic & Bit Operations (Логічні та бітові операції) 111
5.2.9. Поділ User-defined Functions (функції, що визначаються користувачем) 112
5.2.10. Поділ Ports & Subsystems (Порти та підсистеми) 113
5.3. Побудова блок-схем 115
5.3.1. Виділення об'єктів 115
5.3.2. Операції з блоками 116
5.3.3. Проведення з'єднувальних ліній 120
5.3.5. Створення підсистем 125
5.3.6. Зберігання і виведення до друку блок-схеми 127

ЛІТЕРАТУРА **128**

Абетковий покажчик функцій і блоків 129