МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

О.І. Толочко

ПАКЕТИ ПРИКЛАДНИХ ПРОГРАМ ДЛЯ ПЕОМ

Частина 1

MATLAB, SIMULINK, SIMPOWERSYSTEM ОСНОВИ ПРОГРАМУВАННЯ

Лабораторний практикум

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалвра
за спеціальністю 141 «Електроенергетика, електротехніка та електромеханіка»,

Київ
КПІ ім. Ігоря Сікорського
2020

Рецензент: *Островерхов М.Я.,* доктор технічних наук, професор, завідувач кафедри теоретичної електротехніки

Відповідальний редактор: *Марченко А.А.,* кандидат технічних наук, доцент кафедри автоматизації енергосистем

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол №10 від 18.06.2020 р.) за поданням факультету електроенерготехніки та автоматики (протокол № 5 від 24.02.2020 р.)

Електронне мережне навчальне видання

Толочко Ольга Іванівна, д-р техн. наук, проф.

MATLAB, SIMULINK, SIMPOWERSYSTEM ОСНОВИ ПРОГРАМУВАННЯ

MATLAB, Simulink, Simpowersystem. Основи програмування: лабораторний практикум [Електронний ресурс]: навч. посіб. для студентів спеціальності 141 «Електроенергетика, електротехніка та електромеханіка» з дисципліни «Пакети прикладних програм», ч. І, спеціалізація "Системи управління виробництвом і розподілом електроенергії" / О. І. Толочко; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 8241 кБ). – Київ: КПІ ім. Ігоря Сікорського, 2019. – 226 с.

Навчальний посібник присвячено питанням застосування студентами спеціальності 141 «Електроенергетика, електротехніка та електромеханіка» системи програмування *MATLAB* та її додатку *Simulink* з бібліотеками блоків *Simpowersystem*. Посібник отримує методичні вказівки до 16 лабораторних занять. Вказівки до кожного заняття складаються з теоретичних положень, завдання, рекомендацій щодо його виконання та контрольних запитань. Лабораторні роботи виконуються на комп'ютерах в середовищі пакету *MATLAB* та його додатку *Simulink*. Завдання до практичних занять спрямовані на застосування алгоритмічної мови програмування *MATLAB* і програми структурного математичного моделювання *Simulink* у купі з бібліотеками віртуальних фізичних блоків додатку *Simpowersystem* при розв'язанні прикладних задач з галузі електротехніки, електромеханіки, електроніки та енергетики.

Посібник призначений для використання бакалаврами спеціальності 141 «Електроенергетика, електротехніка та електромеханіка» при вивченні дисципліни «Пакети прикладних програм для ПЕОМ», ч.1 та можуть бути використані студентами навчальної освітньої програми при вивченні дисципліни «Математичні методи в електромеханіці» і «Моделювання та аналіз електромеханічних систем в МАТЛАБ», а також при виконанні курсових, бакалаврських та магістерських робіт.

ВСТУП

Знання сучасних програмних пакетів є невід'ємною складовою процесу підготовки кваліфікованих кадрів, оскільки такі знання суттєво підвищують конкурентоспроможність молодих фахівців. Тому доцільним є введення в навчальний процес дисциплін, що дають можливість широкого ознайомлення із можливостями таких пакетів.

Однією з найскладніших задач, що виникають при дослідженні складних динамічних систем керування енергетичними, електротехнічними та електромеханічними об'єктами, є розрахунок перехідних процесів. Їх можна розраховувати як за допомогою програм, написаних будь-якою алгоритмічною мовою, що потребує від дослідника достатньо високої кваліфікації в галузі програмування та обчислювальної математики, так і за допомогою спеціалізованого програмного забезпечення, що дозволяє користувачу задавати моделі у вигляді математичних рівнянь або у вигляді структурних схем, обирати методи розв'язання диференційних рівнянь та їх параметри в діалоговому режимі та отримувати результати у зручній формі.

Серед спеціалізованих систем програмування однією з найбільш розповсюджених натепер є система *MATLAB* фірми *Mathworks*, що отримує у своєму складі об'єктно-орієнтовану алгоритмічну мову, розвинутий графічний інтерфейс, засоби розв'язання задач лінійної алгебри, математичного аналізу, оптимізації, обчислювальної математики, цифрової обробки сигналів, структурного математичного та віртуального фізичного моделювання, аналізу і синтезу систем автоматичного керування та багато інших інструментів [1-6].

Основою для розробки моделей в середовищі програми структурного математичного моделювання *Simulink* пакета *MATLAB* є бібліотеки блоків, з котрих складаються структурні схеми систем автоматичного управління, що повинні бути дослідженими [7-9].

Наразі все більшу актуальність при дослідженні складних електротехнічних об'єктів набуває віртуальне фізичне моделювання, яке не

потребує від користувача знання математичного опису досліджуваного процесу, потребує менше часу для налаштування, є більш наочним, має розвинені засоби візуалізації та анімації результатів досліджень. Такі моделі можна створювати і в *Simulink* шляхом використання віртуальних блоків бібліотек *SimPowerSystems (SPS)*. Блоки бібліотеки *SPS*, призначені для моделювання електричних та електромагнітних кіл, електронних пристроїв, електродвигунів і ліній електропередач, подано у вигляді позначень відповідних елементів на принципових електричних схемах [10-12]. Математичний опис окремих елементів цих бібліотек приховано від користувача, завдяки чому створюється ілюзія віртуального фізичного моделювання.

Маючи структурну математичну або віртуальну фізичну модель досліджуваного об'єкта, складену в програмному середовищі пакета *MATLAB*, можна не тільки отримати графіки перехідних процесів, а і всебічно проаналізувати його властивості у просторі часу, у просторі змінної Лапласа, побудувати частотні характеристики та карти розташування нулів-полюсів.

Лабораторний практикум з дисципліни «Пакети прикладних програм» умовно можна поділити на 3 частини: 1) основи програмування в пакеті *MATLAB* (лабораторні роботи 1-8): 2) основи структурного математичного моделювання в середовищі програми *Simulink* пакета *MATLAB* (лабораторні роботи 9-14), 3) основи віртуального фізичного моделювання електротехнічних та електромеханічних пристроїв і систем з використанням блоків бібліотек додатку *SimPowerSystems* програми *Simulink* пакета *MATLAB* (лабораторні роботи 15-16).

Виконання кожної лабораторної роботи завершується оформленням і захистом звіту. Студенти, що не підготували вчасно звіт, до виконання наступної лабораторної роботи не допускаються. Форма семестрового контролю – диференційний залік.

Звіт з лабораторної роботи складається з титульного аркушу, завдання, викладення основних теоретичних положень, тексту програми та/або моделі досліджуваного об‘єкта, результатів виконання програми та/або результатів моделювання, висновків.

Звіт оформлюється згідно з правилами оформлення науково-технічної документації. На титульному аркуші має бути відображеною назва університету, назва дисципліни, номер та назва лабораторної роботи, шифр групи, ПІБ виконавця, ПІБ та регалії викладача.

Основні вимоги до оформлення

- поля – верхнє та нижнє по 2 см, ліве 2,5 см, праве 0,5 см;
- текстова частина – шрифт Times New Roman, 14 пт., 1,5 інтервали, вирівнювання по ширині, абзац 1,25 см:
- формули – 14 пт. з вирівнюванням по центру або з абзацу;
- рисунки – нумеровані з підрисуночними підписами;
- графіки – з позначеннями осей та необхідними для розуміння текстами;
- тексти на рисунках і схемах моделей повинні мати розмір 12 пт.

Лабораторна робота № 1

ОСНОВИ РОБОТИ В *MATLAB*

Мета роботи: знайомство з інтерфейсом користувача системи та основами роботи в режимі командного інтерпретатора.

1.1 Характеристика пакету

Систему програмування *MATLAB* використовують більш, ніж у 70 найвідоміших університетів світу, в т.ч. у Стенфордському, Каліфорнійському (США), Кембриджському (Англія), Кіото (Японія), Ейндховенському технічному університеті (Нідерланди), в Массачусетському та Хельсінському технологічних інститутах, а також в науково-дослідних центрах НАСА, в компаніях *Aerospace Corp., Boeing Aerospace, General Dynamics Corp., IBM, Lockheed, Siemens AG* та ін.

Історично *MATLAB* розроблявся фірмою *MathWorks* (США, Нейтік, штат Масачусетс) як діалогове середовище для матричних обчислень (*MATrix LABoratory*). Поступово пакет був оснащений гарною графічною системою, доповнений засобами лінійної алгебри від *LinPack*, символьної математики від *Maple*, підсилений засобами обчислювальної математики, додатками для структурного (*Simulink*) та віртуального фізичного моделювання електротехнічних, електромеханічних та електронних пристроїв (*SimPowerSystem*) та різноманітними інструментами (*Toolboxes*), призначеними для ефективного розв'язання спеціальних задач. Серед цих інструментів для фахівців у галузі електротехніки та енергетики найбільш інтересними є засоби для аналізу та синтезу систем керування лінійними і нелінійними об'єктами (*Control, Nonlinear, Robust, Predictive Toolboxes*), розв'язання задач оптимізації (*Optim*), ідентифікації (*Ident*), використання нейронних мереж (*Neuro*) та нечіткої логіки (*Fuzzy*), майстерня реального часу (*Real Time Workshop*).

Перша версія *MATLAB* була написана на Фортрані (*Cleve Moler*) на початку 60-х років. До 3-еї версії пакет працював під ОС *MS DOS*, починаючи з

4-ої – під *Windows*). Версії під *Windows* написані на *C* (автори: інтерпретатор – *Steve Bangert*, графіка – *Steve Kleiman*, більшість функцій – *John Little* і *Cleve Moler*). Може конвертувати файли, написані на Фортрані та *C*.

Отже, до складу *MATLAB* входять інтерпретатор команд, графічна оболонка, текстовий редактор з відладчиком, бібліотеки команд, компілятор, символьне ядро пакета *Maple*, математичні бібліотеки *MATLAB* на *C/C++*, генератор звітів и багатий інструментарій (*Toolboxes*).

Інтерфейс MATLAB відповідає сучасним канонам (див. рис. 1.1, 1.2). Після запуску пакету на екрані монітору з'являється робочий стіл (***Desktop***), що складається із заголовку, головного меню, панелі інструментів і комбінованого вікна. Останнє отримує 4 панелі: ***Command Window*** (вікно команд), ***Command History*** (історія команд), ***Workspace*** (робочий простір – оперативна пам'ять), ***Current Folder*** (поточна папка).

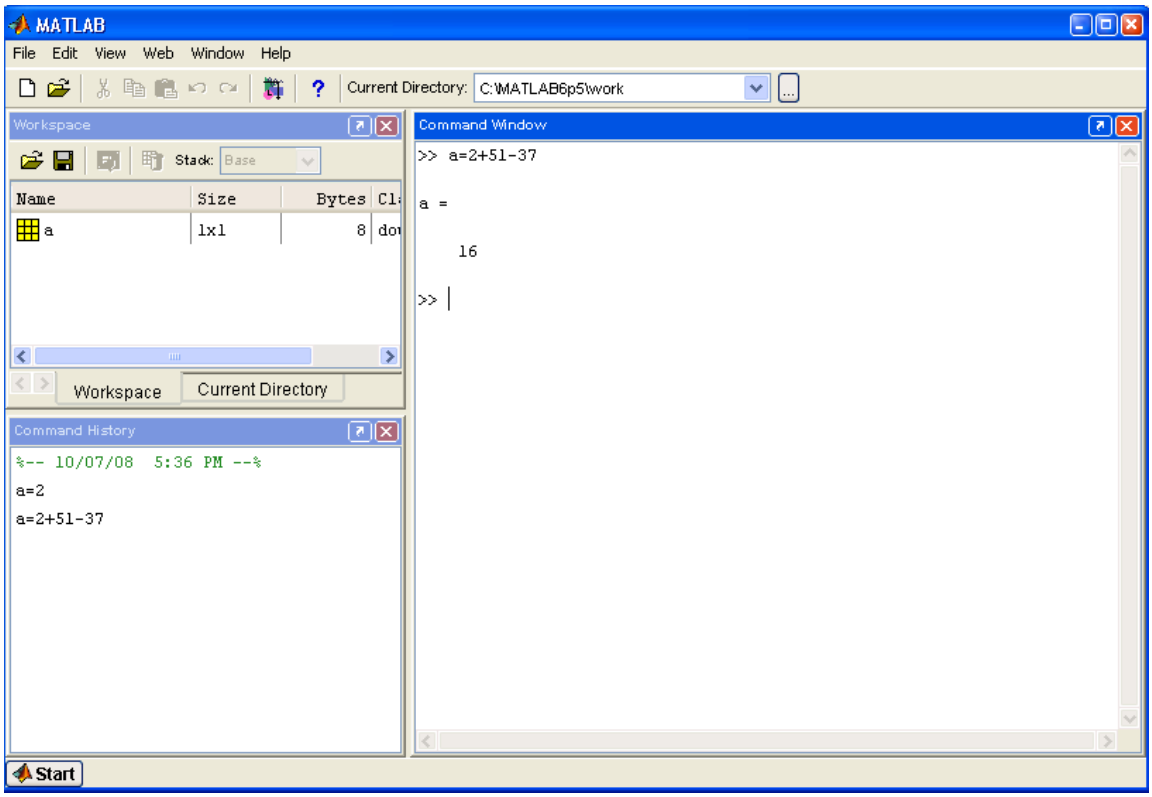


Рис. 1.1. Робочий стіл системи програмування *MATLAB 6.5*

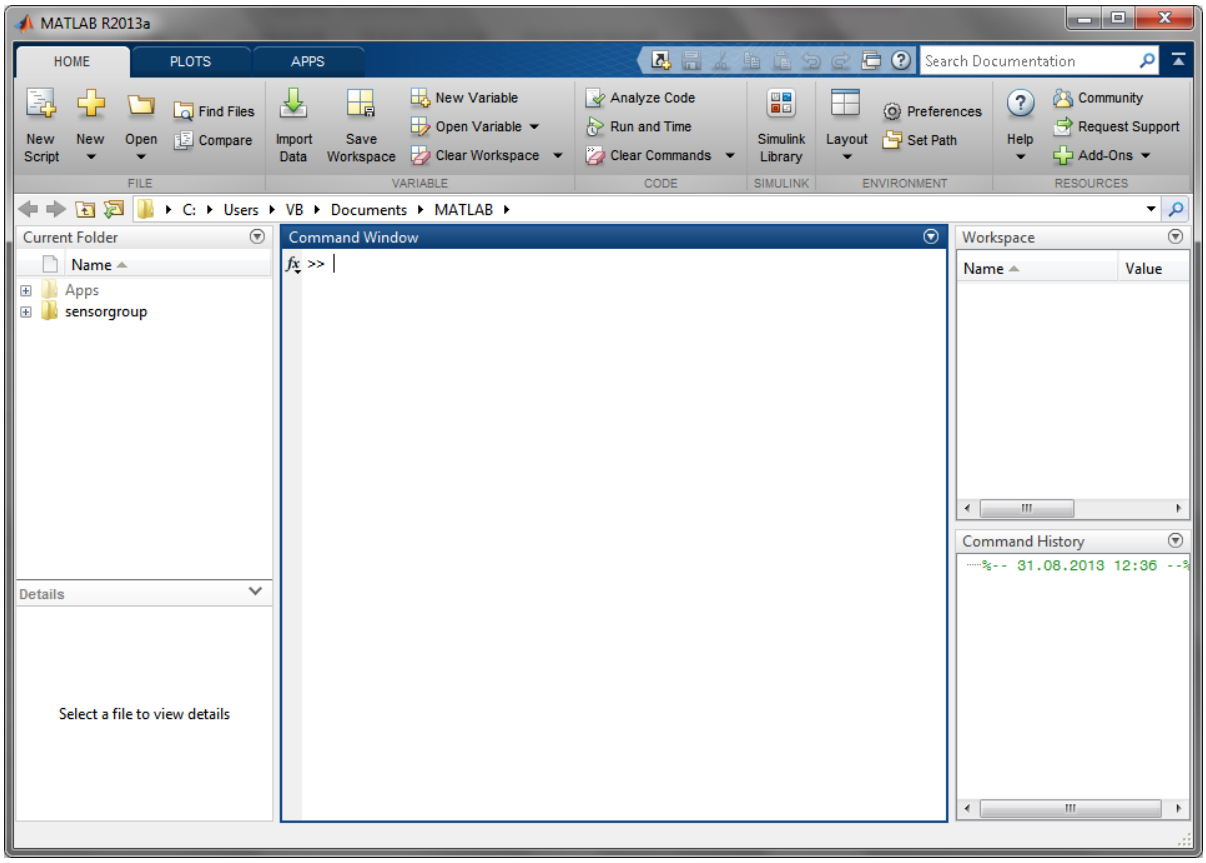


Рис. 1.2. Робочий стіл системи програмування *MATLAB 13a*

В *Command Window* набираються команди, які виконуються одразу після натискання клавіші *Enter*, видаються результати виконання цих команд, виводяться повідомлення системи.

Вікно *Workspace* відображує поточні змінні, та їх значення.

Вікно *Command History* зберігає всі команди, що набирає користувач у командному вікні.

Конфігурацію вікон на екрані та їх склад можна змінювати звичайними засобами, прийнятими у *Windows* та за допомогою команд меню *View*. Щоб відновити конфігурацію, прийняту за замовчанням, треба через функції меню пройти шлях: ***Desktop → Desktop Layout → Default.***

Одним із найпростіших режимів роботи пакету *MATLAB* є режим командного інтерпретатора, в якому інструкції вводяться безпосередньо у вікні *Command Window* при наявності у командному рядку запрошення «>>»

натисканням клавіші *Enter.* При цьому виконується трансляція введеної команди у машинний код і її виконання без запам‘ятовування машинного коду. Команди, розташовані в одному рядку, відокремлюються один від одного символами "," або ";". Якщо команда або оператор не поміщається в одному рядку, її можна продовжити в наступному рядку, використовуючи у якості оператору переносу три або більше точок поряд без пробілів ("…").

Деяка кількість введених команд запам‘ятовується в буфері, з якого їх можна викликати по черзі у зворотному порядку клавішею "↑".

Після виконання введеної команди інтерпретатор готовий до прийому наступної команди, про що свідчить наявність у новому рядку командного вікна символу запрошення "»".

Командне вікно *MATLAB* при необхідності може бути очищено командою clc (*clear screen*).

Щоб зберегти сеанс роботи в режимі командного інтерпретатора у файлі, можна перед початком сеансу виконати команду

diary FileName

або

diary (‘FileName’),

а в кінці сеансу – команду

diary off.

В результаті таких дій у поточній папці буде створено текстовий файл-блокнот, в який будуть занесені як введені команди, так і результати їх виконання, за виключенням графічної інформації. У наступному сеансі можна вивести цей файл у вікно *Command Window* командою type FileName. Вміст файлу можна скопіювати у буфер (^*c*) та включити (^*v*) у файл звіту **.doc* або **.docx*.

1.2 Файлова система пакету *MATLAB*

Основні папки системи *MATLAB 13a* відображені на рис. 1.3.

Серед них слід звернути увагу на такі папки:

- *bin* – командна папка, що утримує у сові файл запуску пакета *matlab.exe*;
- *simulink* – папка додатку для структурного математичного моделювання;
- *toolbox* – папка інструментів для розв'язання різноманітних математичних задач (формується у відповідності з даними, обраними при інсталяції), отримує папки з файли **.m*, що називаються *m*-функціями, і один *m*-файл коментарів (*Contents.m*), в якому кратко (в один рядок) описано призначення функцій, що входять до даної папки (директорії). Більш детальна інформація про *m*-функції знаходиться в коментарях самих функцій. Будь-яку функцію можна завантажити у текстовий редактор або вивести на екран командою type fun, де fun – ім'я функції.

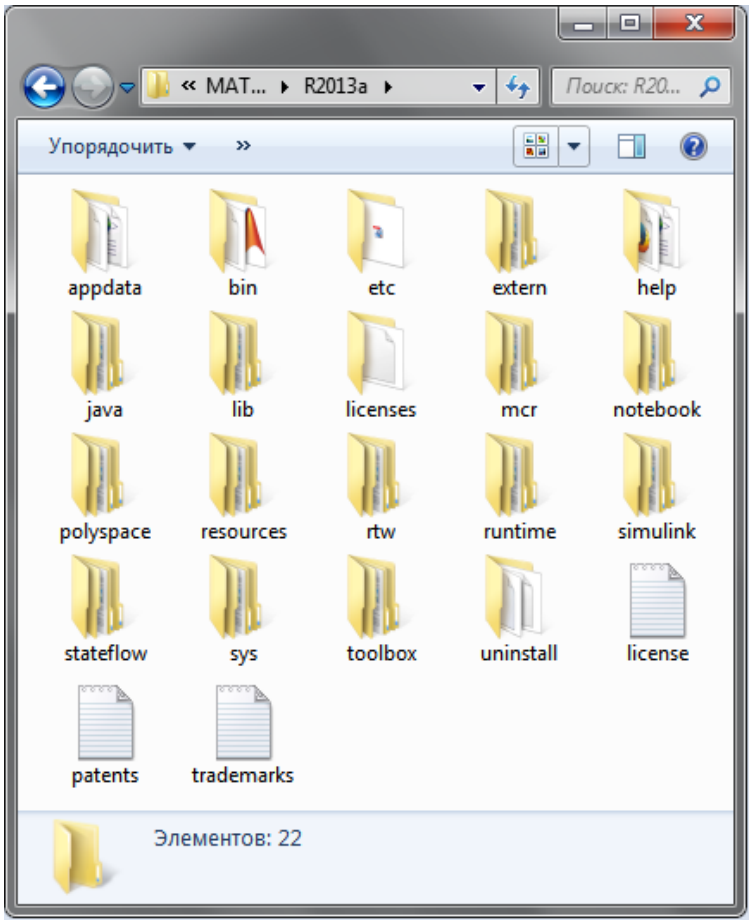


Рис. 1.3. Папки системи *MATLAB*

Кожна *m*-функція, що знаходиться у будь-якій папці інструментів, після оператора заголовку, що починається з ключового слова function отримує розділ коментарів, з яких автоматично формується файл та інформація яка виводиться на екран при зверненні до допомоги. Коментарі починаються з символу "%", дія якого розповсюджується до кінця рядку.

Перший рядок коментаря складається з коротенької інформації про призначення функції. Ця інформація заноситься у файл *Contents.m* та виводиться при виконанні команди help folder, де folder – ім‘я відповідної папки. Наступні рядки коментаря складаються з більш детальної інформації про функцію та її параметри. Вони виводяться при виконанні команди help fun, де fun – ім‘я відповідної функції.

Для знайомства з можливостями пакету корисно ознайомитися з основними папками інструментів. Серед них в першу чергу слід звернути увагу на такі:

elfun – елементарні математичні функції;

elmat – елементарні матриці та вектори та маніпуляції над ними;

datafun – аналіз даних та перетворення Фур‘є;

matfun – операції лінійної алгебри;

ops – математичні операції та спеціальні символи;

general – команди загального призначення;

lang – конструкції алгоритмічної мови;

helptools – функції допомоги;

graph2d – двовимірна графіка;

graph3d – тривимірна графіка;

graphics – обробка графіків (допоміжні графічні функції);

specgraph – спеціальна графіка;

polyfun – операції над степеневими поліномами;

funfun – функції над функціями;

optimfun – функції оптимізації.

1.3 Імена змінних. Константи.
Формати виведення інформації на екран

Імена змінних (ідентифікатори) в середовищі *MATLAB* повинні складатися з латинських букв, цифр та знаку підкреслення і починатися з букви, наприклад, x, X, Alpha1, y_min, c2dim, Number_of_Iter. Кількість символів у імен може бути довільною але *MATLAB* аналізує тільки перші символи кількістю 31 і розрізняє заголовні та прописні букви.

Арифметичні константи розподіляються на такі:

- цілі (-16, +5, 283);
- дійсні у форматі з фіксованою крапкою, тобто у природній формі, коли ціла частина відокремлюється від дробової «крапкою» (25.6, -138.54);
- дійсні у форматі з плаваючою точкою, коли мантиса числа *M* відокремлюється від його порядку *p* символом «e», тобто числу у формі $M \times 10^p$ відповідає константа Mep (0.25e-8, -35e12);
- комплексні у трьох попередніх формах з позначенням уявної частини символами i або j (2.5-5i, -0.2+1e-3j).

MATLAB має ряд ***зарезервованих констант*** (табл.1.1).

Таблиця 1.1

Ім'я константи	Значення константи
ans	answer – результат останньої операції
i, j	sqrt(-1) – уявна одиниця
pi	число π, обчислюється як 4*atan(1) або imag(log(–1))
eps	машинна точність – 2.2204e-16
realmax	модуль максимального дійсного числа – 1.7977e+308
realmin	модуль мінімального дійсного число – 2.2251e-308
intmax	максимальне ціле число – 2147483647
intmin	мінімальне ціле число – -2147483648
namelengthmax	максимальна довжина імені – 63
Inf	Infinity – нескінченність (number/0)
NaN	Not a Number – невизначеність (0/0, inf/inf, inf-inf, 0^inf, ...)
end	найбільше значення індексу масиву

Для виведення чисел на екрані є різні формати. Потрібний формат може бути визначений в меню (*File/Preferences*) або за допомогою команди format. Найбільший інтерес мають формати, подані у табл. 1.2.

Таблиця 1.2

Формат	**Подання**
short e	Число в експоненціальній формі (з плаваючою крапкою) з мантисою із 5 цифр і показником степені із 3 цифр
short	Число у природній формі (з фіксованою крапкою) з 4 цифрами після крапки або в форматі *short e*, якщо формат *short* не підходить для відображення значення числа (за замовчанням)
long e	Число в експоненціальній формі з мантисою із 16 цифр і показником степені із 3 цифр
long	Число у природній формі з 16 цифрами після крапки або в форматі *long e*
rat	Число у вигляді раціонального дробового числа
loose	Інформація виводиться просторо – з пропуском одного рядка між рядками (за замовчанням)
compact	Інформація виводиться щільно (без пропуску рядків)

Строкові константи (рядки символів) складаються з будь-яких символів, розташованих між апострофами (‘Hello!’, ‘2+3=5?’, ’Напруга, В’).

1.4 Операція присвоєння

Символом операції присвоєння є "=". Операція має формат

VarName = expr

де VarName – ім‘я змінної, expr – вираз. При виконанні такого оператора виконується обчислення виразу у правій частині оператора, а результат заноситься в комірку пам‘яті, виділеної для змінної, ім‘я якої записано у лівій частині. Результат виводиться на екран. Якщо оператор присвоєння закінчується символом «;», то виведення результату не виконується.

Вираз уявляє собою сукупність констант, змінних та звернень до функцій, роз‘єднаних знаками операцій. Усі змінні, застосовані у правій частині у момент виконання оператора повинні мати значення. Окремим випадком виразу може бути константа, змінна або звернення до функції, наприклад,

```
>> y=sind(30) % Натискаємо на Enter – бачимо результат
y =
   0.5000
>>x=4+3; % Натискаємо на Enter – результат не виводиться
>>x          % Натискаємо на Enter – бачимо значення введеної змінної
x =
    7
```

Усі змінні з визначеними значеннями зберігаються у *робочому просторі* (*Workspace*) середовища *MATLAB*, тобто в оперативній пам‘яті комп‘ютера. Дані з можна видалити командою clear, а можна і зберегти у файлі **.mat* (за замовчанням командою *matlab.mat*) командою load.

При так званому неявному присвоєнні ліва частина оператора та знак «=» в конструкції оператора відсутній. У такому випадку результат операції присвоюється зарезервованій змінній з ім‘ям ans (від *answer* – відповідь), наприклад,

```
>> 5.32*pi
ans =
  16.7133
>> sin(5)
ans =
  -0.9589
```

Присвоєння змінним комплексних значень виконується аналогічно:

```
>> z=3+4*i; x=2.5-6*j; y=4*exp(i*pi/8);
>> x
x =
  2.5000 - 6.0000i
>> y
y =
  3.6955 + 1.5307i
```

При роботі зі скалярними арифметичними даними у виразах використовуються операції додавання або уточнення знаку +, віднімання або зміни знаку -, множення *, ділення / та підвищення у степінь ^, а також операції порівняння >, <, >=, <=. ==, ~= і логічні операції |, &. ~, які будуть розглянуті пізніше.

Із функцій, що можуть входити у математичні вирази, у першу чергу розглянемо елементарні функції, що знаходяться у папці elfun. Для знайомства з ними наберемо команду

help elfun

В результаті отримуємо інформацію, з якої випливає призначення таких функцій:

- ***тригонометричні з аргументом у радіанах***: sin(x) – $\sin x$, cos(x) – $\cos x$, tan(x) – $\mathrm{tg}\,x$, cot(x) – $\mathrm{ctg}\,x$, sec(x) – $\sec x$, csc(x) – $\mathrm{cosec}\,x$;
- ***тригонометричні з аргументом у градусах (degree)***: sind(x), cosd(x), tand(x), cotd(x), secd(x), cscd(x);
- ***зворотно тригонометричні зі значенням у радіанах***: asin(x) – $\arcsin x$, acos(x) – $\arccos x$, atan(x) – $(-\pi/2)\le\mathrm{arctg}\,x\le\pi/2$, atan2(y,x) – $(-\pi)\le\mathrm{arctg}\,y/x\le\pi$, acot(x) – $\mathrm{arcctg}\,x$, asec(x) – $\mathrm{arcsec}\,x$, acsc(x) – $\mathrm{arccosec}\,x$;
- ***зворотно тригонометричні зі значенням у градусах***: asind(x), acosd(x), atand(x), atand2(y,x), acotd(x), asecd(x), acscd(x);
- ***гіперболічні***: sinh(x) – $\mathrm{sh}\,x=(e^x-e^{-x})/2$, cosh(x) – $\mathrm{ch}\,x=(e^x+e^{-x})/2$, tanh(x) – $\mathrm{th}\,x=(e^x-e^{-x})/(e^x+e^{-x})$, coth(x) – $\mathrm{cth}\,x=(e^x+e^{-x})/(e^x-e^{-x})$, sech(x) – $\mathrm{sech}\,x=2/(e^x-e^{-x})$, csch(x) – $\mathrm{cosech}\,x=2/(e^x+e^{-x})$;
- ***експоненційні, логарифмічні та степеневі***: exp(x) – e^x, log(x) – $\ln x$, log10(x) – $\lg x$, log2(x) – $\log_2 x$, pow2(x) – 2^x, sqrt(x) – $\sqrt{x}$, nthroot(x,n) – $\sqrt[n]{x}$;
- ***функції комплексних аргументів*** $z=x+jy=Ae^{j\varphi}$: abs(z) – $|z|=\sqrt{x^2+y^2}=A$, angle(z) – $\varphi=\mathrm{arctg}(y/x)$, real(z) – x, imag(z) – y, complex(x,y) – $z=x+jy$, conj(z) – $x-jy=Ae^{-j\varphi}$.
- ***округлювання та залишки***: fix(x) – округлювання у напрямку до 0 (дробова частина відкидається); floor(x) – округлювання у напрямку до $-\infty$ (до

найближчого меншого цілого); ceil(x) – округлювання у напрямку до $+\infty$ (до найближчого більшого цілого); round(x) – округлювання до найближчого цілого; rem(x,y) – залишок від цілочислового ділення x на y; sign(x) – знакова функція (дорівнює +1 при x>0, 0 при x=0 та -1 при x<0).

До цього переліку можна додати ще декілька функцій цілочислових аргументів із папки specfun: factor(n) – $n!$; gcd(m,n) – найбільший загальний дільник (*Greatest Common Divisor*); lcm(m,n) – найменший загальний множник (*Least Common Multiple*).

1.5 Генерація векторів та матриць

Основним типом даних в є матриця, тобто двовимірний масив, який можна подати у вигляді таблиці з декількома рядками та стовпцями. Окремим випадком матриці є одновимірні масиви (вектор-рядок та вектор-стовпець) та скаляри, тобто одиночні дані.

При поелементному формуванні векторов і матриць їх елементи поміщають у квадратні дужки [ ], елементи одного рядка відокремлюють один від одного комами «,» або пробілами «␣«, а рядки – крапкою з комою «;» або переводом рядку клавішею *Enter*, наприклад,

```
» x=[2 3 –8]; % Вектор-рядок
» y=[5; 6.5; -2.23; 0];        % Вектор-стовпець
» A=[1,2,3; 4 5 6];   % Матриця 2*3
» B=[1 2; 3 4; 5 6];  % Матриця 3*2
```

Вектори-рядки з рівномірним розподіленням елементів можна утворювати двома способами.

1) Оператори

VarName = InitValue : Step : EndValue

та

VarName = InitValue : EndValue

створюють вектори-рядки за заданими значеннями першого InitValue та останнього EndValue елементів та різницею Step між двома сусідніми елементами. У другому випадку за замовчанням (*default*) Step=1. Наприклад,

```
» z=2:5
z =
    2    3    4    5
» w=-1:0.5:1
w =
  -1.0000   -0.5000    0    0.5000    1.0000
```

Кількість елементів у рядку Number можна підрахувати за формулою:

Number = ((EndValue – InitValue) / Step)+1

Рядки з однаковою кількістю елементів можна поєднувати у матриці:

```
» c=[1:4; 2:5]
c =
     1     2    3    4
     2     3    4    5
```

2) Оператор

VarName = linspace (InitValue, EndValue, Number)

створюють рівномірно розподілені вектори-рядки за заданими значеннями першого InitValue і останнього EndValue елементів та їх кількістю Number. За замовчанням Number = 100. Оператори linspace зручно використовувати при завданні діапазону зміни аргументу при побудові графіків, наприклад,

```
>> a=linspace(3,7,10)
a =
   3.0000    3.4444    3.8889    4.3333    4.7778    5.2222    5.6667    6.1111    6.5556
7.0000

>> x=linspace(0,2.5)
x =
 Columns 1 through 10
      0    0.0253    0.0505    0.0758    0.1010    0.1263    0.1515    0.1768    0.2020    0.2273
 Columns 11 through 20
   0.2525    0.2778    0.3030    0.3283    0.3535    0.3788    0.4040    0.4293    0.4545    0.4798
 Columns 21 through 30
   0.5051    0.5303    0.5556    0.5808    0.6061    0.6313    0.6566    0.6818    0.7071    0.7323
 Columns 31 through 40
   0.7576    0.7828    0.8081    0.8333    0.8586    0.8838    0.9091    0.9343    0.9596    0.9848
 Columns 41 through 50
   1.0101    1.0354    1.0606    1.0859    1.1111    1.1364    1.1616    1.1869    1.2121    1.2374
```

```
Columns 51 through 60
  1.2626   1.2879   1.3131   1.3384   1.3636   1.3889   1.4141   1.4394   1.4646   1.4899
Columns 61 through 70
  1.5152   1.5404   1.5657   1.5909   1.6162   1.6414   1.6667   1.6919   1.7172   1.7424
Columns 71 through 80
  1.7677   1.7929   1.8182   1.8434   1.8687   1.8939   1.9192   1.9444   1.9697   1.9949
Columns 81 through 90
  2.0202   2.0455   2.0707   2.0960   2.1212   2.1465   2.1717   2.1970   2.2222   2.2475
Columns 91 through 100
  2.2727   2.2980   2.3232   2.3485   2.3737   2.3990   2.4242   2.4495   2.4747   2.5000
```

У таких рядках крок між двома сусідніми елементами Step можна підрахувати за формулою:

Step = (EndValue – InitValue) / (Number -1)

Вектори рядки, рівномірно розподілені впродовж логарифмічної осі, можна сформувати оператором

VarName = logspace (pInit, pEnd, Number)

який створює вектори-рядок кількістю Number елементів (за замовчанням Number=50), перший з яких дорівнює 10^{pInit}, останній – 10^{pEnd}, а інші розраховуються у такий спосіб, щоб відношення кожного наступного елементу до попереднього було константою 10^{pStep}, де

pStep = (pEnd – pInit) / (Number -1)

Наприклад,

```
>> w = logspace (-1,3,5)
w =
  1.0e+03 *
   0.0001   0.0010   0.0100   0.1000   1.0000
>> pw=log10(w)
pw =
   -1    0    1    2    3
>> k = logspace(log10(2),log10(2^9),9)
k =
   2.0000    4.0000    8.0000   16.0000   32.0000   64.0000  128.0000  256.0000
512.0000
>> pk = log10(k)
pk =
   0.3010   0.6021   0.9031   1.2041   1.5051   1.8062   2.1072   2.4082   2.7093
>> omega=logspace(-1,2)
omega =
```

```
  Columns 1 through 10
    0.1000    0.1151    0.1326    0.1526    0.1758    0.2024    0.2330    0.2683    0.3089
0.3556
  Columns 11 through 20
    0.4095    0.4715    0.5429    0.6251    0.7197    0.8286    0.9541    1.0985    1.2649
1.4563
  Columns 21 through 30
    1.6768    1.9307    2.2230    2.5595    2.9471    3.3932    3.9069    4.4984    5.1795
5.9636
  Columns 31 through 40
    6.8665    7.9060    9.1030   10.4811   12.0679   13.8950   15.9986   18.4207   21.2095
24.4205
  Columns 41 through 50
   28.1177   32.3746   37.2759   42.9193   49.4171   56.8987   65.5129   75.4312   86.8511
100.0000
>> p_omega=log10(omega)
p_omega =
  Columns 1 through 10
   -1.0000   -0.9388   -0.8776   -0.8163   -0.7551   -0.6939   -0.6327   -0.5714   -0.5102   -
0.4490
  Columns 11 through 20
   -0.3878   -0.3265   -0.2653   -0.2041   -0.1429   -0.0816   -0.0204    0.0408    0.1020
0.1633
  Columns 21 through 30
    0.2245    0.2857    0.3469    0.4082    0.4694    0.5306    0.5918    0.6531    0.7143
0.7755
  Columns 31 through 40
    0.8367    0.8980    0.9592    1.0204    1.0816    1.1429    1.2041    1.2653    1.3265
1.3878
  Columns 41 through 50
    1.4490    1.5102    1.5714    1.6327    1.6939    1.7551    1.8163    1.8776    1.9388
2.0000
```

Деякі спеціальні матриці створюються такими функціями:

zeros – матриця, у якої всі елементи дорівнюють 0;

ones – матриця, у якої всі елементи дорівнюють 1;

rand – матриця, складена із випадкових чисел з рівномірним розподілом;

randn – матриця, складена із випадкових чисел з нормальним розподілом.

Усі вони мають однаковий формат, який розглянемо на прикладі функції zeros:

zeros(n) – формує квадратну матрицю розміром $n \times n$;

zeros(m, n) – формує прямокутну матрицю розміром $m \times n$;

zeros(1, n) – формує вектор-рядок із n елементів;

zeros(m, 1) – формує вектор-стовпець із m елементів;

zeros(size(A)) – формує матрицю, такого ж розміру, як матриця A.

Наприклад,

```
» Z = zeros(2)
Z =
    0    0
    0    0
» zeros(1,3)
ans =
    0    0    0
» ones(size(Z))
ans =
    1    1
    1    1
>> rand(2,3)
ans =
   0.8147    0.1270    0.6324
   0.9058    0.9134    0.0975
>> v = randn(1,5)
v =
   3.0349    0.7254   -0.0631    0.7147   -0.2050
```

До функцій утворення спеціальних матриць та векторів, також належать

eye(n) – формує одиничну діагональну матрицю розміром n×n:

diag(A) – формує з елементів головної діагоналі матриці A вектор-стовпець;

diag(X) – формує діагональну матрицю, застосовуючи у якості головної діагоналі вектор X:

```
» E = eye(3)
E=
    1    0    0
    0    1    0
    0    0    1
» A = [1 2 3; 4 5 6; 7 8 9];
» V = diag(A)
V =
    1
    5
    9
```

```
» Av =  diag(V)
Av =
    1    0    0
    0    5    0
    0    0    9
```

До елементів масивів звертаються за ім‘ям масиву та списку індексів, поміщеному між круглими дужками, наприклад, C(2,3) – елемент 2-го рядка та 3-го стовпця матриці C, x(3) – 3-ій елемент вектору x. Останнє значення будь-якого індексу можна позначати як end: C(end,1) – останній елемент першого стовпця матриці C.

Використання замість одного з індексів символу «:» означає, що цей індекс перебігає усі свої значення. Це дозволяє звертатися до окремих рядків та стовпців матриць: H(:,3) – 3-ій стовпець матриці H, H(1:3,:) – перші 3 рядки матриці, H(:,[2,4]) – 2-ий та 4-ий стовпці матриці.

Пуста (порожня) матриця [ ] у правій частині оператору присвоєння знищує (вилучає) елементи, зазначені у лівій частині, наприклад, операція A(2,:)=[ ] викидає з матриці A другий рядок.

1.6 Завдання

В режимі командного інтерпретатора

1) ввести імена зарезервованих констант і подивитися їх значення, змінюючи формат виведення чисел;
2) виконати довільні оператори явного і неявного присвоєння;
3) створити довільну матрицю розміром 3×5, вектор-рядок із 5 елементів та вектор-стовпець із 3 елементами; сформувати матрицю приєднанням до матриці вектору знизу; сформувати матрицю вставкою між 2-м та 3-м стовпцями вектору ; утворити матрицю, усуненням із матриці 1-го рядка та останнього стовпчика; замінити елемент 3-го стовпчика 2-го рядка нулем;
4) сформувати спеціальні вектори та матриці, описані в теоретичній частині;
5) розрахувати значення виразів, заданих у табл. 1.2;
6) апробувати дію команд version, ver, who, whos, clear, which, beep.

Сеанс зберегти у блокноті.

1.7 Методичні вказівки

1. Перед розробкою програмного файлу створіть власну папку користувача за шляхом *C:\Users\UserName\Documents\MATLAB\...* та приєднайте її до списку дозволених папок командою меню *Set Path→Add Folder…→Save*.
2. Для завантаження текстового редактора скористайтеся командою меню *New→Script*.
3. Збережіть створену програму у власній папці.
4. ***Ім‘я користувача UserName не повинно отримувати символів кирилиці. Ім‘я файлу також, як і імена змінних, складайте з латинських букв, цифр та знаку підкреслення (перший символ – буква).***
5. Після опробування команди format поверніться у стан format scort, format compact.
6. При оформленні звіту додайте до операторів коментарі.
7. ***Списки індексів та параметрів функцій оточуйте круглими дужками та відокремлюйте один від одного комою «,».***
8. При виконанні завдання 5 не забувайте про пріоритети виконання арифметичних операцій та за необхідністю змінюйте цей порядок за допомогою круглих дужок: слідкуйте за тим, щоб кількість відкриваючих та закриваючих дужок була однаковою. Для перевірки правильності розрахунку значення виразу з табл. 2.1 розбийте вираз на декілька частин та виконайте розрахунок поетапно.
9. Після написання імені функції одразу записуйте в дужках її аргументи і тільки потім виконуйте зі вже розрахованою функцією будь-які операції.
10. При будь яких непорозуміннях користуйтеся командами help**,** **doc** та demo**.**

Таблиця 1.2

№	Вираз	Значення змінних
1	$y=\sin\frac{a-x}{c}+10^4\sqrt[3]{\frac{a-kx^2}{2b}}+\frac{\cos kx^2}{tg3}-\frac{bc}{ax}$	$a=-1.3; b=0.91;$ $c=0.75; x=2.32; k=8.$
2	$y=-\frac{(x-d)(x^2+b^2)}{\sqrt[3]{x^2+b^2-cd}}+10^{-3}tgkn-\frac{\cos kx}{\sin 5}.$	$d=1.25; b=0.75; n=4;$ $c=2.2; x=0.32; k=2.$
3	$y=tgik+10^3e^{-5}+\sqrt[3]{\frac{10^2\lvert xk\rvert}{(a+b)^2}}-\frac{ax^3-b}{(a+b)^2}.$	$i=5; b=2.35;$ $a=25.2; x=0.1; k=-2.$
4	$y=\frac{\sqrt{\lvert c-d\rvert+(a+c)^2}}{\sin 2i}+10^{-3}e^{ix}-\frac{\lvert c-d\rvert+a^2}{\sqrt[3]{(a+c)^2}}.$	$a=-1.25; d=2.5; i=5;$ $c=0.05; x=1.35.$
5	$y=\frac{\ln\lvert kx\rvert}{2\sin(\pi/3)}-\sqrt{\lvert x-a^2\rvert}-\frac{10^4a-b}{\cos kx}+\sqrt[3]{x-a^2}+c^3x.$	$a=0.93; b=5.61;$ $c=0.31; x=-2.5;$ $k=2.$
6	$y=10^4\frac{ax}{b^2}-\left\lvert\frac{a-b}{kx}\right\rvert+\frac{\ln 3}{\sqrt[3]{ax^2+b^2}}-e^{-kx}$	$b=0.35; a=3.5;$ $x=1.523; k=-2.$
7	$y=-\frac{\lvert b-a\rvert}{kx}+10^4\sqrt[5]{\lvert\cos kx\rvert}+\sqrt{\frac{abc}{2.4}}-\frac{0.7abc}{\sin(32^\circ)}.$	$a=1.7; b=-1.25;$ $c=-0.3; x=2.5; k=3.$
8	$y=\frac{\lvert a^2-b^2\rvert}{\sin kx}+10^4\sqrt[5]{\lvert\sin kx-bc\rvert}-\frac{k^2+\mathrm{tg}3k}{e^{kx}}.$	$a=1.3; b=2.42;$ $c=0.83; x=1.5; k=2.$
9	$y=\frac{\sqrt[3]{\ln x+a^2}}{0.47x^2}-\left\lvert 0.47x^2-\frac{10^4}{7}\cos^2 k\right\rvert-\frac{c}{e^x}$	$c=1.52; a=-2.4;$ $x=0.29; k=3.$
10	$y=\frac{1.5(a-b)^2}{\lvert a-b\rvert c}+\frac{e^{-x/2}}{5}+10^3\sqrt{\lvert a-b\rvert}-\frac{(a+x^2)\cos(72^\circ)}{ix^2+a^2bc}$	$a=-2.5; b=1.35; i=3;$ $c=-0.72; x=2.75.$
11	$y=10^4\sin^2 i-\frac{0.32x^3+4x+b}{\cos ia}\sqrt[6]{0.32x^3-b}+\lvert b\rvert$	$a=3.5; b=-0.7; i=2;$ $x=0.8.$
12	$y=-\frac{\cos i}{\sin kx}+\frac{ax^2+\lvert d\rvert}{(a+b)^2}-10^4\sqrt[6]{\frac{kx}{(a+b)^2}}.$	$d=-0.01;\ b=1.25;$ $a=4.72;\ i=2;$ $x=2.25;\ k=3.$
13	$y=\cos k(x-a)+10^{-4}\frac{(x+a)^3+x^4d}{k(x-a)^3}+\frac{\sqrt[5]{\lvert x+a\rvert}}{2.4b}.$	$d=0.95; b=0.05;$ $a=-3.25; x=8.2; k=4.$
14	$y=\sqrt[5]{\lvert ax^2-b^3\rvert}+\ln kx-\frac{e^{kx}+c^2}{\sin kx}-10^{-3}\sqrt{2157}.$	$c=1.72; b=-0.31;$ $a=2.01; x=0.48; k=3.$

1.8 Контрольні питання та завдання

1. Які базові інструменти пакету *MATLAB* ви знаєте?
2. Звідки береться інформація для команди help.
3. Перелічте зарезервовані константи *MATLAB*.
4. Як установити формат виведення числової інформації?
5. Перелічте основні елементарні математичні функції пакету *MATLAB*.
6. Як утворюються матриці та вектори в MATLAB?
7. Які спеціальні вектори та матриці ви знаєте?
8. Які команди загального призначення ви знаєте?

Лабораторна робота № 2

ОПЕРАЦІЇ З МАТРИЦЯМИ ТА ВЕКТОРАМИ

Мета роботи: навчитися виконувати матричні та поелементні арифметичні операції над матрицями, транспонувати їх, знаходити матриці, зворотні до заданої, розраховувати визначник та слід матриць; виконувати маніпуляції над матрицями; та базові функції математичного аналізу.

2.1 Арифметичні операції

У системі *MATLAB* реалізовано два типи арифметичних операцій: поелементні та матричні. Інформацію про ці операції можна отримати виконанням команди

```
>> help ops
  Operators and special characters.
   Arithmetic operators.
    plus        - Plus                          +
    uplus       - Unary plus                    +
    minus       - Minus                         -
    uminus      - Unary minus                    -
    mtimes      - Matrix multiply               *
    times       - Array multiply                .*
    mpower       - Matrix power                 ^
    power       - Array power                   .^
    mldivide    - Backslash or left matrix divide  \
    mrdivide    - Slash or right matrix divide     /
    ldivide     - Left array divide             .\
    rdivide     - Right array divide            ./
```

Матричні операції виконуються за правилами лінійної алгебри над масивами відповідних для цих операцій розмірів. Операндами ***поелементних операції*** можуть бути масиви однакових розмірів, скаляри, або скаляр і масив.

Операції «+» та «-» у матричній і поелементній формах не відрізняються одна від одної і позначаються однаково. Операції множення, правостороннього та лівостороннього ділення та підвищення у степінь у матричній формі позначаються як «*», «/», «\» та «^» відповідно, а у поелементній формі – як

«.*», «./», «.\» та «.^», тобто зображення таких операцій утворюються додаванням символу «.» перед основним символом операції.

Для пояснення сенсу поелементних операцій розглянемо результати операцій Z=X.*Y для двох матриць розміром $m \times n$ та D=X.*c для матриці D розміром $m \times n$ і скалярної змінної c. У першому випадку це буде матриця того ж розміру з елементами

$$Z_{i,j} = X_{i,j} \times Y_{i,j}\ ; \quad i=1, 2, \ldots, m; \quad j=1,2,\ldots,n, \tag{2.1}$$

а у другому –

$$D_{i,j} = X_{i,j} \times c\ ; \quad i=1, 2, \ldots, m; \quad j=1, 2, \ldots, n. \tag{2.2}$$

Для ***визначення матричного добутку двох матриць*** їх внутрішні розміри повинні бути однаковими, тобто кількість стовпців у першому множнику повинна дорівнювати кількості рядків у другому множнику. Отже, матричним добутком **C** матриці **A** розміром $m \times k$ на матрицю **B** розміром $k \times n$, який розраховується оператором C=A*B є матриця розміром $m \times n$, елементи якої вираховуються за формулою:

$$C_{ij} = A_{i1}B_{1j} + A_{i2}B_{2j} + \ldots + A_{ik}B_{kj} = \sum_{l=1}^{k} A_{il}B_{lj}, \quad i=1, 2, \ldots, m; \ j=1, 2, \ldots, n. \tag{2.3}$$

Для добутку матриць, у загальному випадку, несправедливий переставний (комутативний) закон, тобто **A*B≠ B*A**.

Окремим випадком добутку матриць є множення матриці **A**, розміром $m \times k$, на вектор-стовпець b, складений із k елементів (c=A*b), та вектор-рядка **a**, складеного із k елементів, на матрицю **B** розміром $k \times n$ (d=a*B).

У першому випадку результатом буде вектор-стовпець з елементами:

$$c_i = A_{i1}b_1 + A_{i2}b_2 + \ldots + A_{ik}b_k = \sum_{j=1}^{k} A_{ij}b_j\ , \quad i= 1, 2, \ldots, m, \tag{2.4}$$

а у другому випадку – вектор-рядок з елементами:

$$d_j = a_1B_{1j} + a_2B_{2j} + \ldots + a_kB_{kj} = \sum_{i=1}^{k} a_iB_{ij}, \quad j = 1, 2, \ldots, n. \tag{2.5}$$

У такий же спосіб при множенні вектору-рядка a на вектор-стовпець b з однаковою кількістю елементів k оператором g=a*b утворюється скалярна величина

$$g = a_1 b_1 + a_2 b_2 + \ldots + a_k b_k = \sum_{i=1}^{k} a_i b_i , \qquad (2.6)$$

а при множенні вектору-стовпця b на вектор-рядок a з однаковою кількістю елементів k оператором D=b*a утворюється квадратна матриця D розміром $k \times k$ з елементами

$$D_{ij} = a_i b_j , \quad i=1, 2, \ldots, k; \quad j=1, 2, \ldots, k. \qquad (2.7)$$

У відповідності до поняття про добуток матриць, у цілу додатну степінь k можна піднести тільки квадратну матрицю, тобто результатом операції A^k буде

$$\mathbf{A}^k = (\underbrace{(\mathbf{A}*\mathbf{A})*\mathbf{A})*\ldots*\mathbf{A}}_{k \text{ співмножників}}). \qquad (2.8)$$

Крім того, у *MATLAB* передбачені операції правостороннього та лівостороннього ділення матриць: **B/A** відповідає $\mathbf{B}*\mathbf{A}^{-1}$, а **A\B** – $\mathbf{A}^{-1}*\mathbf{B}$, де $\mathbf{A}^{-1}$ – обернена матриця.

2.2 Визначення розмірів матриць та векторів

Розмір векторів та матриць у *MATLAB* визначається такими функціями:

L = length(X) – довжина вектору X;

m = size(A,1) – кількість рядків у матриці A, тобто розмір матриці за першим індексом;

n = size(A,2) – кількість стовпців у матриці A, тобто розмір матриці за другим індексом;

[m,n] = size(A) – кількість рядків m та стовпців n у матриці A.

2.3 Маніпуляції над матрицями

Якщо у матриці **A** розміром $m \times n$ замінити рядки відповідними стовпцями, то одержимо матрицю $\mathbf{A}^T$ розміром $n \times m$, яка має назву транспонованої у відношенні до матриці **A**.

Отже,

$a_{i,j}{}^T = a_{j,i}$; (i=1,2,..., n; j=1,2,..., m). (2.9)

У *MATLAB* ця операція позначається символом «′».

Функції, що здійснюють деякі маніпуляції над матрицями знаходяться у папці elmat:

```
>> help elmat
  Elementary matrices and matrix manipulation.
Matrix manipulation.
   reshape  - Reshape array.
   diag     - Diagonal matrices and diagonals of matrix.
   tril     - Extract lower triangular part.
   triu     - Extract upper triangular part.
   fliplr   - Flip matrix in left/right direction.
   flipud   - Flip matrix in up/down direction.
   rot90    - Rotate matrix 90 degrees.
```

Операція **A(:)** створює з елементів матриці **A** розміром $m \times n$ вектор стовпець, складений із послідовно приєднаних один до одного стовпчиків вихідної матриці.

2.4 Операції матричного аналізу та лінійної алгебри

До найчастіше застосованих операцій матричного аналізу та лінійної алгебри відносяться:

- розрахунок визначників;
- визначення мінорів та алгебраїчних доповнень матриці;
- визначення союзних та приєднаних матриць;
- обертання матриць;
- розв‘язання систем лінійних рівнянь.

Визначник матриці або ***детермінант матриці*** – це одна із найважливіших характеристик квадратної матриці, що застосовується при розв‘язанні багатьох задач лінійної алгебри. Він розраховується з усіх перестановок елементів матриці розміром $n \times n$ як сума добутків із n елементів. Кожний доданок складається з добутку елементів у такий спосіб, щоб у ньому були присутні тільки по одному елементу з кожного рядка і з кожного стовпця. Він може позначатися як Δ, $\Delta(\mathbf{A})$, $\det(\mathbf{A})$, $|\mathbf{A}|$. У *MATLAB* цю операцію виконує функція det(A).

Мінором M_{ij} елементу a_{ij} визначника квадратної матриці n-го порядку називають визначник матриці $(n-1)$-го порядку, отриманої із вихідної матриці викреслюванням з неї елементів i-го рядка та j-го стовпчика.

У якості прикладу наведемо програму, що утворює квадратну матрицю розміром 5×5 із випадкових чисел та обчислює для неї мінор M_{32}:

```
a = rand(5)
ap32 = a;
ap32(3,:) = [ ];  ap32 (:,2) = [ ]
Ma32=det(ap32)
```

При виконанні цієї програми отримаємо

```
a =
   0.7577   0.7060   0.8235   0.4387   0.4898
   0.7431   0.0318   0.6948   0.3816   0.4456
   0.3922   0.2769   0.3171   0.7655   0.6463
   0.6555   0.0462   0.9502   0.7952   0.7094
   0.1712   0.0971   0.0344   0.1869   0.7547
ap32 =
   0.7577   0.8235   0.4387   0.4898
   0.7431   0.6948   0.3816   0.4456
   0.6555   0.9502   0.7952   0.7094
   0.1712   0.0344   0.1869   0.7547
Ma32=
  -0.0157
```

Алгебраїчне доповнення A_{ij} може відрізняється від мінору M_{ij} тільки знаком:

$$A_{ij} = (-1)^{i+j} M_{ij}. \tag{2.10}$$

Матриця, складена із алгебраїчних доповнень всіх елементів вихідної матриці, називається ***союзною матрицею***. Вона позначається як $\widetilde{\mathbf{A}}$. Матриця, транспонована до союзної, називається ***приєднаною матрицею***. Вона позначається як $\mathrm{Adj}(\mathbf{A})$ (***adjacency matrix***):

$$\mathrm{Adj}(\mathbf{A}) = \widetilde{\mathbf{A}}^T.$$

Оберненою матрицею $\mathbf{X}=\mathbf{A}^{-1}$ у відношенні до вихідної квадратної матриці **A** називається така квадратна матриця яка, будучи помноженою на вихідну, дає одиничну діагональну матрицю **E** того ж розміру:

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{E}, \tag{2.11}$$

або у розгорнутій формі:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \cdot \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{12} & x_{22} & \dots & x_{2n} \\ \dots & \dots & \dots & \dots \\ x_{n1} & x_{n2} & \dots & x_{nn} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{pmatrix} \tag{2.12}$$

Для матриць невеликого розміру обернення часто виконують вручну, використовуючи формулу:

$$\mathbf{A}^{-1} = \frac{\widetilde{\mathbf{A}}^T}{\Delta} = \frac{\mathrm{Adj}(\mathbf{A})}{\det(\mathbf{A})}. \tag{2.13}$$

При $n>3$ розрахунки вручну за формулою (2.12) стають дуже громіздкими.

Як видно з (2.11), елементи k-го стовпця оберненої матриці **X** можна визначити розв'язуючи систему n лінійних рівнянь з n невідомими.

$$\begin{pmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ ... & ... & ... & ... \\ a_{n1} & a_{n2} & ... & a_{nn} \end{pmatrix} \cdot \begin{pmatrix} x_{1,k} \\ x_{2,k} \\ ... \\ x_{n,k} \end{pmatrix} = \begin{pmatrix} e_{1,k} \\ e_{2,k} \\ ... \\ e_{n,k} \end{pmatrix}, \tag{2.14}$$

де $e_{i,k} = \begin{cases} 0, & при \ \ i \neq k, \\ 1, & при \ \ i = k, \end{cases} \quad k = 1, 2, ..., n.$

Отже, для визначення усіх елементів оберненої матриці необхідно розв'язати n систем рівнянь. Цей підхід часто використовують при машинних розрахунках. Розв'язувати системи рівнянь можна будь-яким з відомих методів, наприклад, методом Гауса.

У *MATLAB* для обернення матриць використовують функцію inv(A).

Слідом квадратної матриці **A** називають суму її діагональних елементів:

$$tr(A) = \sum_{i=1}^{n} a_{ii} \ . \tag{2.15}$$

У *MATLAB* цю операцію виконує функція trace(A).

Ранг матриці **A** визначає кількість лінійно незалежних рядків у вихідній системі. У ML цю операцію виконує функція rank (A, tol).

Система n лінійних рівнянь з n невідомими має вигляд:

$$\begin{cases} a_{11} \cdot x_1 + a_{12} \cdot x_2 + ... + a_{1n} \cdot x_n = b_1, \\ a_{21} \cdot x_1 + a_{22} \cdot x_2 + ... + a_{2n} \cdot x_n = b_2, \\ \qquad ..... \\ a_{n1} \cdot x_1 + a_{n2} \cdot x_2 + ... + a_{nn} \cdot x_n = b_n. \end{cases} \tag{2.16}$$

Її можна записати і у матричній формі:

A*X=B, (2.17)

де $\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ ..... & ..... & ..... & ..... \\ a_{n1} & a_{n2} & ... & a_{nn} \end{pmatrix}$ – квадратна матриця коефіцієнтів;

$\mathbf{B} = \begin{pmatrix} b_1 \\ b_2 \\ ... \\ b_n \end{pmatrix}$ – вектор вільних членів; $\mathbf{X} = \begin{pmatrix} x_1 \\ x_2 \\ ... \\ x_n \end{pmatrix}$ – шуканий вектор коренів.

Способи розв‘язання систем лінійних рівнянь поділяються на дві групи:

- точні методи (метод обернення матриці коефіцієнтів, правило Крамера, метод Гауса та інші);
- ітераційні методи (Ньютона, Зейделя, простих ітерацій та інші).

Якщо матриця **A** неособлива, тобто її визначник не дорівнює нулю, то система має єдине рішення:

$$\mathbf{X}=\mathbf{A}^{-1}\mathbf{B}, \tag{2.18}$$

де $\mathbf{A}^{-1}$ – матриця, обернена до матриці **A**.

Обчислення коренів за формулою (2.18) називається методом обернення матриці коефіцієнтів.

Згідно з правилом Крамера корні обчислюються за формулами:

$$x_1 = \frac{\Delta_1}{\Delta}, x_2 = \frac{\Delta_2}{\Delta}, ..., x_n = \frac{\Delta_n}{\Delta}, \tag{2.19}$$

де Δ – визначник матриці $\mathbf{A}$;

Δ_i – визначники матриць, які отримані з матриці $\mathbf{A}$ шляхом заміни її *i*-го стовпця вектором вільних членів $\mathbf{B}$.

Рішення систем лінійних рівнянь у MATLAB реалізується операцією лівостороннього матричного ділення `X=A\B`**,** у яку закладено декілька алгоритмів розв‘язання, один з яких обирається автоматично в залежності від структури матриці $\mathbf{A}$**,** зокрема від ступеня її розрідженості.

Наведена операція лівостороннього ділення може розв‘язати приблизно (методом найменших квадратів) навіть недовизначену та перевизначену системи лінійних рівнянь, тобто системи, в яких кількість рівнянь є меншою або більшою ніж кількість невідомих змінних, наприклад,

```
>> A=[1 2 3; 7 5 4]
A =
    1    2    3
    7    5    4
>> B=[5;1]
B =
    5
    1
>> x=A\B
x =
  -1.0000
        0
   2.0000
>>F=A*x-B
ans =
  1.0e-15 *
        0
         0.8882
```

2.5 Базові функції математичного аналізу над векторами та матрицями

Базові функції математичного аналізу знаходяться серед інструментів папки datafun:

```
>> help datafun
   Data analysis and Fourier transforms.
   Basic operations.
   max        - Largest component (найбільший елемент).
   min        - Smallest component (найменший елемент).
   mean       - Average or mean value (середнє арифметичне).
   median     - Median value (серединне значення).
   std        - Standard deviation (середньоквадратичне відхилення).
   var        - Variance (дисперсія).
   sort       - Sort in ascending order (сортування за збільшенням).
   sortrows   - Sort rows in ascending order (сортування рядків за збільшенням).
   sum        - Sum of elements (сума елементів).
   cumsum     - Cumulative sum of elements (накопичення суми).
   prod       - Product of elements (добуток елементів).
   cumprod    - Cumulative product of elements (накопичення добутку).
   trapz      - Trapezoidal numerical integration (чисельне інтегрування методом трапецій).
```

cumtrapz - Cumulative trapezoidal numerical integration (накопичення визначеного інтегралу).

Finite differences.

diff - Difference and approximate derivative (прямі різниці або апроксимовані похідні).

Особливістю наведених вище функцій, як і багатьох інших, полягає у тому, що для матриць вони виконуються для кожного стовпчика, а не для матриці у цілому. Щоб виконати відповідну операцію для кожного рядка, треба транспонувати вихідну матрицю і отриманий результат, наприклад, Amean_rows=(mean(A'))'. Щоб виконати бажану операцію над матрицею у цілому, треба звернутися до цієї функції двічі, наприклад, S=sum(sum(A)).

Функції max та min можуть працювати як з одним аргументом, так і з двома. У другому випадку обидва аргументи повинні мати однаковий розмір, тому що відповідна операція виконується поелементно.

Серединним значенням при непарній кількості елементів є значення середнього елемента упорядкованого вектору, а при парній кількості – середнє арифметичне двох середніх елементів.

Середньоквадратичне значення (Standard deviation) вектору $s(x)$ обчислюється функцією std (x, flag) за формулою

$$s(x)=\begin{cases}\sqrt{\dfrac{1}{n-1}\cdot\sum_{i=1}^{n}(x_i-x_{mean})^2} & \text{при} \quad flag=0,\\ \sqrt{\dfrac{1}{n}\cdot\sum_{i=1}^{n}(x_i-x_{mean})^2} & \text{при} \quad flag=1.\end{cases} \tag{2.20}$$

За замовчанням flag=0.

Дисперсія (variance) вектору розраховується функцією var (x, flag) як квадрат середньоквадратичного відхилення:

$$D(x)=s^2(x). \tag{2.21}$$

Визначені інтеграли $z(x)=\int_{i=1}^{n} x(i)di$ та $z(x)=\int_{i=1}^{k=1::n} x(i)di$ розраховуються методом трапецій функціями trapz (X) та cumtrapz (X) відповідно.

Першими прямими різницями одномірного масиву

$$\mathbf{V}=\begin{bmatrix} v_1 & v_2 & v_3 & \dots & v_{n-2} & v_{n-1} & v_n \end{bmatrix}$$

називають вектор

$$\Delta\mathbf{V}=\begin{bmatrix} \Delta v_1 & \Delta v_2 & \dots & \Delta v_{n-2} & \Delta v_{n-1} \end{bmatrix}=\begin{bmatrix} v_2-v_1 & v_3-v_2 & \dots & v_{n-1}-v_{n-2} & v_n-v_{n-1} \end{bmatrix}$$

Другі прямі різниці – це прямі різниці від перших прямих різниць, тобто

$$\Delta^2\mathbf{V}=\begin{bmatrix} \Delta^2 v_1 & \dots & \Delta^2 v_{n-2} \end{bmatrix}=\begin{bmatrix} \Delta v_2-\Delta v_1 & \dots & \Delta v_{n-1}-\Delta v_{n-2} \end{bmatrix}=\begin{bmatrix} v_3-2v_2+v_1 & \dots & v_n-2v_{n-1}+v_{n-2} \end{bmatrix}$$

2.6 Завдання

1. Виконати операції над матрицями у відповідності з виразами, наведеними у другому стовпчику таблиці 2.1. Вихідні данні:

$$\mathbf{A}=\begin{bmatrix} 2 & 3{,}1 \\ 4{,}5 & 10{,}7 \\ 7 & 1 \end{bmatrix};\ \mathbf{B}=\begin{bmatrix} 7{,}5 & 11 & 1{,}7 \\ 5 & 4 & 2 \end{bmatrix};\ \mathbf{C}=\begin{bmatrix} 1 & 0 \\ 7 & 8 \\ 5 & 6 \end{bmatrix};\ \mathbf{D}=\begin{bmatrix} 7{,}4 & 5 \\ 9 & 8 \end{bmatrix}.$$

2. Для матриці **К** виконати дії з колонки 3 табл. 2.1
3. Матрицю **К** доповнити довільно до квадратної матриці **Н** розміром 5×5. Для отриманої у такий спосіб матриці обчислити визначник, слід та обернену матрицю. Для оберненої матриці перевірити умову (2.12) та знайти її останній стовпець за формулою (2.14).
4. Для елемента h_{ij} матриці **Н** із заданими у табл. 2.2 значеннями індексів визначити мінор M_{ij} та алгебраїчне доповнення A_{ij}.
5. Отримати вектор **V** шляхом витягування матриці **Н** по рядкам у 1 рядок та упорядковати його за збільшенням елементів (вектор **Vu**). Знайти для вектору **V** перші, другі та треті прямі різниці, $\int_{i=1}^{25} V(i)di$, а також середнє, серединне і середньо-квадратичне значення;
6. Розв'язати систему лінійних рівнянь, наданих у табл. 2.3.

Таблиця 2.1

Вар.	**ЗАВДАННЯ**	
1	2	3
1	K= A*D+B^T-C	Знайти суми елементів кожного рядка
2	K= $(A+C)^T$.*B	Знайти добутки елементів кожного стовпчика
3	K= (A-C)*(D*B)	Знайти середнє арифметичне першого рядка
4	K= (B^T-A-C)*D	Знайти максимальний елемент останнього стовпчика
5	K= $(C*D-A)^T$+B	Знайти мінімальний елемент матриці та його позицію
6	K= (C-A)*D^T*B	Упорядкувати стовпці за зменшенням
7	K= (A.*C)*B	Упорядкувати рядки за збільшенням
8	K= A*(D*C^T-B)	Перегорнути матрицю у горизонтальній площині
9	K= C*D-A-B^T	Перегрупувати матрицю 3×2 у матрицю 2×3
10	K= A*D^2+C	Перегорнути матрицю у вертикальній площині
11	K= (A+C)*B	Обернути матрицю на 180°
12	K= $(A*B)^2$	Знайти суму максимальних елементів рядків
13	K= D*C^T+B	Знайти добуток мінімальних елементів стовпчиків
14	K= A*D+B^T-C	Знайти кумулятивні суми рядків
15	K= A*(D*C^T-B)	Знайти кумулятивні добутки стовпчиків

Таблиця 2.2

Вар.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i	1	3	2	3	5	2	4	1	2	5	3	4	1	2	3
j	3	2	5	4	3	1	5	5	4	1	3	1	2	2	5

2.7 Методичні вказівки та рекомендації

1. Середнє геометричне значення вектору розраховуйте за формулою

$$G(x)=\sqrt[n]{\prod_{i=1}^{n} x_i} \qquad (2.22)$$

2. Для виконання пункту 5 завдання використайте функцію diff.
3. Для перевірки точності розв‘язання системи лінійних рівнянь розрахуйте вектор нев‘язок F=A*X-B.

Таблиця 2.3

Вар.	Система рівнянь	Вар.	Система рівнянь
1	$3,14x_1-2,22x_2+1,17x_3=1,27$ $-2,12x_1+1,32x_2-2,45x_3=2,13$ $1,17x_1-2,45x_2+1,18x_3=3,14$	9	$0,78x_1+1,08x_2-1,35x_3=0,57$ $1,08x_1-1,28x_2+0,37x_3=1,27$ $-1,35x_1+0,37x_2+2,86x_3=0,47$
2	$2,45x_1+1,75x_2-3,24x_3=1,23$ $1,75x_1-1,16x_2+2,18x_3=3,43$ $-3,24x_1+2,18x_2-1,85x_3=-0,16$	10	$0,83x_1+2,18x_2-1,73x_3=0,28$ $2,18x_1-1,41x_2+1,03x_3=-1,18$ $-1,73x_1+1,03x_2+2,27x_3=0,72$
Продовження таблиці 2.3			
3	$1,65x_1-2,27x_2+0,18x_3=2,25$ $-2,27x_1+1,73x_2-0,46x_3=0,93$ $0,18x_1-0,46x_2+2,16x_3=1,33$	11	$2,74x_1-1,18x_2+1,23x_3=0,16$ $-1,18x_1+1,71x_2-0,52x_3=1,81$ $3,14x_1-2,22x_2+1,17x_3=1,27$
4	$3,23x_1+1,62x_2+0,65x_3=1,28$ $1,62x_1-2,33x_2-1,43x_3=0,87$ $0,65x_1-1,43x_2+2,18x_3=-2,87$	12	$1,35x_1-0,72x_2+1,81x_3=0,88$ $-0,72x_1+1,45x_2-2,18x_3=1,72$ $1,38x_1-2,18x_2+0,93x_3=-0,72$
5	$0,93x_1+1,42x_2-2,55x_3=2,48$ $1,42x_1-2,87x_2+2,36x_3=-0,75$ $-2,55x_1+2,36x_2-1,44x_3=1,83$	13	$3,14x_1-2,13x_2+1,17x_3=2,54$ $1,75x_1-1,45x_2+2,67x_3=0,65$ $-0,78x_1+2,04x_2+1,52x_3=1,33$
6	$1,42x_1-2,15x_2+1,07x_3=2,48$ $-2,15x_1+0,76x_2-2,18x_3=1,15$ $1,07x_1-2,18x_2+1,23x_3=0,88$	14	$0,83x_1+2,18x_2-1,73x_3=0,28$ $-1,18x_1+1,71x_2-0,52x_3=1,81$ $2,18x_1-1,41x_2+1,03x_3=-1,18$
7	$2,23x_1-0,71x_2+0,63x_3=1,28$ $-0,71x_1+1,45x_2-1,34x_3=0,64$ $0,63x_1-1,34x_2+0,77x_3=-0,87$	15	$0,63x_1-1,34x_2+0,77x_3=-0,87$ $1,42x_1-2,87x_2+2,36x_3=-0,75$ $1,65x_1+1,27x_2-0,84x_3=1,51$
8	$1,63x_1+1,27x_2-0,84x_3=1,51$ $1,27x_1+0,65x_2+1,27x_3=-0,63$ $-0,84x_1+1,27x_2-1,21x_3=2,15$	16	$5,62x_1+1,37x_2-2,74x_3=3,35$ $0,42x_1-2,89x_2+1,33x_3=-0,84$ $1,86x_1+3,25x_2-0,28x_3=2,68$

2.8 Контрольні питання та завдання

1. Поясніть різницю між операціями * та .*, ^ та .^, / та \. Наведіть приклади.
2. Перелічите основні операції матричної алгебри. Як вони виконуються в *MATLAB*? Як перевірити результати виконання цих операцій?
3. Як програмно визначити розміри матриць та векторів?

4. Надайте визначення оберненої матриці. Які методи її розрахунку ви знаєте?
5. Що таке мінор, алгебраїчне доповнення, союзна матриця?
6. Які маніпуляції над матрицями та векторами можна здійснити в програмному середовищі *MATLAB*?
7. Перелічите базові функції математичного аналізу. Якими є особливості виконання цих функцій для матриць в середовищі *MATLAB*?

Лабораторна робота № 3

ПОБУДОВА ТА ОФОРМЛЕННЯ БАЗОВИХ ДВОВИМІРНИХ ГРАФІКІВ ФУНКЦІЙ

Мета роботи – навчитися будувати двовимірні графіки заданих функцій, оформлювати підписи графіків та осей, виводити графіки у різні графічні вікна.

3.1 Загальні відомості

Графічні засоби пакета *MATLAB* орієнтовані не на створення графічних примітивів (прямокутників, кіл, тощо), а на побудову дво- и тривимірних графіків функціональних залежностей у самому різноманітному вигляді. Графіки будуються в окремих вікнах, що мають назву *Figure*.

Початкову інформацію про основні функції двовимірної графіки можна отримати виконанням команди *help graph2d*

```
>> help graph2d
Two dimensional graphs.
Elementary X-Y graphs.
   plot         - Linear plot.
   loglog       - Log-log scale plot.
   semilogx  - Semi-log scale plot.
   semilogy  - Semi-log scale plot.
   polar        - Polar coordinate plot.
   plotyy       - Graphs with y tick labels on the left and right.
Axis control.
   axis      - Control axis scaling and appearance.
   grid      - Grid lines.
   hold      - Hold current graph.
   subplot  - Create axes in tiled positions.
Graph annotation.
   title       - Graph title.
   xlabel    - X-axis label.
   ylabel    - Y-axis label.
   text       - Text annotation.
   gtext      - Place text with mouse.
Hardcopy and printing.
   print       - Print graph or Simulink system; or save graph to MATLAB file.
   printopt   - Printer defaults.
   orient     - Set paper orientation.
```

3.2 Побудова двовимірних графіків в Декартових координатах

Графіки, побудовані за допомогою функцій *plot*, *semilogx*, *semilogy* та *loglog* відрізнятимуться один від одного тільки масштабуванням осей (лінійне, напівлогарифмічне або логарифмічне). Звернення до цих функцій має однаковий формат. Тому розглянемо способи їх застосування на прикладі функції *plot*.

Домовимося, що в робочому просторі (*Workspace*) пакета *MATLAB* існують значення таких перемінних: x, y, – вектори-рядки із n елементів; u – вектор-стовпець із m елементів; A, B, C – матриці розміром m×n; z – комплексне число, Z – вектор комплексних чисел. Тоді

plot (y) – будує графік функції $y(i) = f(i)$ по точках з абсцисами $i = [1, 2, \ldots, n]$ та ординатами $[y(1), y(2), \ldots, y(n)]$;

plot (x,y) – будує графік функції $y(i) = f\big(x(i)\big)$;

plot (Z) – те ж саме, що і plot(real(Z),imag(Z));

plot (A) – будує в одній системі координат n графіків, кожний з яких уявляє собою залежність елементів одного стовпчика матриці A від порядкового номеру рядка;

plot (u,A) – те ж саме, але у функції елементів вектору-стовпця u.

plot(x,A) – будує в одній системі координат m графіків, кожний з яких уявляє собою залежність елементів одного рядка матриці A від елементів вектору-рядка x.

Якщо треба побудувати графік функції, заданої аналітично, у лінійному масштабі, то вектор аргументів зручно задавати оператором

x = linspace (x0, xk)

Приклад 3.1. Побудувати в одній системі координат графіки функцій $y = \sin 2x \cdot \cos x$ та $z = e^{1/(x+1)}$ на інтервалі $x \in [0,10]$.

```
x =linspace(0,10);
y = sin (2*x) .* cos (x);
plot (x,y)
z = exp (1./(x+1)); A = [y; z]; plot (x, A)
```

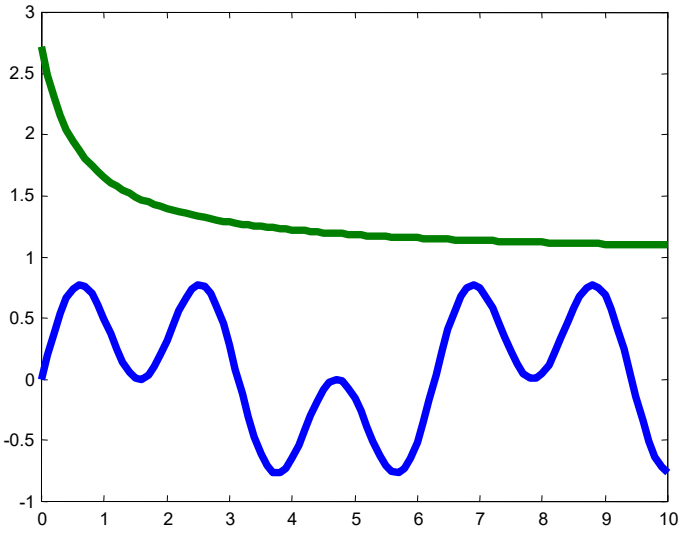


Рис. 3.2. Графіки функцій до прикладу 3.1

Графік, отриманий при виконанні цієї програми подано на рис. 3.1

Цей графік можна імпортувати у *Microsoft Word* або у будь-який графічний редактор, наприклад, у *Microsoft Office Visio*, скориставшись через меню графічного вікна командою *Edit→Copy Figure*. У графічному редакторі можна скоріше і якісніше, ніж у середовищі пакета *MATLAB* доповнити графік допоміжними підписами та відкоригувати зображення.

Одним графічним оператором можна вивести в одній системі координат декілька графіків у такий спосіб:

plot (x1,y1, x2,y2, …, xN,yN)

3.3 Керування стилями та кольорами зображення ліній та маркерів на графіках

Кожну пару векторів, що застосовуються для виводу одного з графіків, можна доповнити ***строковим параметром*** Style, який може мати у своєму складі ***від одного до чотирьох символів***, що визначатимуть колір та способи зображення маркера вузлових точок таблиці та лінії, що поєднує сусідні точки:

plot(x1,y1,Style1,x2,y2,Style2,…,xN,yN,StyleN)

Можливі значення символів у параметрі Style та їх сенс відображено у табл. 3.1. Отже, параметр Style може виглядати так: 'r', 'm:', 'o', 'k-.x', тобто він може визначати одну, дві або три властивості графіка із табл. 3.1.

Кольори, наведені у табл. 3.1 отримані змішуванням трьох основних кольорів (червоного, зеленого та синього) так званої **rgb**-палітри однакової

(максимальної) інтенсивності. У пакеті *MATLAB* кольори можна задавати вектором-рядком із 3-х чисел в діапазоні від 0 до 1, які визначають інтенсивність змішуваних основних кольорів: [red green blue].

Таблиця 3.1

Колір лінії	**Тип маркера**	**Тип лінії**
b – ***b****lue* (синій)	**.** – крапка	**-** – суцільна (*solid*)
g – ***g****reen* (зелений)	**o** – коло	**:** – дрібний (точковий) пунктир (*dotted*)
r – ***r****ed* (червоний)	**x** – хрестик	**--** – штрихова (*dashed*)
y – ***y****ellow* (жовтий)	**+** – плюс	**-.** – штрих-пунктирна (*dashdotted*)
m – ***m****agenta* (пурпурний)	***** – зірка	
c – ***c****yan* (блакитний)	**s** – квадрат (***s****quare*)	
w – ***w****heit* (білий)	**d** – ромб (***d****iamond*)	
k – *blac****k*** (чорний)	**v** – трикутник вершиною вниз	
	^ – трикутник вершиною угору	
	< – трикутник вершиною вліво	
	> – трикутник вершиною вправо	
	p – п‘ятикутна зірка (***p****entagram*)	
	h – шестикутна зірка (***h****exagram*)	

Кольорам з табл. 3.1 відповідають такі значення **rgb**-векторів:

[0 0 1] – blue [1 0 1] – magenta

[0 1 0] – green [1 1 0] – yellow

[1 0 0] – red [0 0 0] – black

[0 1 1] – cyan [1 1 1] – white

1 означає максимальну інтенсивність, а 0 – нульову.

Треба відзначити, що кольори *cyan*, *magenta*, та *yellow* виявляються занадто яскравими та погано видними на білому фоні, особливо при чорно-

білому друку. Тому за замовчанням в *MATLAB* при зображенні декількох графіків в одній системі координат, що виводяться одним оператором plot застосовують інші кольори, а саме:

[0 0 1] – blue
[0 0.5 0] – dark green
[1 0 0] – red
[0 0.75 0.75] – dark cyan
[0.75 0 0.75] – violet
[0.75 0.75 0] – light brown
[0.25 0.25 0.25] – dark grey

На рис. 3.2 показані графіки, які демонструють кольори, відображені у табл. 3.1 (а), послідовність кольорів, прийнята у MATLAB за замовчанням (б), стилі зображення маркерів точок графіків (в) та стилі зображення ліній на графіках (г).

Окремі властивості (*Properties*) графіків можна змінювати, застосовуючи засоби так званої дескрипторної (низькорівневої) графіки, застосовуючи функцію `plot` у такому форматі:

plot (x1,y1, PropName1,PropValue1, PropName2,PropValue2, …)

Інформацію про імена властивостей (*PropertyName*) та їх значення за замовчанням (*PropertyValue*) можна побачити після виконання функції get(gca), де gca означає ***G****raphic* ***C****urrent* ***A****xes*. Наприклад, товщина лінії ('LineWidth') за замовченням має значення 0.5, а палітра чергування кольорів ('ColorOrder') є матрицею розміром 7×3, складеною з векторів рядків, наведених вище. Цих властивостей дуже багато. Тому не треба намагатися змінювати властивості, сенс яких є не зрозумілим.

Змінити колір і товщину лінії можна так:

plot(x1,y1, 'LineWidth',2, 'Color',[0.8 0.8 0])

або

plot(x1,y1), set (gca, 'LineWidth',2, 'Color',[0.8 0.8 0])

Теж саме можна виконати за допомогою функцій меню графічного вікна, що буде предметом вивчення у наступній лабораторній роботі.

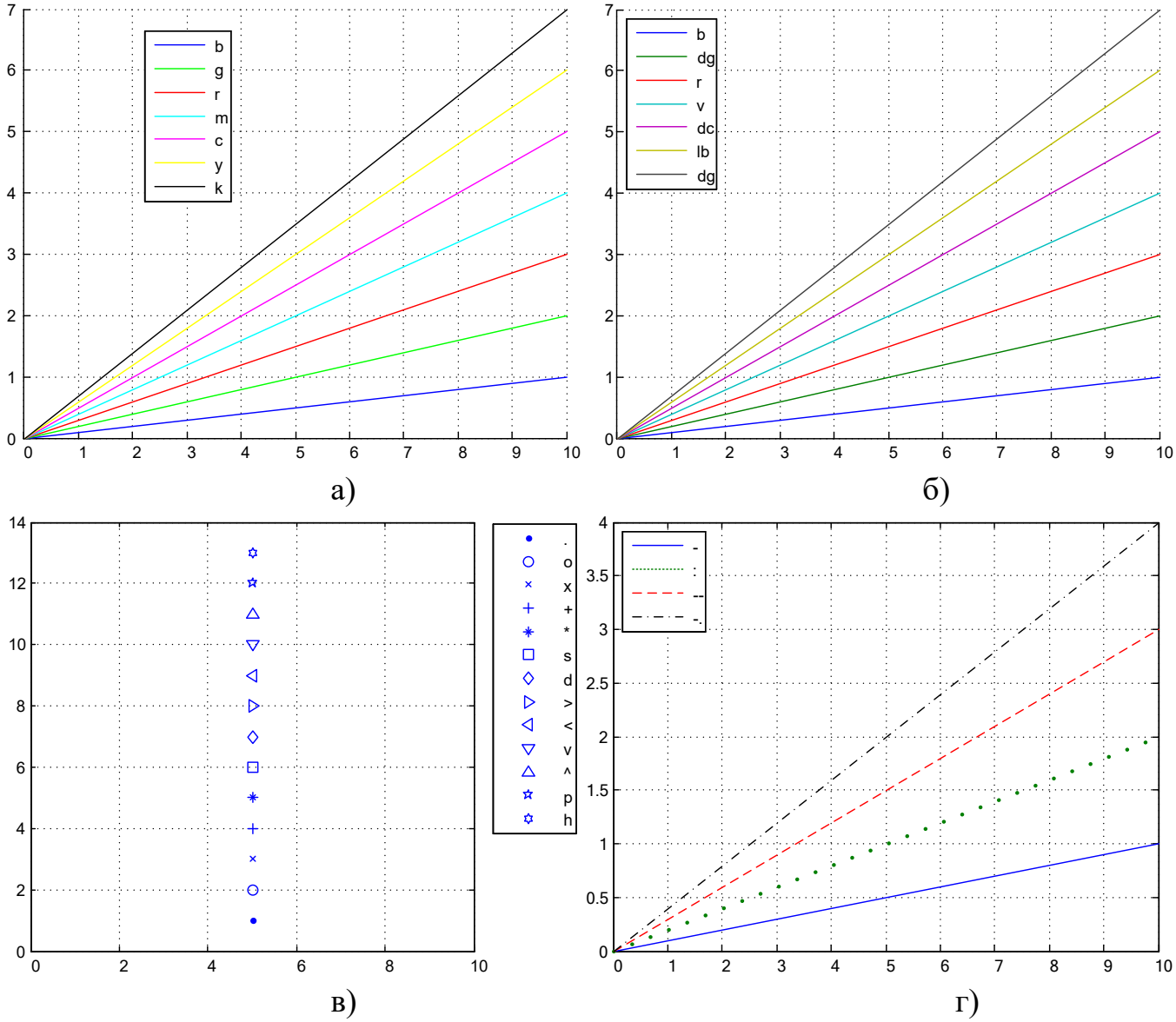


Рисунок 3.2. Кольори та стилі зображення графіків

3.4 Нанесення на графік заголовків, позначень осей, текстових пояснень та коментарів

Для ***нанесення заголовку*** в пакеті *MatLab* призначена функція title, для ***позначення осей*** – функції xlabel та ylabel, а для ***нанесення інших текстів*** (коментарів, позначень графіків, тощо) – функції text та gtext.

Перші три функції виводять тексти у наперед заданих позиціях: заголовок зверху, а позначення осей уздовж осей. Тому у найпростішому варіанті вони можуть застосовувати тільки один вхідний аргумент – 'Текст', наприклад,

title ('Вольт-амперна характеристика діода'),

xlabel ('Струм,А'), ylabel('Напруга,В')

Для функції text треба ще задати координати виводу (у системі координат, що застосовується при побудові графіка): text (x,y,'текст'), а функція gtext ('текст') позиціонує текст за допомогою миші. підпис осі абсцис.

За замовчанням усі тексти виводяться стилем Helvetica нормальної жирності (normal), розмір шрифту 10 пт. За бажанням ці параметри можна змінити, додавши після параметру 'Текст' конструкцію 'FontName', 'НазваШрифту', 'FontSize', РозмірШрифту, 'FontWeight', 'ЖирністьШрифту, наприклад,

Xlabel ('Струм,А', 'FontName',’Arial', 'FontSize',12, 'FontWeight',’bold’)

Розташування заголовка та позначень осей продемонстровано на рис 3.3.

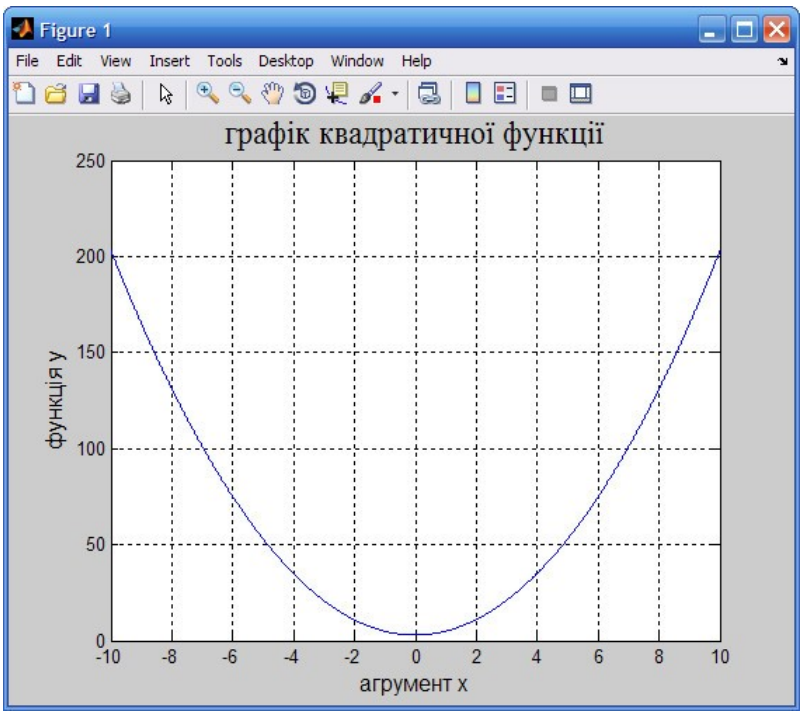


Рис. 3.3. Приклад оформлення підписів графіка

Цей графік отримано в результаті виконання наступної програми:

```
x=linspace(-10,10);
y=2*x.^2+3;
plot(x,y)
grid on
xlabel('агрумент x','FontName','Arial','FontSize', 12)
ylabel('функція y','FontName','Arial','FontSize', 12)
title('графік квадратичної функції', 'FontName','Times New Roman','FontSize',16)
```

При наявності в одній системі координат декількох графіків, зображених різними кольорами або стилями, доцільно додавати до них табличку з поясненнями, яка створюється функцією

legend ('text1', 'text2', …, 'textN', Pos)

Кількість текстових параметрів, що отримують коментарі, пояснення або позначення графіків, дорівнює кількості графіків. Останній параметр визначає позицію розташування легенди і може приймати такі значення: 1, 2, 3, 3 – номер квадранта, -1 – найбільш зручне місце у полі графіків, 0 – поза полем графіків.

На рис. 3.2 параметр Pos у функції legend має для графіка а) значення 0, для графіка в)значення -1, а для графіків б) і г) значення 2. Якщо визначена позиція виведення легенди виявиться невдалою, її можна змінити в інтерактивному режимі, перетягнувши легенду в інше місце за допомогою миші.

Мишка застосовується при зверненні ще до однієї графічної функції, яка визначає чисельні значення координат виділених за допомогою миші точок графіка:

[x,y]=ginput(n)

де n – кількість точок.

3.5 Керування системою координат

Нанесенням на графік координатної сітки керує команда grid у таких модифікаціях: grid on – увімкнути режим зображення сітки, grid off – вимкнути цей режим, grid – змінити стан команди.

За аналогією команда hold керує ***режимом затримки графіка***, побудованого попереднім оператором plot, при виконанні наступного оператора plot, тобто hold on вмикає цей режим, hold off вимикає його, а hold змінює поточний стан команди. У стані hold off, який застосовується за замовчанням, у результаті виконання операцій

```
plot (x1,y1),  plot (x2,y2)
```

відкривається графічне вікно, у нього виводиться перший графік, який миттєво замінюється другим графіком.

Щоб побачити обидва графіки послідовно в одному вікні, треба між функціями plot поставити оператор pause, який затримує виконання наступної операції до натискання будь-якої клавіші.

Вивести обидва графіки разом в одне вікно можна сукупністю операторів

```
plot (x1,y1), hold on, plot (x2,y2)
```

або

```
plot (x1,y1, x2,y2)
```

Якщо два графіки в одному вікні мають суттєво різні масштаби, то для їх зображення зручно застосувати функцію

```
plotyy (x1,y1, x2,y2)
```

яка виводить дві осі ординат (зліва для першого графіка і зліва для другого, наприклад, графік, зображений на рис. 3.4, є результатом виконання програми

```
x=linspace(-5,5);
y1=x.^2-12*x;  y2=100*x.^2-2*x-3;
plotyy(x,y1,x,y2)
```

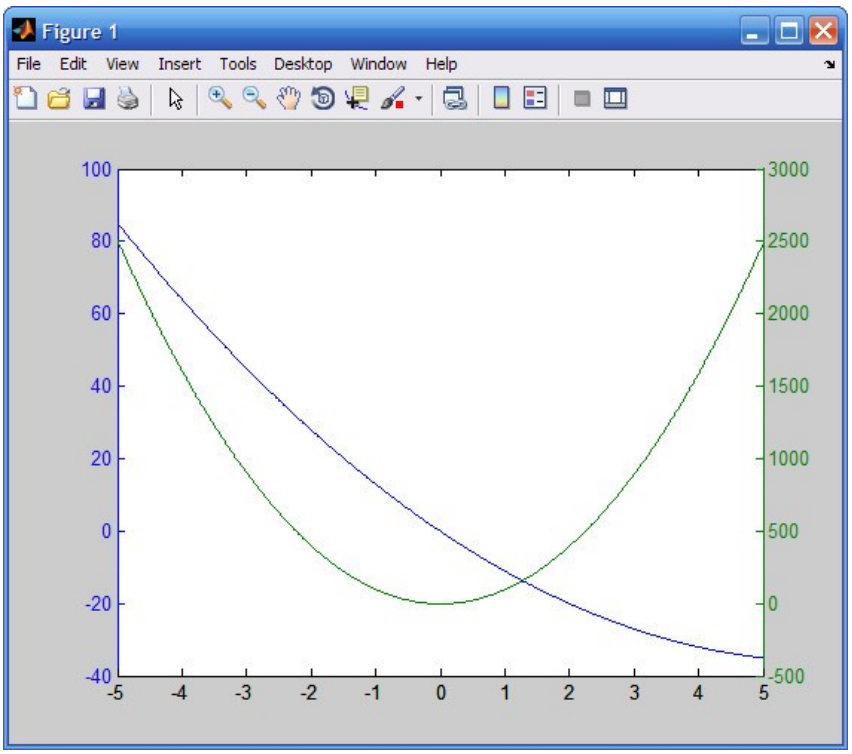


Рис. 3.4. Приклад суміщення двох графіків з різними масштабами

Функції

```
xlim ( [x_left  x_reight] ),
ylim ( [y_bottom  y_top] )
```

та

```
axis ( [x_left  x_reight  y_bottom  y_top] )
```

встановлюють ***межі системи координат***.

3.6 Керування графічними вікнами

Для керування графічними вікнами використовують такі команди:

`figure` – відкриває нове графічне вікно з черговим номером;

`figure (n)` – відкриває або активізує графічне вікно з заданим номером;

`close` – закриває активне графічне вікно;

`close (n)` – закриває графічне вікно з заданим номером;

`close all` – закриває усі графічні вікна;

`clf` – видаляє зображення з активного графічного вікна;

`clf (n)` – видаляє зображення із заданого графічного вікна;

`shg` – показує активне графічне вікно на передньому плані.

Кожне графічне вікно може бути розбите на декілька підвікон. Для цього використовується команда

subplot (m,n,p)

яка створює «матрицю» підвікон розміром m×n, здійснює їх наскрізну нумерацію за рядками і активізує підвікно з номером p ($1 \le p \le m \cdot n$), у якому будуть виконуватися дії, передбачені наступним графічним оператором.

Для переходу у інше графічне підвікно треба знову виконати оператор subplot, змінюючи у ньому тільки значення останнього параметру, тобто номер активного підвікна, наприклад.

subplot (1,2,1), plot(x1,y1), subplot (1,2,2), plot(x2,y2)

Підвікна, розташовані поруч, можна поєднувати в єдине підвікно, наприклад,

subplot (2,3,[1,2]),…, subplot (2,3,[4,5]),…, subplot (2,3,[3;6]),…

Наведемо приклад побудови графіків у підвікнах

```
figure                  %відкриття загального графічного вікна
subplot(2,3,1)    %відкриття першого підвікна
x=linspace(-10,10);     %завдання аргументу
y=2*x.^2-4*x-3;   %обчислення першої функції
plot(x,y)         %побудова першої функції у першому підвікні
grid on                 %активація сітки
subplot(2,3,2)    %відкриття другого підвікна
y=-5*x.^2+8*x-3; %обчислення другої функції
plot(x,y), grid   %побудова другої функції у другому підвікні
subplot(2,3,3);   % відкриття третього підвікна
y=x.^2+3*x-1;           % обчислення третьої функції
plot(x,y),grid    %побудова третьої функції у третьому підвікні
subplot(2,3,4)    % відкриття четвертого підвікна
y=3*x+10;         % обчислення четвертої функції
plot(x,y), grid   %побудова четвертої функції у четвертому підвікні
subplot(2,3,5)    % відкриття п’ятого підвікна
y=sin(x);         % обчислення п’ятої функції
plot(x,y),grid    %побудова п’ятої функції у п’ятому підвікні
subplot(2,3,6)    % відкриття шостого підвікна
y=cos(x);         % обчислення шостої функції
plot(x,y),grid    %побудова шостої функції у шостому підвікні
```

Результат виконання програми показаний на рис. 3.5.

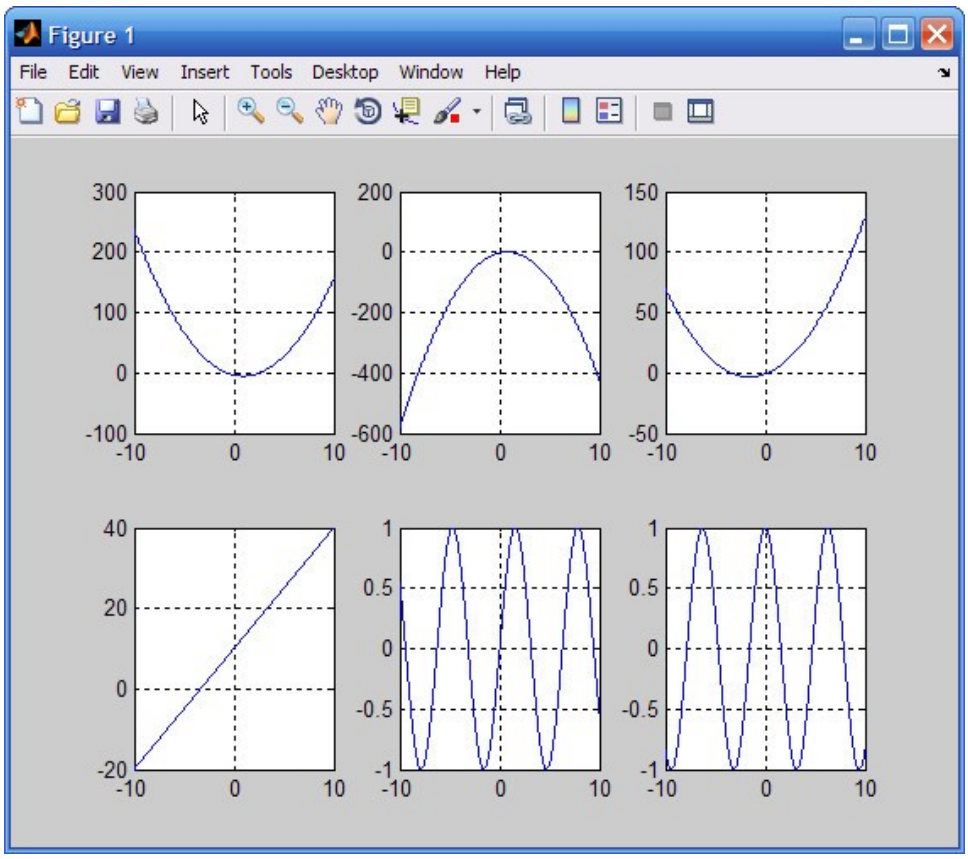


Рис. 3.5. Приклад побудови графіків у графічних підвікнах

У кожному з графічних підвікон можна робити підписи осей, графіків, змінювати типи ліній і т.п. Якщо виникла помилка і необхідно перебудувати який-небудь графік, то потрібно знову звернутися до конкретного підвікна командою subplot і далі провести необхідні зміни.

3.7 Побудова графіків у полярних координатах

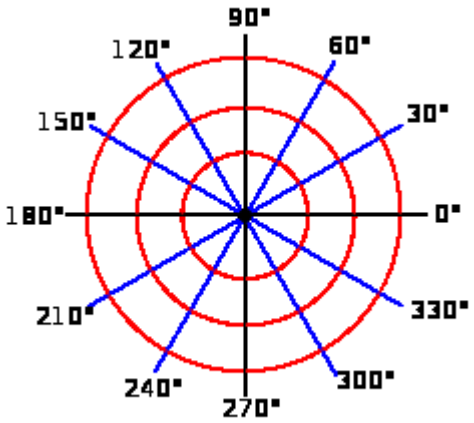


Рис. 3.6. Полярна система координат

Деякі графіки зручно будувати не в Декартових, а у полярних координатах (див. рис. 3.6).

Полярна система координат (СК) – це двовимірна система координат, в якій кожна точка на площині визначається двома числами: ***полярним кутом*** **φ** та ***полярним радіусом*** **ρ**:

$$\rho = f(\varphi). \tag{3.1}$$

Полярна СК задається ***нульовим променем*** або ***полярною віссю***. Точка, з якої виходить цей промінь називається началом координат або ***полюсом***. Радіальна і кутова координати можуть приймати значення від 0 до ∞. Кутова координата може бути як додатною, так і від‘ємною. Додатному куту відповідає поворот полярного радіуса проти годинникової стрілки, а від’ємному – за годинниковою стрілкою. Для побудови графіків у полярних координатах використовують функцію

polar(fi,ro)

Дуже часто полярні графіки задають так званими параметричними формулами:

$$\begin{cases} x = f_x(\varphi), \\ y = f_y(\varphi). \end{cases} \tag{3.2}$$

Тоді після розрахунку Декартових координат треба визначити за теоремою Піфагора полярний радіус і скористатися функцією polar. Графік функції, визначеної параметрично, можна побудувати і функцією plot, але у цьому разі, щоб не спотворити пропорції графіка, треба доповнити його командою axis equal.

3.8 Завдання

1. Ознайомитися з функціями для побудови та оформлення графіків різних типів.
2. Для заданих чотирьох математичних функції відповідно до варіанту у табл. 3.2 написати програми для побудови їх графіків:
   a) у чотирьох різних графічних вікнах;
   b) в одному графічному вікні;
   c) у двох підвікнах, розташованих по горизонталі (непарні варіанти) та по вертикалі (парні варіанти) по 2 графіки у кожному;
   d) в одному графічному вікні, розбитому на 4 підвікна, по одному графіку в кожному підвікні.

У кожному випадку зробити підписи осей, графіків, зробити кожен графік різним кольором, різним типом лінії та (або) з різними маркерами. Перші 2 графіки побудувати тільки відрізками прямих між точками, третій графік зобразити тільки маркерами, а четвертий – і маркерами і відрізками прямих між ними. Для третього і четвертого графіків кількість точок в масивах абсцис та ординат підібрати так, щоб можна було б побачити тип маркера. Вивести легенду графіків у зручному місці.

Апробувати дію функцій масштабування системи координат.

Таблиця 3.2

№ вар.	Функція 1	Функція 2	Функція 3	Функція 4
1	$y_1 = x - \sin x$	$y_2 = -\cos x$	$y_3 = x + 1/x$	$y_4 = \cos x + \lvert\cos x\rvert$
2	$y_1 = \arctan x + 2^x$	$y_2 = x + \log x$	$y_3 = \lvert x^2 - 5\lvert x\rvert + 6\rvert$	$y_4 = 1 - \cos x$
3	$y_1 = x^3 - 3x$	$y_2 = e^{-x}\sin x$	$y_3 = 1/2(\lvert\sin x\rvert - \sin x)$	$y_4 = \sin x + \cos x$
4	$y_1 = x^2 - x^4$	$y_2 = xe^{-x}$	$y_3 = \arcsin(x-1)$	$y_4 = \sqrt{\lvert x\rvert}$
5	$y_1 = x^2 + \lvert x\rvert - 2$	$y_2 = e^x / x$	$y_3 = x^3 + 3x$	$y_4 = 1/\sqrt[3]{(1-x^2)^2}$
6	$y_1 = \lvert x^2 - 1\rvert$	$y_2 = e^{-x^2}$	$y_3 = \tan(x/3)$	$y_4 = \sqrt[3]{(1-x)/(1+x)}$
7	$y_1 = \cos^{2/3} x$	$y_2 = e^{-x}/x^2$	$y_3 = -x^2 + x + 2$	$y_4 = \arcsin\lvert x-1\rvert$
8	$y_1 = \tan^{2/3} x$	$y_2 = 3x/(1+x^3)$	$y_3 = 1/2(\lvert\arctan x\rvert + \arctan x)$	$y_4 = \sqrt{(2-x)^3}$

3. Написати програму для побудови графіків інтервально-заданих функцій згідно з табл. 3.3. Оформити графіки відповідно до табл. 3.4.

Таблиця 3.3

№ вар.	функція 1	інтервал 1	функція 2	інтервал 2	функція 3	інтервал 3
1	$y = \sin 2x$	0…3	$y = \cos 3x$	3…5	$y = \sin x + \cos x$	5...10
2	$y = \sin(4x-1)$	-1...4	$y = \cos^2 2x$	4…7	$y = \sin 3x$	7...11
3	$y = \sin x$	-2...0	$y = 2\cos x + \sin x$	0…4	$y = \cos^2 x$	4…8
4	$y = \sin(x+2)$	-5...-1	$y = \cos 2x$	-1…3	$y = \sin^3 x$	3…9
5	$y = \sin^2(x-4)$	0…4	$y = \cos 2x + \sin 3x$	4…8	$y = \sin 4x$	8...12
6	$y = \cos^2 x$	2…5	$y = 10\sin(x-3)$	5…8	$y = \cos^2(x-3)$	8...10
7	$y = \cos 3x$	0…2	$y = 2\sin 4x$	2…7	$y = \cos x - \sin^2 x$	7...10
8	$y = 2x\sin x$	-4…4	$y = \cos^2 x$	4…8	$y = \sin x - 2\cos^2 x$	8...11

Таблиця 3.4

№ бриг ади	Колір лінії	Тип маркера	Тип лінії	Положення легенди	Тип шрифту та розмір тексту
1	жовтий	крапка	пунктир	1	Arial, 11
2	голубий	кружок	штрихова	2	Times New Roman, 12
3	пурпурний	хрестик	штрих-пунктир	3	Verdana, 13
4	червоний	плюс	дрібний пунктир	4	Arial, 12
5	зелений	зірка	штрихова	0	Times New Roman, 11
6	синій	квадрат	штрих-пунктир	-1	Verdana, 13
7	жовтий	ромб	пунктир	1	Arial, 12
8	червоний	кружок	дрібний пунктир	3	Times New Roman, 11

4. Побудувати графік кола з радіусом R і центром у точці з координатами (0, 0) у полярних координатах за рівнянням

$$\rho(\varphi)=R=const, \qquad \varphi\in[0,2\pi] \tag{3.3}$$

та у Декартових координатах за рівняннями

$$\begin{cases} x = R\cos\varphi \\ y = R\sin\varphi \end{cases} \tag{3.4}$$

5. За даними табл. 3.5 побудувати графіки у полярних координатах.

3.9 Методичні вказівки та рекомендації

1. При виконанні пункту 2 ***для кожного графіка підбирайте діапазон зміни аргументу*** так, щоб запобігти занадто великих або безкінечних значень функції або протяжних діапазонів з майже незмінними значеннями функцій.
2. При виконанні пункту 3 на кожному інтервалі задавайтесь не більше, ніж 10 точками.
3. Для копіювання графічної фігури застосовуйте із функції меню ***Edit*** команду ***Copy Figure***. Щоб надати можливість подальшого редагування графіка, його можна завантажити спочатку в графічний редактор ***Microsoft Office Visio***, а потім перевантажити у *Word*.

Таблиця 3.5

Вар.	Рівняння	Назва поверхні	φ	Пара-метри
1	$\rho = k\varphi$	Спіраль Архімеда	$0 \div 6\pi$	$k = 1$
2	$\begin{cases} x = R\cos^3\varphi \\ y = R\sin^3\varphi \end{cases}$	Астроїда	$0 \div 2\pi$	$R = 3$
3	$\rho = 1 - \sin\varphi$	Циклоїда	$0 \div 2\pi$	
4	$\rho = 2 - 4\sin\varphi$	Завиток Паскаля	$0 \div 2\pi$	
5	$\begin{cases} x = k\cos\varphi - \cos(k\varphi) \\ y = k\sin\varphi - \sin(k\varphi) \end{cases}$	Епі-циклоїда	$0 \div 2\pi$	$k = 4$
6	$\rho = \dfrac{1}{1 - \cos\varphi}$	Парабола	$0 \div 2\pi$	
7	$\begin{cases} x = 2R\,\mathrm{ctg}\,\varphi \\ y = R(1 - \cos^2\varphi) \end{cases}$	Локон Аньєзі	$\dfrac{\pi}{20} \div \pi - \dfrac{\pi}{20}$	$R = 2$
8	$\rho = \sin k\varphi$	Полярні рози	$0 \div 2\pi$	$k = 6$
9			$0 \div 8\pi$	$k = 7/4$
10			$0 \div 8\pi$	$k = 3/4$
11	$\begin{cases} x = R(k-1)\left(\cos\varphi + \dfrac{\cos(k-1)\varphi}{k-1}\right) \\ y = R(k-1)\left(\sin\varphi - \dfrac{\sin(k-1)\varphi}{k-1}\right) \end{cases}$	Гіпо-циклоїди	$0 \div 6\pi$	$k = 5/3$
12			$0 \div 4\pi$	$k = 11/2$
14	$\rho = e^{\sin\varphi} - 2\cos(4\varphi) + \sin^5\dfrac{2\varphi - \pi}{24}$	Серце	$0 \div 2\pi$	
15	$\begin{cases} x = 16\sin^3\varphi \\ y = 13\cos\varphi - 5\cos(2\varphi) - 2\cos(3\varphi) - \cos(4\varphi) \end{cases}$			

3.10 Контрольні запитання та завдання

1. Яка команда використовується для побудови двовимірного графіка?
2. Назвіть основні типи маркерів та як вони наносяться

3. Назвіть команди підпису осей та формат їх написання.
4. Які команди керування системами координат ви знаєте?
5. Як масштабуються координатні осі? Як нанести легенду?
6. Які команди керування графічними вікнами ви знаєте?
7. Що таке полярна система координат?
8. Яку команду треба виконати, щоб круг, побудований у Декартових координатах не був схожим на еліпс?

Лабораторна робота № 4

ОБРОБКА ТА РЕДАГУВАННЯ ГРАФІКІВ ЗА ДОПОМОГОЮ ФУНКЦІЙ ГРАФІЧНИХ ВІКОН

Мета роботи: вивчення можливостей редагування графіків за допомогою меню графічних вікон.

4.1 Деякі функції меню графічного вікна

Для редагування графіків, підпису осей, та ін. можна використовувати не тільки команди *MatLab*, а ще й функції меню та «кнопки» інструментів графічного вікна. Типове графічне вікно має меню, панель інструментів та поле для побудови графіка. Верхня частина цього вікна показана на рис. 4.1. На рис. 4.2 та рис. 4.3 зображені панелі інструментів, які виводяться відповідними функціями команди *View*.

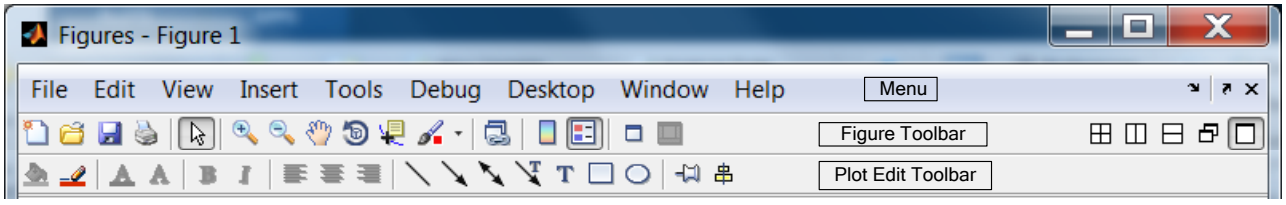


Рисунок 4.1. Заголовок, меню та інструменти графічного вікна

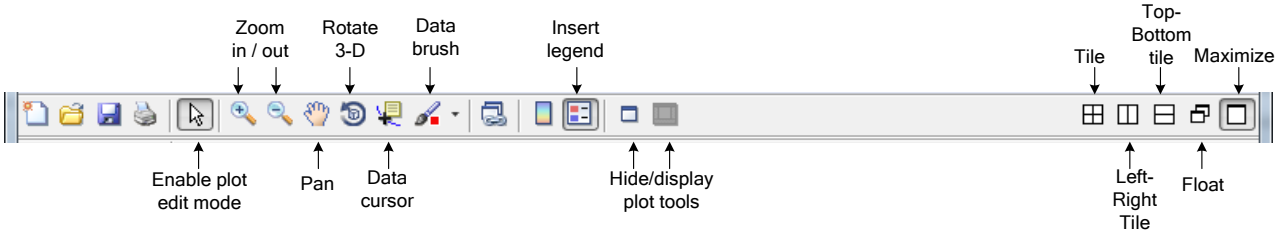


Рисунок 4.2. Панель *Figure Toolbar* графічного вікна

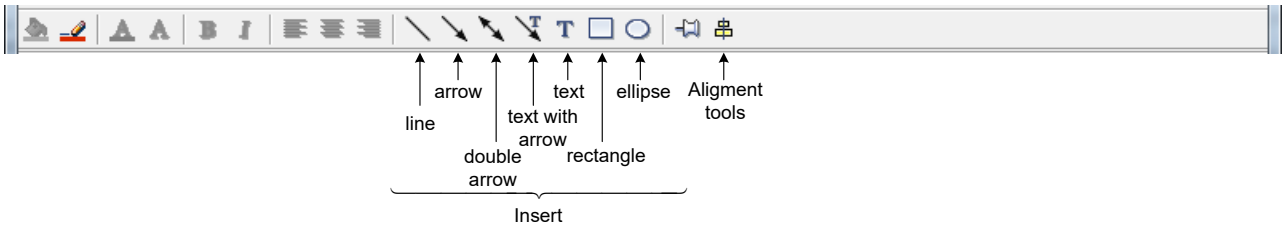


Рис. 4.3. Панель *Plot Edit Toolbar* графічного вікна

На рис. 4.4 відображене вікно, що відкривається при натисканні на кнопку *Alignment Tools* (вирівнювання).

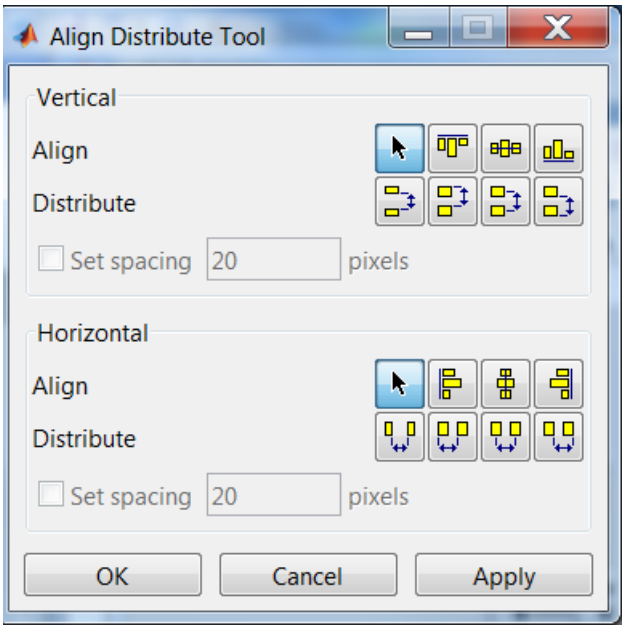


Рис. 4.4. Вікно, призначене для вирівнювання графічних об‘єктів

Перелічимо найважливіші пункти меню графічного вікна, виключивши із них загально відомі, що не потребують пояснень.

1) ***File*** (Файл):

- ***Close (^W)*** –закриття графічного вікна без збереження;
- ***Save (^S)*** – збереження графічного вікна у файлі з розширенням ***.fig***;
- ***Preferences → Figure Copy Template*** – налаштування графічного вікна;
- ***Print Preview*** – попередній перегляд графіка перед друком;
- ***Print*** – друк вмісту графічного вікна.

2) ***Edit*** (Редагування):

- ***Copy Figure*** – копіювання вмісту графічного вікна у буфер обміну;
- ***Copy Options*** – налаштування копіювання вмісту графічного вікна (доступні також у пункті меню ***File→ Preferences***);
- ***Figure Properties…*** – редагування властивостей графічного вікна;
- ***Axis Properties…*** – редагування властивостей системи координат;
- ***Current Object Properties…*** – редагування властивостей виділеного об’єкта;

- ***Colormap…*** – редагування карти кольорів (для тривимірних графіків та рисунків).

3) ***Viev*** (Вигляд):

- ***Figure Toolbar*** – активація/дезактивація панелі для редагування графічного вікна;
- ***Camera Toolbar*** – активація/дезактивація панелі керування камерою вигляду графічного вікна;
- ***Plot Edit Toolbar*** – активація/дезактивація панелі редагування графіків.

4) ***Insert*** (Вставка):

- ***Xlabel***, ***Ylabel***, ***Zlabel*** – вставка підпису осей абсцис, ординат, аплікат;
- ***Title*** – вставка підпису графіка;
- ***Legend*** – вставка легенди графіків;
- ***Line*** – вставка лінії;
- ***Arrow*** – вставка стрілки;
- ***Text Arrow*** – вставка стрілки з текстом;
- ***Double Arrow*** – вставка подвійної стрілки;
- ***TextBox*** – вставка тексту;
- ***Rectangle*** – вставка прямокутника;
- ***Ellipse*** – вставка еліпса;
- ***Axes*** – вставка системи координат.

5) ***Tools*** (Інструменти):

- ***Edit Plot*** – режим редагування графіка;
- ***Zoom In***, ***Zoom Out*** – збільшення та зменшення зони перегляду графіка;
- ***Pan*** – захват та рух графіка разом з координатною віссю;
- ***Rotate 3D*** – обертання графіка у тривимірній площині за допомогою миші;
- ***Data Cursor*** – визначення координат точок, виділених курсором;
- ***Brush*** – дає можливість виділити фрагменти графіка іншим кольором та іншою товщиною;

- ***Reset View*** – скидання всіх масштабних змін;
- ***Options*** – опції масштабу та захвату;
- ***Data Statistics*** – деякі статистичні дані по кожному з графіків (мінімум, максимум, середнє значення, інтервал і т.п.).

6) ***Desktop*** (Робочий стіл):

- ***dock figure*** – закріплює графік у вікні редагування властивостей графіка або на робочому столі;
- ***undock figure*** – відкріпляє графік від робочого стола або видаляє його з вікна редагування.

7) ***Window*** (Вікно) – переключення між різними графічними вікнами, якщо їх більше одного.

8) ***Help*** (Допомога): ***Graphics help, Plotting tools, Annotation Graph, Printing und exporting***

4.2 Редагування вмісту графічного вікна

Одним із способів перейти у цей режим є вибір опції ***Property editor*** із сукупності команд функції ***View*** основного меню графічного вікна. Після цього під графіком відкривається вкладка *Property editor - Figure*, зображена рис. 4.5. У такий стан *editor* повертається при щиглику мишею на фоні графічної фігури, що оточує систему координат.

Рис. 4.5. Вкладка *Property editor-Figure* графічного вікна

Вона дозволяє змінити колір фону вікна, надати або змінити його ім‘я та обрати палітру, що має сенс для тривимірних графіків та рисунків або фотографій.

Якщо клацнути кнопкою миші всередині системи координат або на одній з її осей, то відкривається вкладка *Property editor - Axes* (див. рис. 4.6).

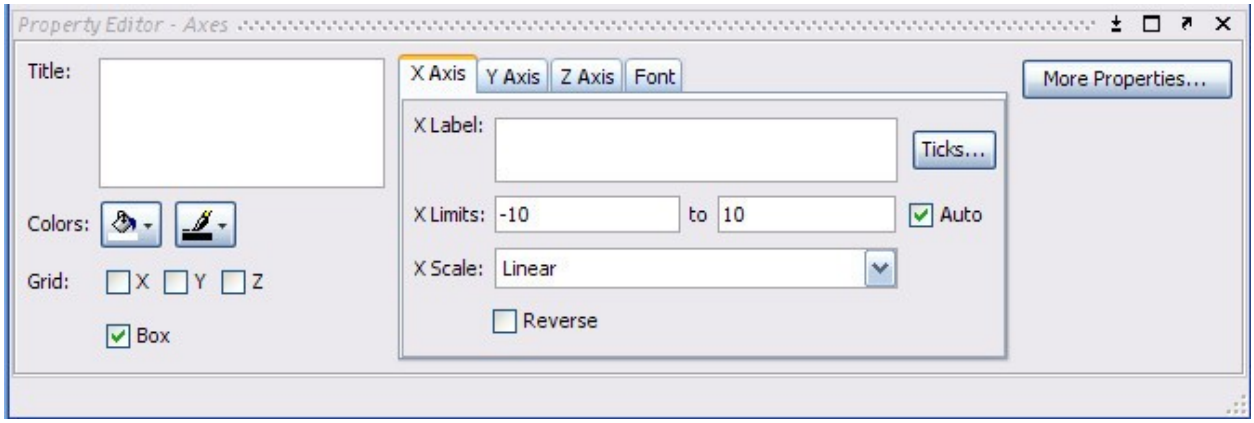


Рис. 4.6. Вкладка *Property editor- Axes* графічного вікна

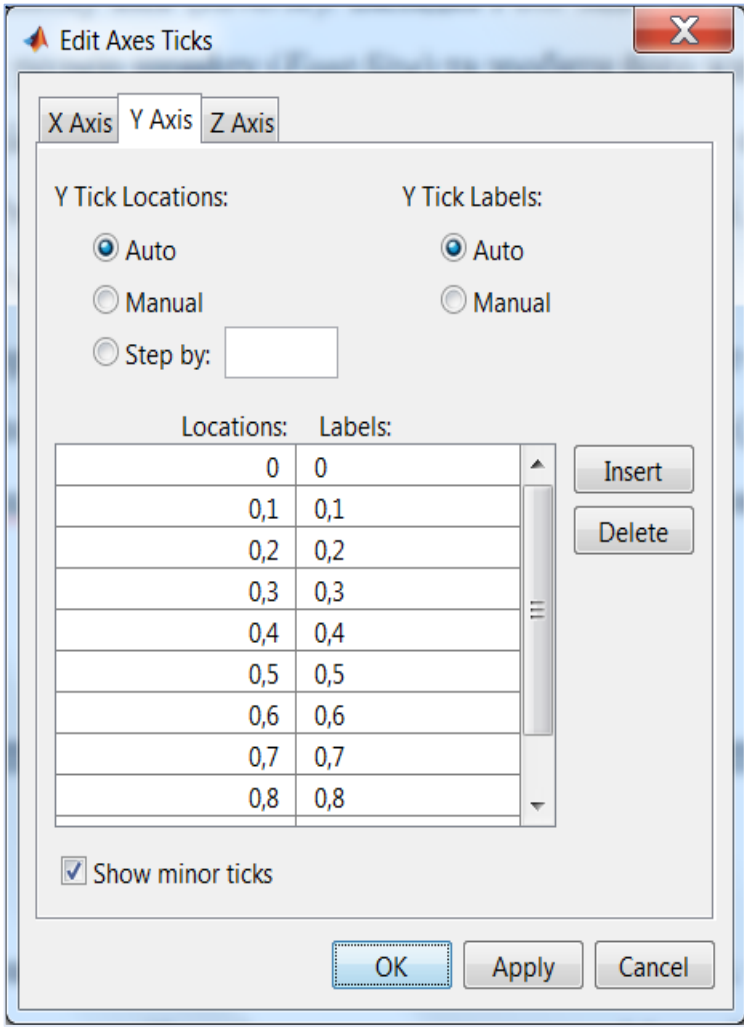


Рис. 4.7. Вікно редагування міток на осях

Вкладка ***Property editor - Axes*** дозволяє редагувати координатну сітку, а саме: створити назву графіка (*Title*), створити сітку по всім осям (*Grid*) та змінити її колір і колір заливки (*Colors*), вставити мітки осей (*X label, Y label, Z label*), змінити межі осей (*X limits, Y limits, Z limits*) та їх поділки (кнопка *Ticks*). Вигляд вікна *Edit Axes Ticks* показаний на рис. 4.7. змінити лінійну шкалу на логарифмічну (*Scale*) та реверсувати координатну вісь (*Reverse*).

Функція *Font* задає ім’я шрифту підпису осей (*Font Name*), розмір шрифту (*Font Size*) та зробити його жирним або курсивом (*Weight* та *Angle*). Для більш детальних опцій редагування осей слід натиснути кнопку *More Properties*.

Щиглик мишею на лінії або будь-якій точці графіка відкриває вкладку *Property editor – Lineseries* (рис. 4.8).

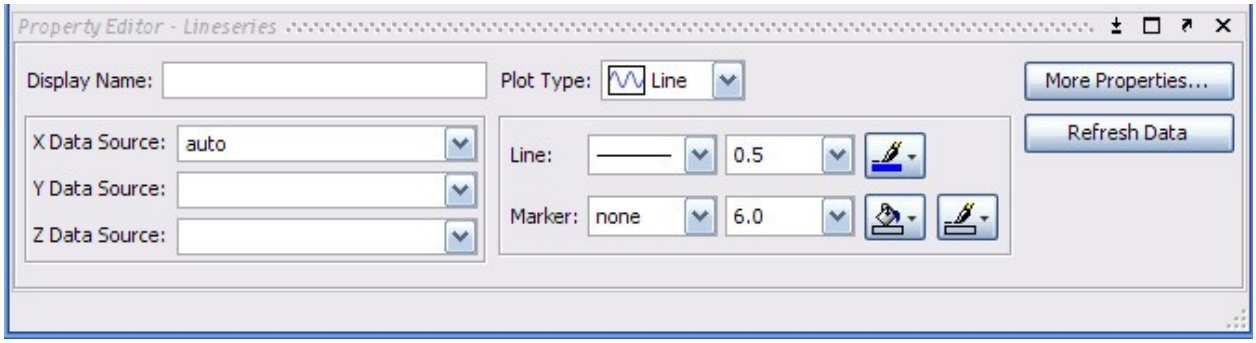


Рис. 4.8. Вкладка *Property editor- Lineseries* графічного вікна

Вкладка *Property editor - Lineseries* дозволяє редагувати сам графік, а саме: ввести текст для легенди (*Display Name*). вибрати тип графіка (*Plot Type*), тип лінії, її товщину та колір (*Line*), тип маркера, його розмір, колір та заливку (*Marker*). Для відображення більш детальних опцій слід натиснути кнопку *More Properties*.

4.3 Апроксимація графіків в діалоговому режимі

Графіки, побудовані за табличними значеннями аргументів і функцій, отриманими експериментально, неточно відображають результати експериментів внаслідок наявності похибок вимірювання і дискретного характеру інформації. Похибки виникають і при розв‘язанні математичних та прикладних науково-технічних задач чисельними методами. Зокрема це стосується розв‘язання диференційних рівнянь ітераційними методами з автоматичним вибором кроку чисельного інтегрування. В цьому разі кількість розрахованих точок навіть при досить високій точності їх розрахунку іноді виявляється недостатньою для якісної візуалізації.

Для згладжування таких результатів в пакеті *MatLab* передбачена можливість апроксимації табличних функцій.

Приклад 4.1. За даними табл. 4.1 побудувати графік функції P(v) та в разі необхідності апроксимувати табличну функцію степеневим поліномом методом найменших квадратів і побудувати графік апроксимуючої функції.

Таблиця 4.1.

Швидкість повітряного потоку, м/с	10	13	15.9	18.5	21.5	24.4	27.1	29.5	31.5
Потужність, Вт	25	62.5	125	175	250	375	500	625	800

Після виконання програми

```
V=[10 13 15.9 18.5 21.5 24.4 27.1 29.5 31.5];
P=[25 62.5 125 175 250 375 500 625 800];
plot(V,P)
```

на екрані отримуємо графік, зображений на рис. 4.9.

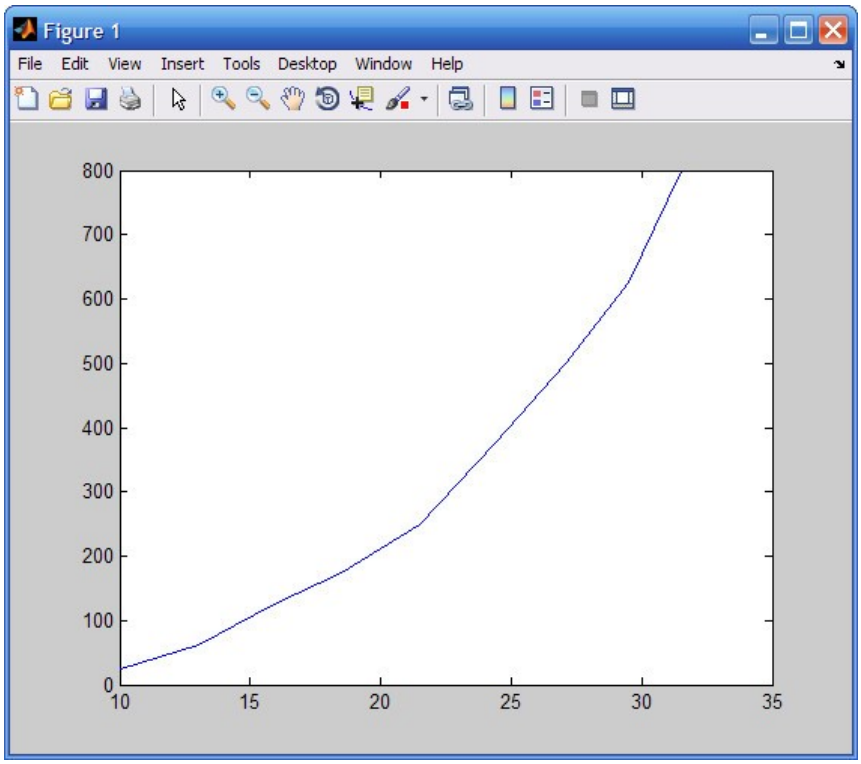
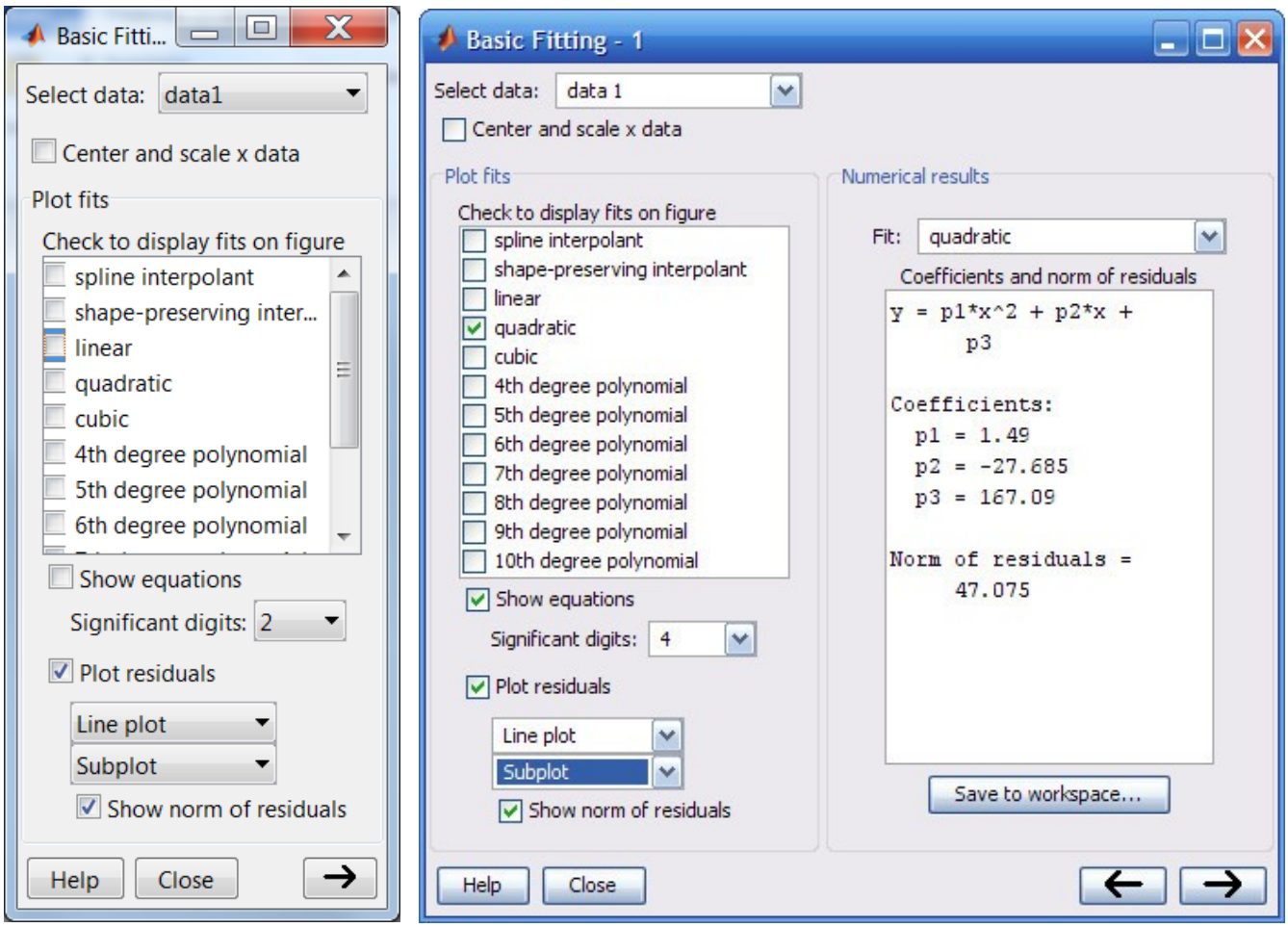


Рис. 4.9. Графік, побудований за даним табл. 4.1

Як бачимо, графік має видимі точки зламу. Щоб позбавитися від цього недоліку скористаємося опцією *Basic Fitting* меню *Tools*. При цьому відкриється вікно, зображене на рис. 4.10а. В ньому пропонуються інтерполяція кубічними сплайнами (*Spline Interpolant*) та апроксимація степеневими поліномами 1-го (*Linear*), 2-го (*Quadratic*), *3-го* (*Cubic*) та 4-10-го порядку (*4th-10th degree polynomial*), що здійснюється методом найменших квадратів



а) б)

Рис. 4.10. Вікно *Basic Fitting* до (а) та після (б) установки параметрів

На рис. 4.10б показано вікно згладжування після вибору у ньому методу апроксимації *Quadratic*, установки прапорця у полі *Show Equations*, вибору кількості цифр у коефіцієнтах апроксимуючої параболи (*Significant digits*).

Для порівняння апроксимованої залежності і реального графіка можна вивести їх різницю, поставивши прапорець *Plot Residuals*, в якому вибрати тип лінії (*Bar*, *Scatter* або *Line*) та спосіб відображення (*Subplot* – вікно розіб'ється на 2 підвікна або *Separate Figure* – відкриється друге графічне вікно). Також можна поставити прапорець на *Show norm of residuals* (показати суму квадратичних відхилень апроксимуючої функції від вихідної табличної у вузлових точках).

В результаті отримаємо графік, показаний на рис. 4.11.

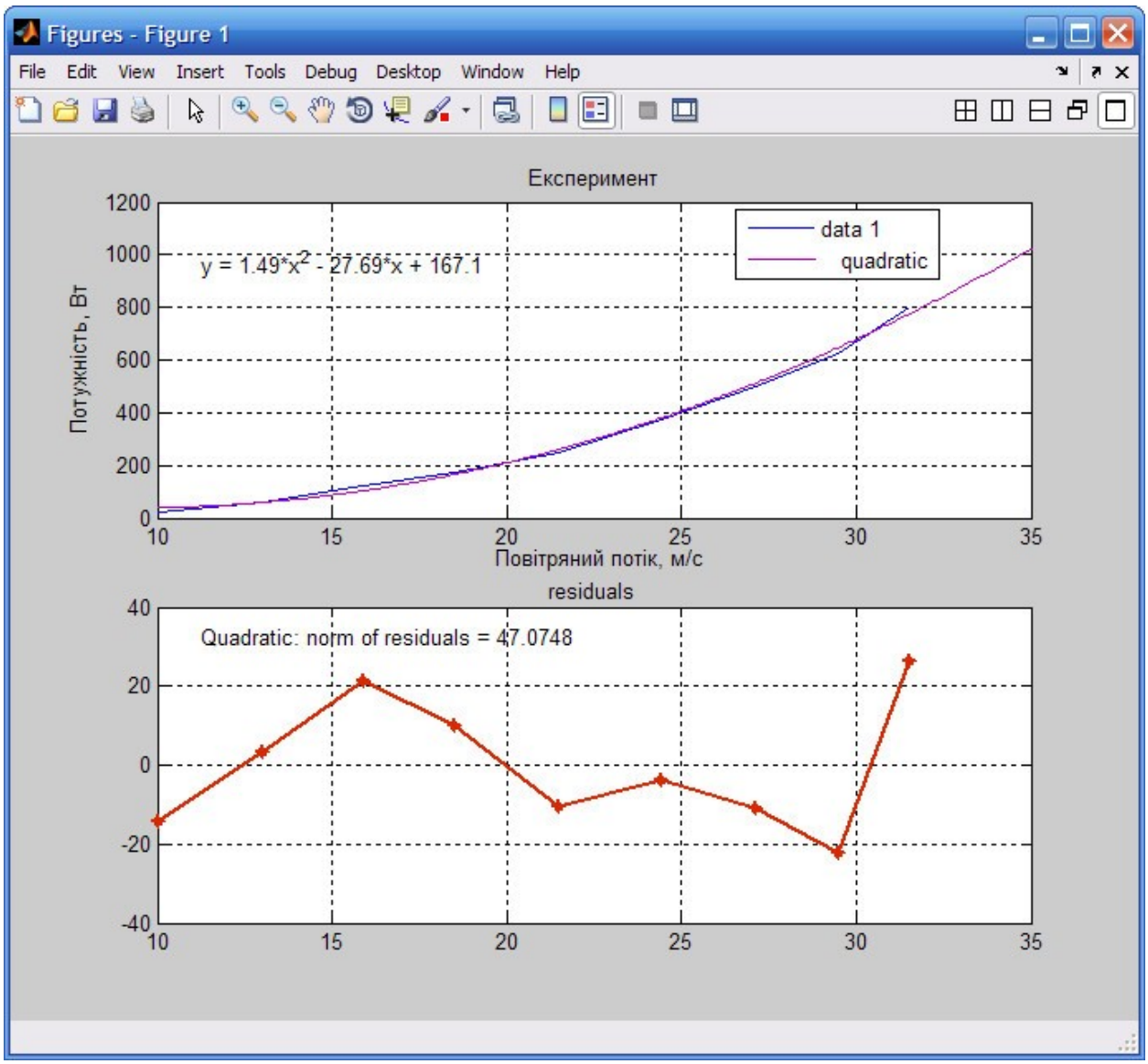


Рис. 4.11. Вихідний та апроксимований графіки до прикладу 4.1 та їх різниця

4.4 Редагування тексту за допомогою функцій TeX

Текстові об‘єкти *MATLAB* можна створювати і коригувати засобами видавницького текстового редактора *TeX*. Він може створювати верхні та нижні індекси, спеціальні символи та грецькі літери. Для активації функцій редактора *TeX* необхідно перед написанням тексту поставити один з ключів, приведених у табл. 4.2. Можна застосовувати як один, так і кілька поспіль ключів.

Таблиця 4.2

Редагування тексту			
Значення	***Команда***		
Виділення курсивом	\it Текст		
Виділення жирним	\bf Текст		
Жирний курсив	\bf\it текст		
Ім'я та розмір шрифту	\fontname {ІмяШрифту}\fontsize {Розмір} текст		
Верхній індекс	текст^{індекс}		
Нижній індекс	текст_{індекс}		
Грецькі символи			
Значення	***Команда***	***Значення***	***Команда***
Альфа (α)	\alpha	Ро (ρ)	\rho
Бета (β)	\beta	Сігма (σ)	\sigma
Гама (γ)	\gamma	Тау (τ)	\tau
Дельта (δ)	\delta	Фи (φ)	\phi
Епсілон (ε)	\epsilon	Чи (χ)	\chi
Ета (η)	\eta	Пси (ψ)	\psi
Тета (θ)	\theta	Омега (ω)	\omega
Капа (κ)	\kappa	Велика Гама (Г)	\Gamma
Лямбда (λ)	\lambda	Велика Дельта (Δ)	\Delta
Мю (μ)	\mu	Велика Тета (Θ)	\Theta
Ню (υ)	\nu	Велика Лямбда (Λ)	\Lambda
Ксі (ξ)	\xi	Велика Фі (Ф)	\Phi
Спеціальні символи			
Значення		***Команда***	
Менше-рівно ($\leq$)		\leq	
Більше-рівно ($\geq$)		\geq	
Двостороння стрілка ($\leftrightarrow$)		\leftrightarrow	
Стрілка вліво ($\leftarrow$)		\leftarrow	
Стрілка вправо ($\rightarrow$)		\rightarrow	
Плюс-мінус ($\pm$)		\pm	
Нескінченність (∞)		\propto	
Часткова похідна (∂)		\partial	
Стрілка униз ($\downarrow$)		\downarrow	
Стрілка угору ($\uparrow$)		\uparrow	

Розглянемо приклад застосування функцій *TeX*. Нехай є знятий графік перехідного процесу кутової швидкості та моменту двигуна $\omega = f(t)$ та $M = f(t)$, показаний на рис. 4.12. Потрібно підписати осі 14-м шрифтом *Times New Roman*, а сам графік жирним курсивом та 16-м шрифтом. Літеру ω зробити грецькою за допомогою функції *TeX*. На графік нанести сітку та стрілками показати швидкість і момент (напис зробити 12-м шрифтом). Лінії зробити товщиною 2 і різними кольорами.

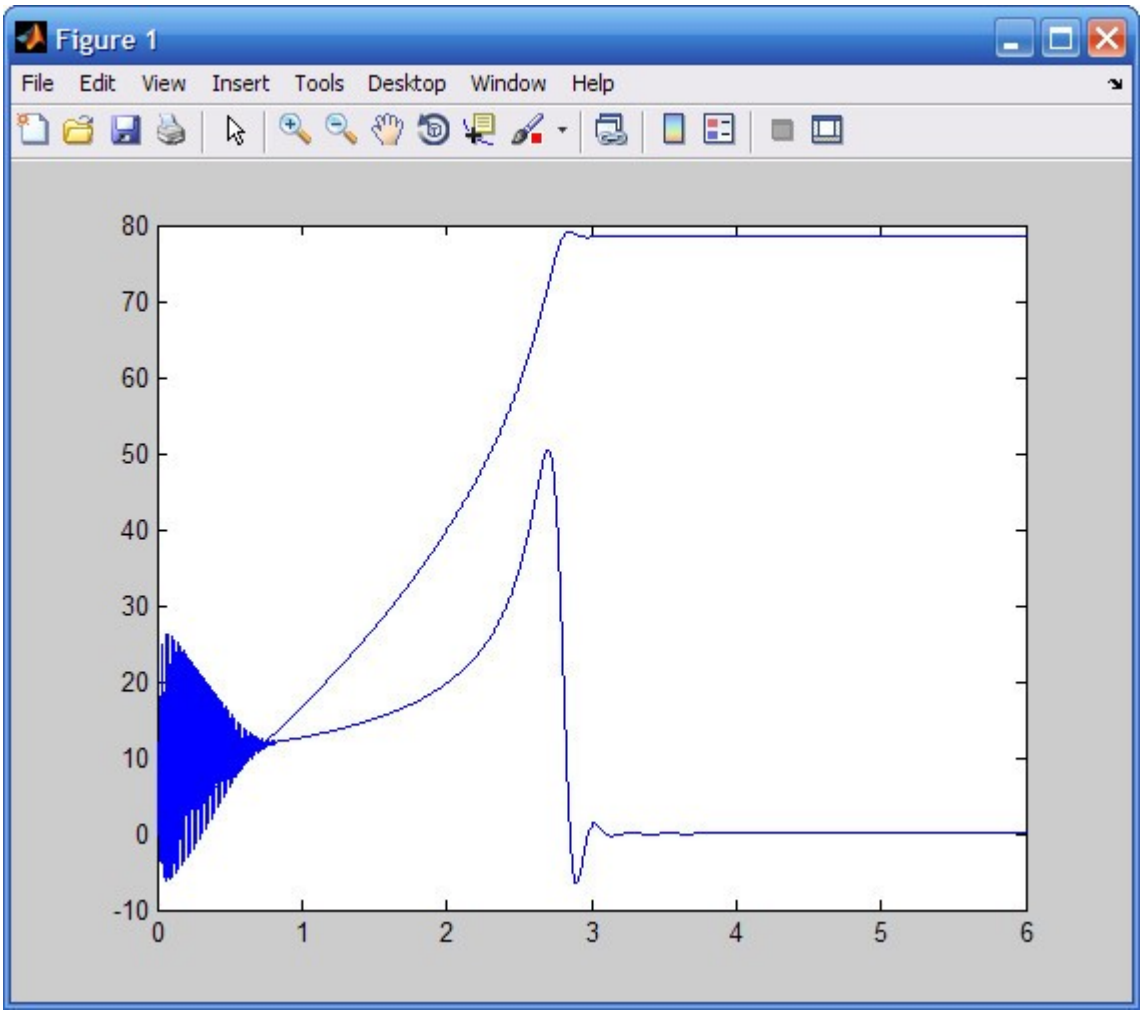


Рис. 4.12. Вихідні графіки перехідних процесів

Відкриваємо вікно *Property editor – Axes*. У полі *Title* пишемо

\bf\it\fontname{TimesNewRoman}\fontsize{16} Графік перехідного процесу швидкості та моменту

Потім ставимо прапорець *Grid* на осях X та Y. Далі у полі *Xlabel* пишемо

\fontname{TimesNewRoman}\fontsize{14} Час перехідного процесу t, с

У полі *Ylabel* пишемо

\fontname{TimesNewRoman}\fontsize{14}Швидкість \omega, рад/с Момент М, Н*м

Далі два рази натискаємо на криву швидкості і обираємо вкладку *Property editor – Lineseries*. Виставляємо там товщину лінії – 2. Те ж саме повторюємо для графіка моменту. Змінюємо також колір цієї графічної функції на червоний. За допомогою меню *Insert→Text Arrow* вставляємо стрілки з текстом \fontsize{12}Швидкість\omega для швидкості та \fontsize{12}Момент М для моменту. Закриваємо вкладку і отримуємо графік, показаний на рис. 4.13.

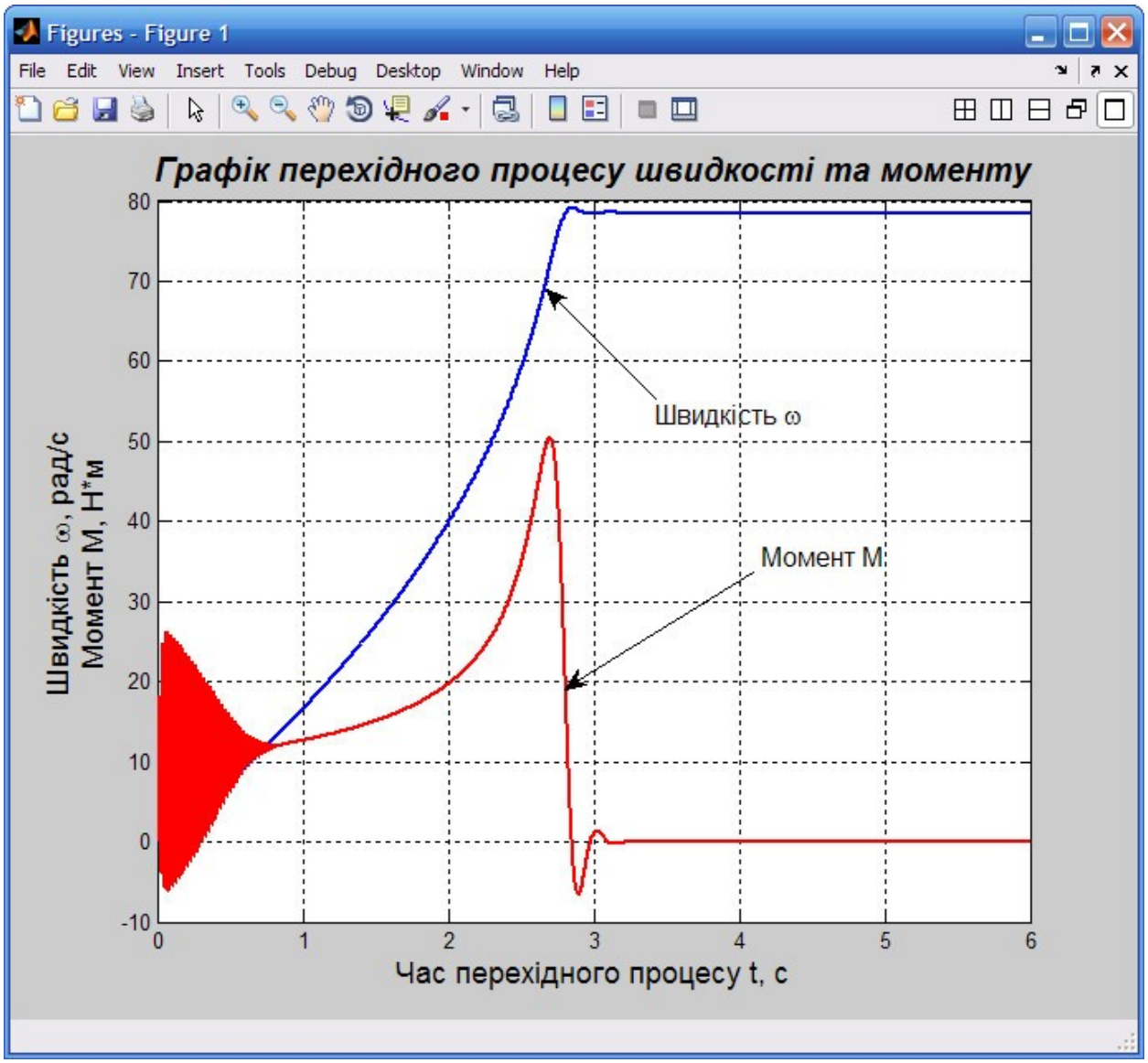


Рис. 4.13. Приклад оформлення графіків з використанням функцій *TeX*

4.5 Завдання

1. Ознайомитися з правилами оформлення графічного вікна, способами апроксимації та командами функції TeX.
2. Для даних з табл. 4.3 виконати такі дії:
   a) побудувати залежності $H = f(Q)$ та $\eta = f(Q)$ на одному графіку, на різних та у двох підвікнах одного графічного вікна;
   b) за допомогою функцій редагування графічного вікна оформити підписи осей, титулу, стрілками показати кожен з графіків, побудувати графіки різними кольорами, показати маркери, вивести легенду;
   c) підібрати метод та апроксимувати кожен з графіків, отримати рівняння кожної кривої, вивести графіки різниць типу *bar* або *scatter* в різних підвікнах одного графічного вікна для кожної кривої;
   d) побудувати графіки за отриманими апроксимованими рівняннями на одному графіку із експериментальними і порівняти результати.

Оформлення графіків проводити відповідно до номеру варіанту згідно табл. 4.4.

Таблиця 4.3

H	4.2	7.7	9.9	11.1	13.2	14.4	16.4	17.8	19.3	21.3	21.9	22.4	22.8	23.1
Q	4.24	3.75	3.43	3.21	2.86	2.66	2.26	1.95	1.55	0.95	0.69	0.44	0.12	0
η, %	28.8	47.1	55.8	58.9	65.2	66.9	69.9	69.3	65.9	53.1	45.1	32.7	9.49	0

4.6 Методичні вказівки та рекомендації

1. Апробуйте якомога більше функцій графічного вікна та запам‘ятайте ті з них, які вам сподобались.
2. Побудову заданого графіка можете замінити побудовою будь-яких інших графіків за даними, отриманими при виконанні лабораторних робіт з дисциплін, що вивчаються паралельно.

Таблиця 4.4

№ вар.	1	2	3	4	5	6	7	8
Підпис осі H	Times New Roman 14	Arial 12 жирний	Verdana 13 курсив	Times New Roman 12	Verdana 12 жирний курсив	Arial 14 жирний курсив	Times New Roman 13	Verdana 14 жирний
Підпис осі Q	Verdana 14 жирний	Times New Roman 13	Arial 14 жирний курсив	Verdana 12 жирний курсив	Times New Roman 12	Verdana 13 курсив	Arial 12 жирний	Times New Roman 14
Підпис осі η	Times New Roman 12	Arial 14 жирний курсив	Verdana 12 жирний курсив	Times New Roman 13	Verdana 14 жирний	Times New Roman 14	Verdana 13 курсив	Arial 12 жирний

Продовження таблиці 4.4

Титули	Times New Roman 16	Arial 16 жирний	Verdana 16 курсив	Times New Roman 16	Verdana 16 жирний курсив	Arial 16 жирний курсив	Times New Roman 16	Verdana 16 жирний
Колір H	червоний	синій	зелений	чорний	жовтий	рожевий	голубий	синій
Колір Q	синій	жовтий	чорний	голубий	рожевий	жовтий	чорний	зелений
Колір η	зелений	чорний	голубий	рожевий	зелений	червоний	зелений	чорний
Маркер H	ромб	хрестик	точка	п’ятикутна зірка	три-ник угору	плюс	квадрат	кружок
Маркер Q	шестик. зірка	три-ник вліво	кружок	ромб	хрестик	три-ник униз	точка	зірка
Маркер η	три-ник угору	кружок	квадрат	три-ник вліво	шестик. зірка	точка	три-ник вправо	ромб
Колір графіка різниць	жовтий	зелений	рожевий	зелений	червоний	чорний	жовтий	жовтий
Маркер графіка різниць	кружок	плюс	шестик. зірка	точка	ромб	кружок	зірка	точка
Положення легенди	1	2	3	4	1	2	3	4
Тип лінії Q	пунктир	штрих.	штрих пункт.	суцільна	суцільна	пунктир	штрих пункт.	штрих.
Тип лінії H	штрих.	штрих пункт.	суціл.	штрих пункт.	штрих.	суціл.	штрих.	суціл.
Тип лінії η	штрих-пункт.	суціл.	пункт.	штрих.	пункт.	штрих пункт.	пункт.	пункт.
Тип лінії графіка різниць	суцільна	пункт.	штрих.	пункт.	штрих пункт.	штрих.	суціл.	штрих пункт.

4.7 Контрольні запитання та завдання

3. Назвіть основні параметри та пункти меню графічного вікна.

4. Як відкрити режим редагування графічного вікна?

5. Як змінювати параметри координатної сітки?

6. Як змінювати параметри ліній графіка?

7. Як змінювати параметри графічного вікна?

8. Як додавати на графік стрілки з текстом та інші об’єкти?

9. Як будуються графіки за знятими експериментальними даними?

10. Як виконати апроксимацію табличної функції?

11. Які типи апроксимації існують?

12. Назвіть основні команди функції *TeX*.

Лабораторна робота № 5
ОСНОВИ ПРОГРАМУВАННЯ В СЕРЕДОВИЩІ *MATLAB*

Мета роботи: навчитися основам програмування алгоритмічною мовою MATLAB.

5.1 Основні правила створення і виконання *script*-файлів у пакеті *MATLAB*

Щоб зберегти послідовність операторів, що виконують певний алгоритм, у пам‘яті як єдине ціле, придатне для подальшого використання та редагування у вікні текстового редактора *edit* створюють файли, що зберігаються під маскою **.m* і називаються *m*-файлами. М-файли розподіляються на *script*-файли та *m*-функції. *Script*-файли – це головні програми, які у багатьох сучасних програмних продуктах називають «сценаріями».

Перед формуванням програмних файлів треба створити власну папку і зробити її доступною, як це описано у пункті 1.3 (методичні вказівки до виконання лабораторної роботи 1.

Для створення *script*-файлу треба клацнути мишею по віртуальній кнопці *New Script* у командному вікні MATLAB або виконати клавішну команду ^N або ввести з командного рядка команду edit. В усіх перелічених випадках на екрані з’явиться вікно текстового редактора, зображене на рис. 5.1.

Введення інформації виконується за тими ж правилами, що і в командних рядках, а саме:

- в кожному рядку можна набрати одну (один) або декілька команд або операторів, що відокремлюються одне від одного комою «,» або крапкою з комою «;»; останній символ має сенс ставити тільки після оператору присвоєння з метою заборонити виведення результату цієї операції на екран після її виконання; після останнього оператора у рядку кому можна не ставити;
- перехід на наступний рядок виконується клавішею *Enter*;

- елементи операторів відокремлюються один від одного пробілами або спеціальними символами до яких належать знаки операцій, апострофи та дужки; там де стоять спеціальні знаки можна вставляти довільну кількість пробілів;
- у будь-якому рядку можна записувати коментарі, ознакою початку яких є знак «%», а кінцем – кінець рядка;
- щоб розмістити один оператор у декількох рядках, у кінці кожного рядка ставлять три крапки «…»;
- у кінці програмного файлу не треби ставити оператор end.

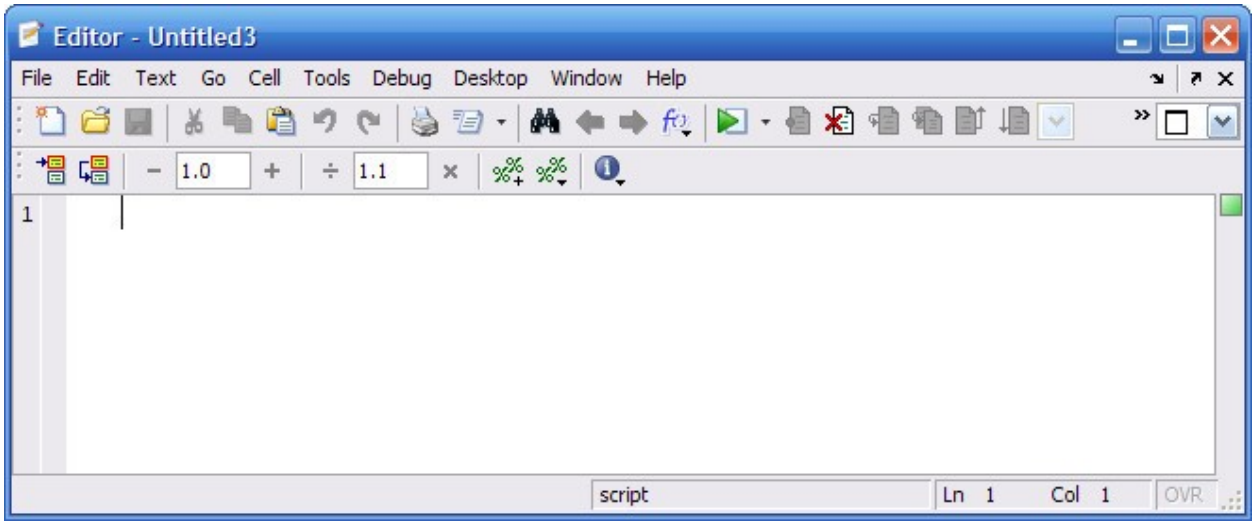


Рис. 5.1. Вікно текстового редактора

Після завершення набору *script*-файлу його необхідно зберегти у створеній заздалегідь папці клавішею *Save as…* та запустити на виконання клавішею *Run* або просто натиснути *F5* на клавіатурі, або набрати у командному рядку ім‘я файлу та натиснути клавішу *Enter*. Якщо у *script*-файлі будуть помилки, *MATLAB* повідомить про це у командному вікні з посиланням на конкретний рядок програми. Після усунення помилок в командне вікно (*Command Window*) будуть виведені результати виконання програми.

При створенні *script*-файлів дотримуйтесь таких рекомендацій:

- починайте файл з коментарів, у яких поясніть призначення програми, вкажіть своє прізвище і групу, наведіть за своїми міркуваннями іншу корисну інформацію, наприклад, номер лабораторної роботи, дату, тощо;
- до введення основних операторів доцільно набрати команди clc – очистити екрану, clear all – очистити пам‘ять, close all – закрити усі графічні фігури (якщо ви працюєте з графічними операторами), format compact– установити компактний стиль виведення інформації;
- якщо ви хочете бачити на екрані не тільки результати виконання операторів, але й самі оператори, скористайтеся командою echo on; відмінити її може команда echo off;
- доповнюйте програму коментарями, що пояснюють хід її виконання.

5.2 Характеристика лінгвістичних можливостей пакету *MATLAB*

Пакет *MATLAB* має у своєму складі розвинену алгоритмічну мову. З основними операторами цієї мови можна ознайомитися за допомогою команди

```
>>help lang
```

Наведемо скорочений список лінгвістичних конструкцій, що виводить ця команда з коментарями українською мовою:

```
Programming language constructs (конструкції алгоритмічної мови).
 Control flow.
   if          - Conditionally execute statements (умовний оператор).
   else        - Execute statement if previous IF condition failed (гілка «інакше»).
   elseif      - Execute if previous IF failed and condition is true( гілка «інакше»)..
   end         - Terminate scope of control statements (кінець складеного оператора).
   for         - Repeat statements a specific number of times (заголовок циклу з відомим
числом повторень).
   while       - Repeat statements an indefinite number of times (заголовок ітераційного
циклу з передумовою.
   break       - Terminate execution of WHILE or FOR loop (переривання циклу).
   switch      - Switch among several cases based on expression (ключ до оператора
вибору).
```

case - SWITCH statement case (оператор вибору).

otherwise - Default SWITCH statement case (гілка «інакше» оператора вибору).

try - Begin TRY block (оператор спроби).

catch - Begin CATCH block (гілка «інакше» оператора спроби).

error - Display message and abort function (переривання виконання програмного файлу з виведенням повідомлення про помилку).

Evaluation and execution.

eval - Execute string with MATLAB expression (виконувати рядок символів як вираз).

feval - Execute the specified function (виконувати рядок символів як функцію).

run - Run script (виконати script-файл).

Message display.

disp - Display array (виведення рядку символів або масиву).

Interactive input.

input - Prompt for user input (введення даних з клавіатури).

До перелічених вище операторів можна додати функції menu (меню), fprintf (форматне виведення), pause (пауза).

5.3 Характеристика основних операторів

Оператор консольного вводу input має формат:

VarName = input ('Prompt')

При виконанні цього оператора на екран виводиться запрошення Prompt у вигляді текстового повідомлення та перехід до режиму очікування користувачем введення значення змінної з клавіатури, що закінчується натисканням клавіші *Enter*, після чого введене значення присвоюється змінній з іменем VarName та зберігається у робочому просторі, наприклад,

```
» n = input ('Input  length of vector   n=')
Input  length of vector   n= <пауза до введення значення> 5 <Enter>
n =
    5
```

Оператор неформатованого виводу disp має формат

disp (VarName)

Він виводить значення змінної на екран без відображення її імені:

```
» k=1:10;
» disp(k)
   1    2    3    4    5    6    7    8    9   10
```

При виведенні одним оператором змішаних (символьних і числових) даних числові дані необхідно перетворити у рядок символів, що досягається використанням функцій int2str (integer) та num2str (number):

```
» disp (['Довжина вектору складає ', int2str(length(k)), ‘ елементів’])
Довжина вектору складає 10 елементів
» R =10.6; disp (['R=’, num2str(R), ‘ Ом - активний  опір кола'])
R=10.6 Ом - активний  опір кола
```

Оператор затримки pause призупиняє виконання програми до натискання користувачем будь-якої клавіші. Оператор pause (n) виконує затримку виконання наступного оператора на n секунд.

Оператор меню menu, що має формат

VarName = menu ('Title’, 'Opt1', 'Opt2', …, ’ OptN’)

можна розглядати як різновидність оператору введення.

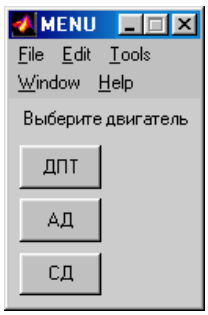


Рис. 5.2

Він генерує на екрані графічне вікно з кнопками, що надає користувачеві можливість вибору мишею однієї із заданих опцій. Порядковий номер обраної опції присвоюється змінній, ім‘я якої VarName зазначено у лівій частині оператора, наприклад, при виконанні оператора

» k = menu ('Оберіть двигун', 'ДПТ', 'АД', 'СД')

на екрані з‘являється вікно, зображене на рис. 5.2, і програма переходить у режим очікування вибору. При виборі опції ДПТ змінній k присвоюється значення 1, при виборі АД – значення 2, а при виборі СД – 3.

Складений оператор циклу з відомою кількістю повторень for…end має формат:

```
for VarName = Values % заголовок циклу
    % операторы тіла циклу:
    Stat1; Stat2; … StatN;
end % кінець циклу
```

Він організує виконання операторів Stat, що складають тіло циклу, послідовно для усіх значень Values змінної циклу VarName, після чого керування програмою передається оператору записаному після оператора кінця циклу end.

Значення змінної циклу можна задавати векторами будь-яких арифметичних констант (цілих і дійсних додатних та від'ємних чисел з додатним та від'ємним кроком, з рівномірним розподіленням впродовж дійсної або логарифмічної осі або розташованих у довільному порядку, комплексних чисел), а також рядками символів, наприклад,

```
for i = 1:10
for  x = -2.5:0.5:4
for k = 12:-2:0
for m = [-4:1, 1:4]
for  w = logspace( -2,3,6)
for  a = [3 18 -10 0, pi]
```

Параметр циклу може бути також матрицею. У цьому разі він по черзі приймає значення кожного стовпця матриці. Цикли можуть бути вкладеними. Тоді при кожному значенні змінної зовнішнього циклу змінна внутрішнього циклу перебирає послідовно усі свої значення.

Приклад 5.1. Сформувати вектор X із 3-х елементів шляхом введення його з клавіатури.

Складаємо програму

```
clc
for i = 1:3
   X(i) = input (['x(' int2str(i) ')=']);
end
X
```

В результаті її виконання отримуємо:

```
x(1)=5
x(2)=-8
x(3)=12
X =
    5   -8   12
```

Приклад 5.2. Сформувати матрицю розміром 4×5 за формулою $A_{ij}=2+ij$.

Складаємо програму

```
A=zeros(4,5);
for i=1:4
   for j=1:5
      A(i,j)=2+i*j;
   end
end
A
```

В результаті її виконання отримуємо:

```
A =
    3    4    5    6    7
    4    6    8   10   12
    5    8   11   14   17
    6   10   14   18   22
```

Складений умовний оператор if…else…end, що використовують для програмування розгалужених алгоритмів, має формат:

```
if  LogExpression
      Statements1; % оператори гілки «так»:
else
      Statements2; % оператори гілки «ні»:
end % кінець оператору
```

Він обчислює логічний вираз LogExpression і аналізує його результат. Якщо результатом є 1 (істина, англ. *true*), то виконуються оператори гілки «так» Statements1, інакше (тобто якщо результатом є 0 – хибність, англ. *false*) виконуються оператори гілки «ні» Statements2, наприклад,

```
if  x>0,  y = sqrt(x)
else y = x^2
end
```

При багатократному розгалуженні оператори if вкладаються один в інший. Щоб запобігти використання зайвого оператору end, замість конструкції else if в такому випадку використовують конструкцію elseif, наприклад,

```
if  x>0,  sg = 1
   elseif x<0, sg = -1
   else sg=0
end
```

У логічних виразах зазвичай застосовують операції відношення (порівняння) та/або логічні операції. Результатами таких операцій є істина (*true*) та хибність (*false*). У *MATLAB* їм відповідають логічні константи 1 та 0.

До операцій відношення належать:

> – більше; **>=** – більше або рівно; **<** – менше; **<=** – менше або рівно; **= =** – рівно; **~=** – не рівно.

При порівнянні двох масивів однакового розміру результатом є масив того ж розміру, складений із нулів та одиниць, як результатів поелементних операцій відношення. Якщо ж в одним із операндів є скаляр, то кожний елемент масиву по черзі порівнюється зі скаляром:

```
» 2>1
ans =
    1
» 1>=3
ans =
    0
» A = [1 8 3; 6 2 5];  B = [4 3 1; 2 6 8];
» A>3
ans =
    0    1    0
    1    0    1
» A<B
ans =
    1    0    0
    0    1    1
```

Логічні операції виконуються над логічними даними, тобто операндами логічних операцій в мажуть бути тільки 0 (хибність) та 1 (істина) або їх сполучення.

До основних логічних операцій в MATLAB належать:

& – "I" (логічне множення, або кон‘юнкція), в алгебрі логіки найчастіше позначається як ^;

| – "АБО" (логічне додавання, або диз‘юнкція), в алгебрі логіки найчастіше позначається як ∨;

xor – «виключне АБО»;

~ – «НЕ» (логічне заперечення, або інверсія), позначається символом ¬.

Операції | та & є двоаргументними, а ~ – одноаргументна.

Результати наведених операцій при різних сполученнях аргументів наведені у табл. 5.1.

Таблиця 5.1

x	y	x&y	x|y	x xor y	~x
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Логічні операції в пакеті *MATLAB* доповнюються такими логічними функціями:

any(x) – повертає 1, якщо деякі з елементів масиву x не нулі, і 0 у противному випадку;

all(x) – повертає 1, якщо усі елементи масиву x не нулі, і 0 у противному випадку;

find(x) – повертає індекси ненульових елементів масиву x.

Наприклад, при виконанні програми

```
x = [1 5 2 -7 4 -9]
if any(x<0)
   k = find(x<0),  x(k) = x(k).^2,  y = sqrt(x)
end
z = all(y>1)
if all(x~=0) v = log10(x), end
```

отримуємо

```
x =
    1    5    2   -7    4   -9
k =
    4    6
```

```
x =
    1    5    2   49    4   81
y =
   1.0000   2.2361   1.4142   7.0000   2.0000   9.0000
z =
    0
v =
         0   0.6990   0.3010   1.6902   0.6021   1.9085
```

Складений оператор вибору switch…case…otherwise…end, що застосовують при програмуванні багатократно розгалужених процесів, має у найпростішому випадку такий формат:

```
switch  VarName
        case Value1, Statements1
        case Value2, Statements2
        …
        case ValueN, StatementsN
        otherwise  StatementsAlternative
end
```

Для такого оператору гілка Statements1 виконується, якщо VarName = Value1, гілка Statements2 – якщо VarName = Value2, гілка StatementsN – якщо VarName = ValueN та гілка StatementsAlternative – коли ключова змінна отримує будь-яке інше значення. Конструкція otherwise не є обов‘язковою.

У більш складних випадках ключова змінна і її значення можуть задаватися виразами (Expression).

Складений оператор спроби try…catch…end, що застосовується при налагодженні програми має такий формат:

```
try  StatementeBase
catch  StatementAlternative
end
```

Оператор-альтернатива StatementAlternative виконується у тому випадку, коли при виконанні оператора-спроби фіксуються помилки. Текст повідомлення про помилку можна вивести за допомогою команди lasterr, наприклад,

```
» A=[1 2; 3 4]; B=[2 4];
» try, X=A\B, catch, lasterr, X=A\B', end
ans =
Error using ==> \
Matrix dimensions must agree.
X =
    0
    1
```

Складений оператор ітераційного циклу з передумовою while…end має формат:

```
while  LogExpression   % заголовок циклу
    % операторы тіла циклу:
     Stat1; State2; … StateN;
end % кінець циклу
```

Він організує виконання операторів stat, що складають тіло циклу, до тих пір, поки значення логічного виразу LogExpression є істиною (1). Вихід із циклу відбувається, коли значення цього виразу стає хибним (0). Оператори тіла циклу повинні змінювати значення хоча б однієї змінної, що входять до логічного виразу в заголовку циклу, або отримувати у своєму складі умовний оператор з оператором break у гілці «так».

Приклад 5.3. Знайти мінімальне ціле число n, що задовольняє умові $n!>10^{12}$:

```
M = 1e12; n = 1; F = 1;
while F<= M,  n = n+1;  F = F*n;  end,  F, n
```

В результаті отримуємо:

```
F =
  1.3077e+12
n =
   15
```

Приклад 5.4. Скласти програму, яка в залежності від вибору опції, виконаного мишею за допомогою сформованого меню, виконує одну з 3-х програм, а при виборі опції Exit завершує виконання програми.

Розв‘язання цієї задачі має вигляд

```
while 1
    k = menu (‘Оберіть файл для виконання’, ‘File 1’, ‘File 2’, ‘File 3’, ‘Exit’)
    switch  k
          case 1, FileName1
          case 2, FileName2
          case 3, FileName3
          otherwise  break
    end
end
```

Оператор обчислення рядку символів як виразу eval (від англ. *evaluate*) має формат

eval (String)

Наприклад,

```
» eval ('sqrt(2)')
ans =
   1.4142
```

Оператор виконання функції, ім‘я якої задано рядком символів, feval має формат

feval (String, Arg1, Arg2, …, ArgN)

Наприклад,

```
» feval('sin',pi/2)
ans =
    1
```

Приклади, які доведуть доцільність використання операторів eval та feval, які за своєю конструкцією більш схожі на функції, будуть наведені пізніше.

5.4 Основні правила створення і виконання *m*-функцій

M-функції застосовуються для програмування типових алгоритмів, які показують, як із формальних вхідних параметрів утворити вихідні, або як використати вхідні параметри для побудови графіків тощо. Тобто *m*-функція уявляє собою нібито розгорнуту формулу, у якій формальні вхідні параметри є тільки позначеннями, тобто умовними іменами аргументів.

Щоб отримати конкретний результат за допомогою *m*-функції треба виконати заміну формальних вхідних аргументів фактичними, які при зверненні до функції повинні мати конкретні значення, і виконати усі операції з цими параметрами.

Для створення *m*-функції треба клацнути мишею по віртуальній кнопці *New* у командному вікні MATLAB та обрати опцію *Function*. В результаті на екрані з'явиться вікно текстового редактора зі сформованим у ньому форматом функції, зображене на рис. 5.3.

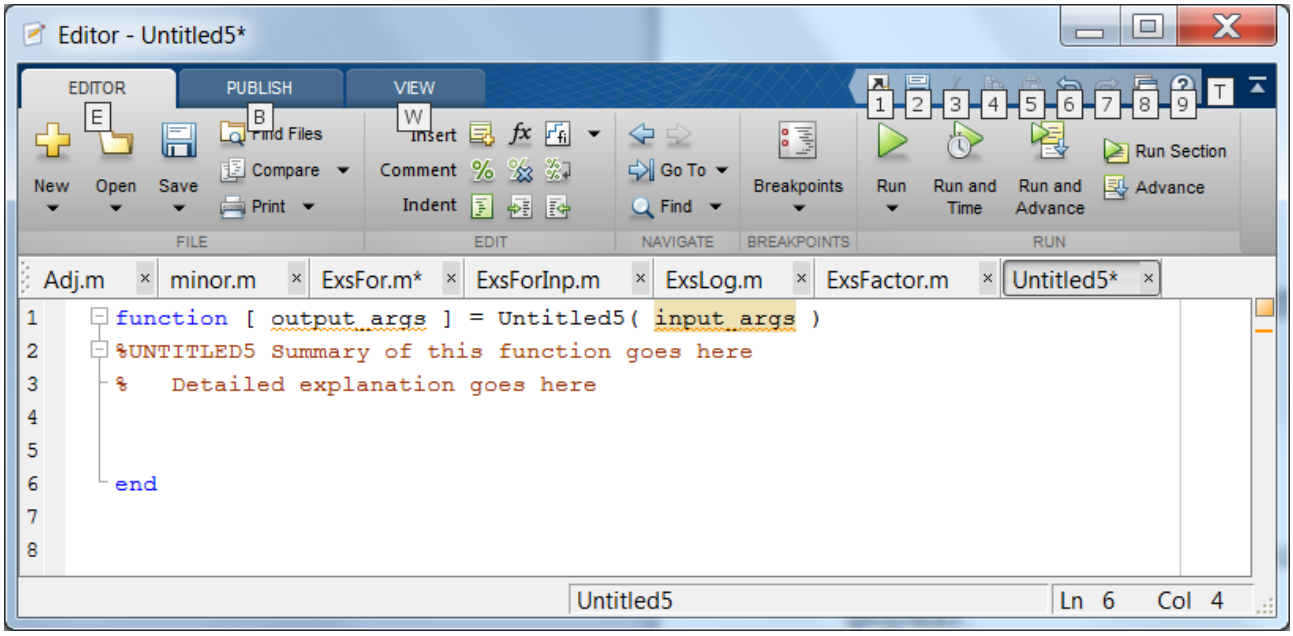


Рис. 5.3. Вікно текстового редактору для створення *m*-функції

Як бачимо, *m*-функція має формат

```
% Заголовок функції:
function [ List of ResultsNames ] = FunName (List of InputArgsNames )
% Необов'язкові коментарі:
% Contents String (global information of this function)
% Detailed information of this function
…
%
        % оператори тіла функції
         Stat1; Stat2; …; StatN;
end  % кінець функції
```

Імена змінних у списку результатів виконання функції List of ResultsNames (в *MATLAB* їх називають вихідними аргументами – *output arguments*) та у

списку аргументів функції List of InputArgsNames (в *MATLAB* їх називають вхідними аргументами – *input arguments*) відокремлюються одне від одного комами «,».

Оператори та коментарі в *m*-функціях вводяться за тими ж правилами, які використовуються при створенні *script*-файлів (див. пункт 5.1.1). Зберігати *m*-функцію треба обов‘язково у файлі, ім‘я якого співпадає з іменем функції FunName. Такий файл автоматично отримує розширення «.*m*».

На відміну від *script*-файлу *m*-функцію не можна виконати командою run. Для виконання функції до неї необхідно звернутися з будь-якого програмного файлу або з командного рядка по імені та списку фактичних аргументів у круглих дужках.

Якщо функція не потребує наявності вхідних або вихідних параметрів, вони в заголовку функції позначаються у вигляді пустого вектору [ ], або зовсім опускаються.

Функції в MATLAB можуть використовуватися з різною кількістю вхідних і вихідних параметрів, а також для різнотипних параметрів. Наприклад, розглянемо різні варіанти використання базової функції математичного аналізу min:

```
>>a = -45; b = 12;
>>c = min (a,b)
c =
  -45
>>X = [2 7 -5 3];  Y= [15 6 22 -1];
>>Xmin = min(X)
Xmin =
   -5
>> [Xmin,Imin] = min(X)
Xmin =
   -5
Imin =
    3
>>Z = min (X,Y)
Z =
    2    6    -5    -1
>>A = [1 3 0; 4 2 8], B = [5 1 2; -3 7 4]
```

```
A =
    1    3    0
    4    2    8
B =
    5    1    2
   -3    7    4
>>Amin = min(A)
Amin =
    1    2    0
>> [Amin,imin] = min(A)
Amin =
    1    2    0
imin =
    1    2    1
>>A_min = min(min(A))
A_min =
    0
>>C=min(A,B)
C =
    1    1    0
   -3    2    4
```

Щоб створити такі функції, в них застосовують стандартні *m*-функції nargin та nargout, значеннями яких є кількість вхідних та вихідних фактичних аргументів відповідно.

Для того, щоб користувач правильно формував фактичні вхідні аргументи, в тілі функції повинні бути присутніми оператори, що аналізують типи та розміри цих параметрів, та при наявності помилок припиняють виконання програми з виведенням повідомлення про помилку.

Приклад 5.5. Скласти функцію maxi(x), яка в заданому векторі x визначає максимальний елемент та його позицію.

```
function [xm,im] = maxi(x)
    [m,n]=size(x);
    if (m~=1)&(n~=1)  % Якщо x - матриця
      error('Помилка в розмірності аргументу');
    end
    k = length(x);
    if k==1, xm=x; im=1; break; end
    xm=x(1);
    for i=2:k
      if x(i)>xm  xm = x(i); end
```

```
    end
    if nargout==2     % Два вихідних параметри
      im = find(x==xm);
    end
end
```

5.5 Завдання

1. Виконати приклади, наведені в теоретичній частині.
2. Скласти функцію користувача згідно з завданням, наведеним у табл. 5.1. Застосувати її для розв‘язання декількох довільних тестових задач. Перевірити правильність отриманого результату.

Таблиця 5.1

№	Задача *m*-функції	Перевірка
1	Розрахувати матричний добуток $\mathbf{C}=\mathbf{AB}$ матриці $\mathbf{A}$ розміром $m\times k$ та матиці $\mathbf{B}$ розміром $k\times n$ за формулою (2.3).	C=A*B
2	Для квадратної матриці $\mathbf{A}$ розрахувати $\mathbf{B}=\mathbf{A}^2$ як матричний добуток $\mathbf{AA}$.	B = A^2
3	Для квадратної матриці $\mathbf{A}$ розрахувати $\mathbf{B}=\mathbf{A}^k$ з використанням операції *.	B = A^k
4	Для квадратної матриці $\mathbf{A}$ розміром $n\times n$ та вектору-стовпчику $\mathbf{B}$ розміром $n\times 1$ розрахувати матрицю керованості $\mathbf{Q}_c=[\mathbf{B}\ \mathbf{AB}\ \mathbf{A}^2\mathbf{B}\ldots\mathbf{A}^{n-1}\mathbf{B}]$ з використанням операції *.	Qc = ctrb (A,B)
5	Для квадратної матриці $\mathbf{A}$ розміром $n\times n$ та вектору-рядку $\mathbf{C}$ розміром $1\times n$ розрахувати матрицю керованості $\mathbf{Q}_o=[\mathbf{C}\ \mathbf{CA}\ \mathbf{CA}^2\ \ldots\mathbf{CA}^{n-1}]^T$ з використанням операції *.	Qo = obsv (A,B)
6	Розрахувати визначник квадратної матриці $\Delta=\lvert\mathbf{A}\rvert$ з використанням прямого ходу методу Гауса	d = det (A)
7	Розв‘язати систему лінійних алгебраїчних рівнянь методом Гауса	X = A\B
8	Розв‘язати систему лінійних алгебраїчних рівнянь методом Крамера з використанням функції	X = A\B
9	Для квадратної матриці $\mathbf{A}$ розрахувати союзну матрицю $\tilde{\mathbf{A}}$.	As=(inv(A)*det(A))’
10	Для квадратної матриці $\mathbf{A}$ розрахувати приєднану матрицю $Adj(\mathbf{A})$.	Ad=inv(A)*det(A)
11	Для квадратної матриці $\mathbf{A}$ розрахувати обернену матрицю $\mathbf{A}^{-1}$ за формулою (2.13).	Ai = inv (A)

		Продовження таблиці 5.1
12	Розрахувати обернену матрицю шляхом розв‘язання систем лінійних рівнянь для кожних із стовпчиків матриць $\mathbf{A}^{-1}$ та одиничної діагональної матриці $\mathbf{I}$ матричного рівняння $\mathbf{A}^{-1}\mathbf{A}=\mathbf{I}$.	Ai = inv (A)
13	Розрахувати середнє квадратичне значення вектору за формулою (2.20).	s = std (x)
14	Розрахувати дисперсію вектору за формулою (2.21).	D = var (x)
15	Розрахувати середнє геометричне значення вектору за формулою (2.22).	
16	Розрахувати k-ті прямі різниці вектору $\mathbf{V}$.	dVk = diff (V,k)
17	Розрахувати суму квадратичних відхилень двох векторів однакової довжини $F=\sum_{i=1}^{n}(y_i-x_i)^2$	
18	Розрахувати визначений інтеграл табличної функції методом трапецій за формулою $Z=\frac{1}{2}\sum_{i=1}^{n-1}(y_{i+1}+y_i)(x_{i+1}-x_i)$	Z = trapz (x,y)

5.6 Методичні вказівки та рекомендації

1. Не давайте своїм функціям імена, що збігаються з зарезервованими іменами пакету *MATLAB.*
2. Організуйте у функціях перевірку розмірів вхідних параметрів. У разі невідповідності їх умовам задачі організуйте виведення в командне вікно повідомлення про помилку за допомогою функції *error*.
3. Для розв‘язання тестових задач складіть *script-file*.

5.7 Контрольні питання та завдання

1. Як здійснити діалогове введення даних?
2. Як програмуються в *MATLAB* розгалужені процеси?
3. Як програмуються в *MATLAB* ітераційні цикли?
4. Як програмуються в *MATLAB* цикли з відомою кількістю повторень?
5. Чим відрізняються файли-сценарії від файлів-функцій?

Лабораторна робота №6
СПЕЦІАЛЬНА ГРАФІКА

Мета роботи: ознайомитися з функціями спеціальної графіки; навчитися формувати стовпчикові та кругові діаграми, ступінчаті графіки та графіки решітчастих функцій, гістограми; закріпити навички роботи з базовими функціями математичного аналізу.

6.1 Характеристика засобів спеціальної графіки пакету *MATLAB*

До найбільш часто застосованих засобів спеціальної графіки (див. help specgraph) відносяться

bar – Bar graph (стовпчикова діаграма);
barh – Horizontal bar graph.
stairs – Stairstep plot (ступінчатий графік);
stem – Discrete sequence or "stem" plot (решітчаста функція);
scatter – Scatter plot (точкова діаграма);
area – Filled area plot;
fill – Filled 2-D polygons;
pie – Pie chart (кругова діаграма);
hist – Histogram (гістограма);
rose – Angle histogram plot (кутова гістограма);
compass – Compass plot (векторна діаграма);
fplot – Plot function;
ezplot – Easy to use function plotter;
ezpolar – Easy to use polar coordinate plotter.

Стовпчикові діаграми досить часто застосовують для наочного порівняння деяких кількісних показників за роками, наприклад, народжуваність, продуктивність підприємств, кількість землетрусів, тощо. Іще їх зручно використовувати при відображенні випадкових процесів.

Кругові діаграми найчастіше застосовують для наочного відображення деяких показників у процентах, коли площа повного кола приймається за 100%, а площі секторів, пофарбовані у різні кольори, відображають частки цілого.

Ступінчаті графіки застосовують при зображенні цифрових сигналів, коли інформація дискретно змінюється у дискретні моменти часу.

Решітчасті функції відображають властивості ідеальних імпульсних сигналів.

Гістограми будують під час обробки результатів вимірювань. Для цього вектор вимірювань сортирують, діапазон результатів ділять на декілька рівних інтервалів і на кожному з них підраховують кількість вимірювань. Залежність кількості вимірювань на кожному інтервалі від діапазонів цих інтервалів, подану у вигляді стовпчикової діаграми, називають гістограмою. Гістограма дає уявлення про найбільш достовірний діапазон виміряних значень.

Зображення векторів комплексних чисел у полярній системі координат можна отримати функцією compass(z) або функцією compass(x,y), де x – вектор дійсних частин, а y – вектор уявних частин комплексних чисел.

Точкові діаграми використовують здебільш для побудови діаграм розсіювання, які ще називають діаграмами розкиду даних або скаттер-діаграмами.

Основні види спеціальних графіків, побудовані при виконанні програми

```
clc, close all
x=0:1:10;
y=(x-4).^2.+2;
for i=1:8
   subplot(4,2,i)
   switch i
      case 1, bar(x,y), xlim([0 10]), title('bar(x,y), xlim([0 10])')
      case 2, barh(x,y),ylim([0 10]), title('barh(x,y),ylim([0 10])')
      case 3, stairs(x,y), title('stairs(x,y)')
      case 4, pie(sqrt(y(6:10))), title('pie(sqrt(y(6:10)))')
      case 5, stem(x,y), title('stem(x,y)')
      case 6, scatter(x,y), title('scatter(x,y)')
      case 7, area(x,y), title('area(x,y)')
      case 8, fill(x,y,'g'), title('fill(x,y)')
   end
end
```

показані на рис. 6.1.

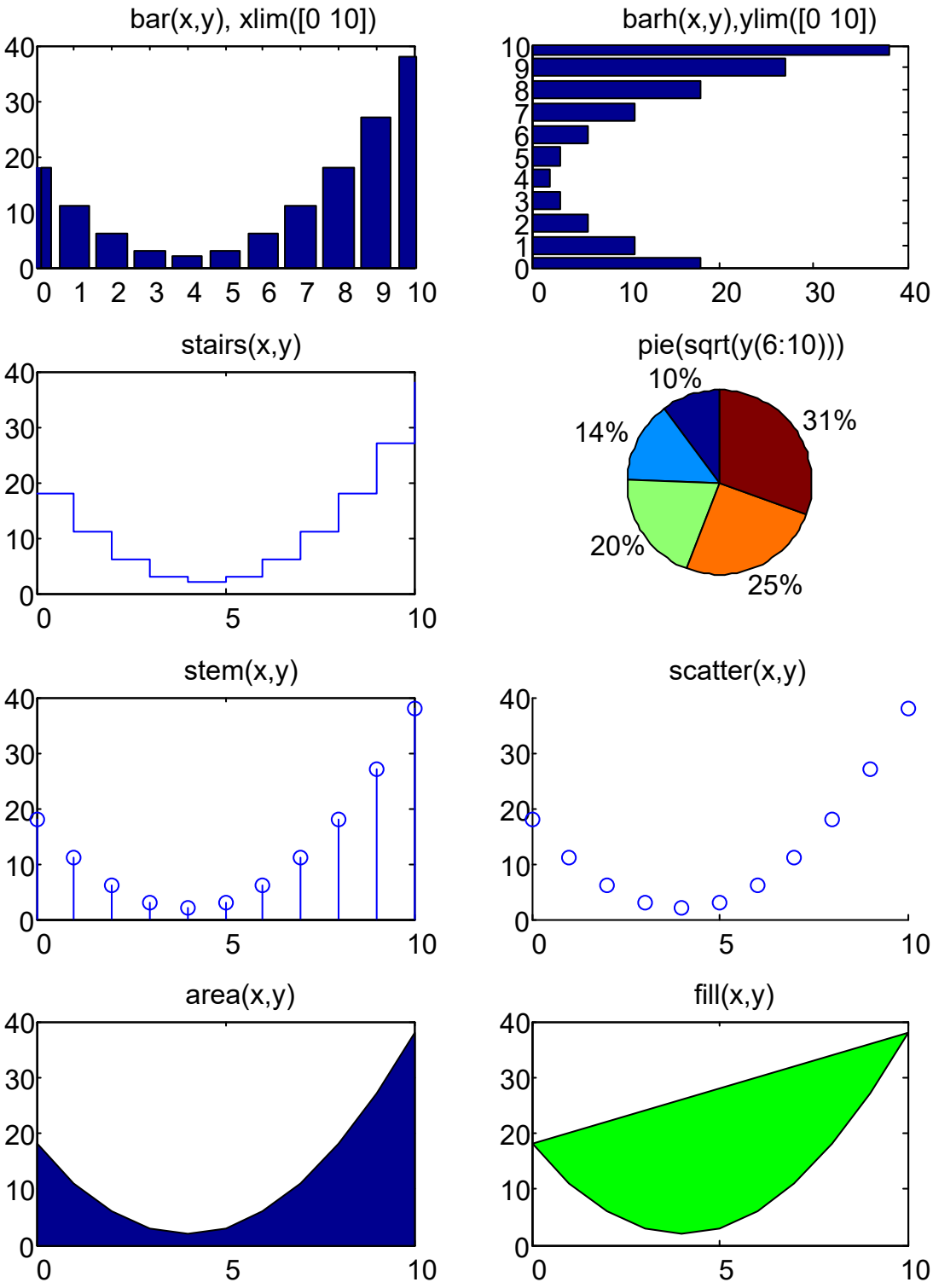


Рис. 6.1. Види спеціальних графіків

Якщо функція, графік якої треба побудувати, описана у вбудованій *m*-функції або в *m*-функції користувача, то для її побудови можна використати функції fplot, ezplot та ezpolar.

Функцію fplot зазвичай застосовують у форматах fplot (fun, limits) та

fplot (fun, limits, Style).

Функція ezplot має більше варіацій:

- ezplot (fun) будує графік функції fun(x) на інтервалі $-2\pi<x<2\pi$;
- ezplot (fun, [xmin, xmax]) будує графік функції fun(x) на інтервалі $x_{\min}<x<x_{\max}$;
- ezplot (fun2) будує графік функції fun2(x,y)=0 на інтервалах $-2\pi<x<2\pi$, $-2\pi<y<2\pi$;
- ezplot (fun2, [xmin, xmax, ymin, ymax]) будує графік функції fun2(x,y)=0 на інтервалах $x_{\min}<x<x_{\max}$, $y_{\min}<y<y_{\max}$;
- ezplot (funx, funy) будує параметрично визначений графік функції funy(t)(funx(t)) в діапазоні $0<t<2\pi$
- ezplot (funx, funy [tmin, tmax]) будує параметрично визначений графік функції funy(t)(funx(t)) в діапазоні $t_{\min}<t<t_{\max}$.

Функція ezpolar будує в полярних координатах графік функції fun(fi). Вона застосовується у варіантах ezpolar (fun) та ezpolar (fun,[a,b]). За замовчанням $0<\varphi<2\pi$.

Параметр fun у цих функціях може задаватися декількома способами:

- виразом ‘Expression’ або ім‘ям функції ‘FunName’ у вигляді рядка символів;
- посиланням на функцію @FunName;
- анонімною функцією @(x) Expression.

Для прикладу наведемо програму і результат її виконання (рис. 6.2).

```
for i=1:6
   subplot(3,2,i)
   switch i
      case 1, fplot(@tanh,[-2 2]), grid
      case 2, fplot('myfun_sg',[-20 20]), grid
      case 3, ezplot(@cos), grid
      case 4, ezplot('(x-4)^2',[0 10]), grid
      case 5, ezpolar(@(t) 1+cos(t))
      case 6, ezpolar('sin(7*fi/4)', [0 8*pi])
   end
end
```

```
function y=myfun_sg(x)
    y(:,1) = 200*sin(x(:))./x(:);
    y(:,2) = x(:).^2;
end
```

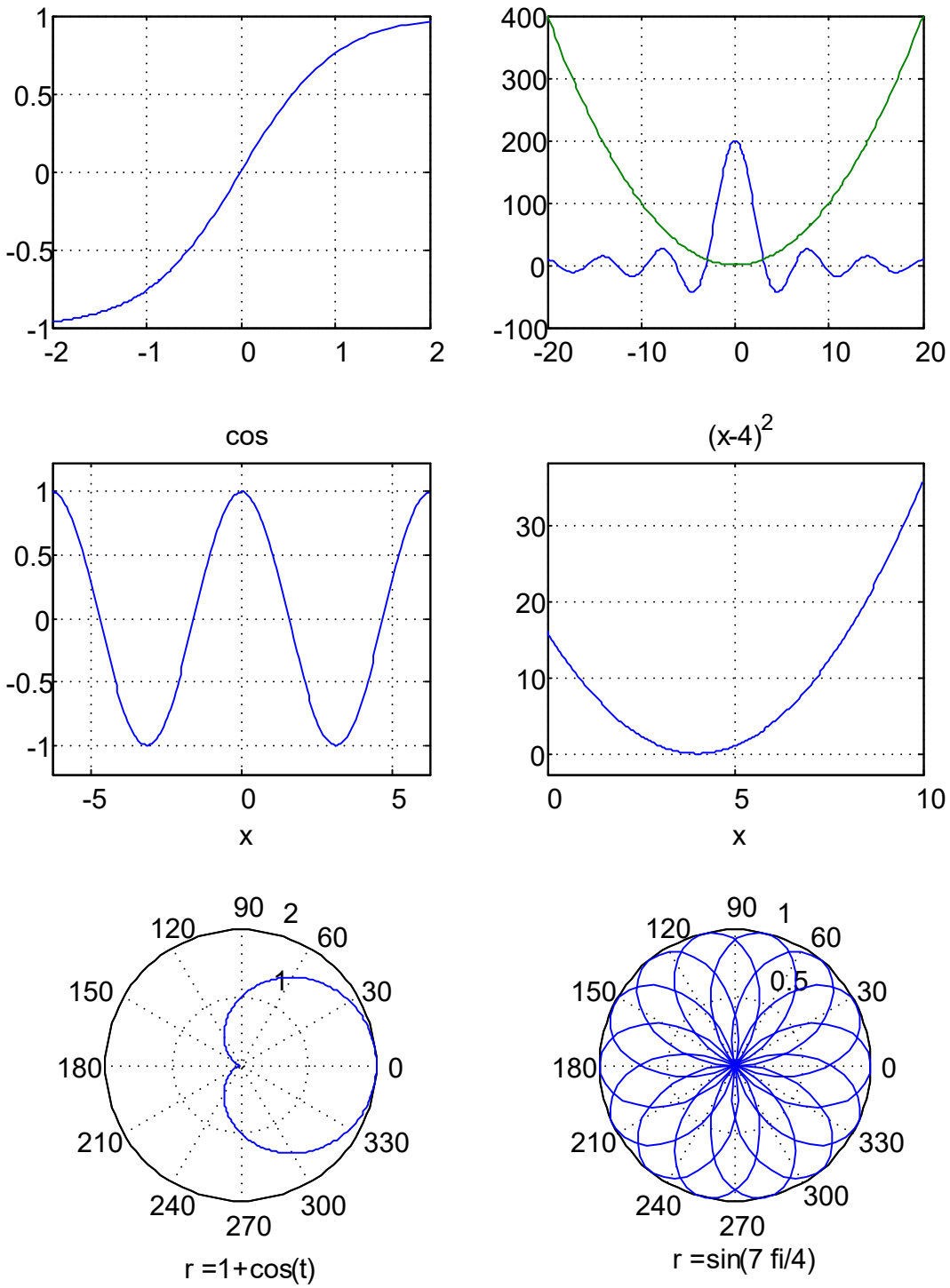


Рис. 6.2. Приклади використання функцій fplot, ezplot та ezpolar

6.2 Завдання

1. Сформувати два вектори-рядки випадкових чисел із 100 елементів: з рівномірним розподіленням елементів на інтервалі [*a,b*] та нормальним розподіленням елементів за замовчанням.
2. Відсортувати їх за збільшенням елементів.

3. Відобразити у чотирьох підвікнах одного графічного вікна графіки вихідних та відсортованих векторів у вигляді стовпчикових діаграм (вихідні масиви) та у вигляді решітчастих функцій (відсортовані масиви); зробіть висновки щодо різниці між рівномірним та нормальним розподіленнями випадкових чисел.
4. Знайти для відсортованих масивів середнє арифметичне та серединне значення.
5. Створити із кожного *i*-го елементу відсортованих масивів нові вектори та побудуйте ступінчаті графіки створених масивів.
6. Побудувати для скороченого відсортованого вектору з рівномірним розподіленням кругову діаграму.
7. Задати 2 довільних комплексних числа. Знайти їх суму, різницю, доданок та частку від ділення. Зобразити задані та отримані в результаті виконаних операцій числа на комплексній площині у вигляді векторів. Пояснити результат.

6.3 Методичні вказівки та рекомендації

1. Для формування вектору випадкових чисел із 100 елементів: з рівномірним розподіленням елементів на інтервалі [*a,b*] (пункт 1 завдання) скористайтеся оператором

```
X=a+(b-a)*rand(1,100)
```

2. Перед виконанням пункту 4 завдання розберіться у різниці між середнім і серединним значенням одномірного масиву та виконайте команди help mean і help median.
3. Для виконання пункту 5 завдання застосуйте операцію

```
Y=Xs(i:i:100)
```

6.4 Контрольні питання та завдання

1. Як візуалізувати графік одномірний масив даних у вигляді стовпчикової діаграми, кругової решітчастої, точкової та ступінчатої функції?

2. Поясніть різницю між функціями area та fill.
3. Для чого застосовують функції? Яка між ними різниця?
4. Якими способами можна визначати функцію для операцій fplot, ezplot та ezpolar? Наведіть приклади.

Лабораторна робота №7
ПОБУДОВА ЧАСТОТНИХ ХАРАКТЕРИСТИК

Мета роботи: ознайомитися з операціями над комплексними числами, з видами частотних характеристик та з методикою їх побудови.

7.1 Визначення частотних характеристик

Частотні характеристики будуються на підставі функцій комплексної змінної $W(j\omega)$, уявною частиною якої є частота. Таку функцію завжди можна перетворити до вигляду

$$W(j\omega) = W_{re}(\omega) + jW_{im}(\omega) \tag{7.1}$$

або

$$W(j\omega) = A(\omega)e^{j\varphi(\omega)}. \tag{7.2}$$

Складові формул (7.1) та (7.2) пов‘язані між собою співвідношеннями:

$$A = \sqrt{W_{re}^2 + W_{im}^2}, \qquad \varphi = \operatorname{arctg}(W_{im}/W_{re}); \tag{7.3}$$

$$W_{re} = A\cos\varphi, \qquad W_{im} = A\sin\varphi. \tag{7.4}$$

При частотному аналізі сигналів та динамічних систем застосовують такі частотні характеристики:

$W_{re}(\omega)$ – ***дійсна частотна характеристика*** (***ДЧХ***)***;***

$W_{im}(\omega)$ – ***уявна частотна характеристика*** (***УЧХ***);

$A(\omega)$ – ***амплітудно- частотна характеристика*** (***АЧХ***);

$\varphi(\omega)$ – ***фазо- частотна характеристика*** (***ФЧХ***);

$\lg A(\lg\omega)$ – ***логарифмічна амплітудно- частотна характеристика*** (***ЛАЧХ***);

$\varphi(\lg\omega)$ – ***фазо- частотна характеристика*** (***ЛФЧХ***);

$A(\varphi)$ – ***амплітудно-фазова частотна характеристика*** (***АФЧХ***).

ЛАЧХ та ЛФЧХ називають також ***діаграмами Боде*** (***Bode Plots***).

АФЧХ називають ***годографом або діаграмою Найквіста*** (***Nyquist Plot***).

Для побудови ДЧХ, УЧХ, АЧХ і ФЧХ частоту, як вектор абсцис частотних функцій, зручно задавати оператором

```
omega = linspace (0, omega_k);
```

Потім за заданим аналітичним виразом і його параметрами розраховують функцію комплексної змінної W.

7.2 Розрахунок та побудова частотних характеристик

Для розрахунку векторів ординат відповідних частотних функцій треба виконати такі оператори присвоєння:

```
Wre = real (W)
Wim = imag (W)
A = abs (W)
fi = angle (W)
```

За бажанням фазу можна перетворити з радіан на градуси:

```
fi_d=fi*180/pi
```

За розрахованими даними перелічені вище частотні характеристики будують оператором plot, наприклад, `plot (omega, A)`.

Для побудови діаграм Боде частоту треба задавати оператором

```
omegal = logspace (d0, dk);
```

для побудови ЛАЧХ застосувати оператор

```
loglog (omegal, A)
```

а для побудови ЛФЧХ – оператор

```
semilogx (omegal ,fi_d)
```

Для побудови годографа Найквіста можна будувати як у полярних, так і у Декартових координатах. У першому випадку застосовують оператор

```
polar (fi, A}
```

а у другому –

```
plot (Wre, Wim)
```

Слід відзначити, що при побудові АФЧХ досить складно підібрати такі значення частот, щоб однаково добре було відображено усі ділянки годографа.

7.3 Завдання

Побудувати частотні характеристики динамічної системи за її передавальною функцією, заданою в табл. 7.1.

Таблиця 7.1

Вар.	Передавальна функція	Параметри
1	$W(p)=\dfrac{k}{T_2 T_1 p^2+T_1 p+1}$	$k=10,\ T_1=1,\ T_2=0.5$
2		$k=5,\ T_1=1,\ T_2=1$
3	$W(p)=\dfrac{T_3 p}{T_2 T_1 p^2+T_1 p+1}$	$T_1=1,\ T_2=2,\ T_3=0.5$
4		$T_1=1,\ T_2=1,\ T_3=1$
5	$W(p)=\dfrac{k(T_3 p+1)}{T_2 T_1 p^2+T_1 p+1}$	$k=10,\ T_1=1,\ T_2=2,\ T_3=1$
6		$k=2,\ T_1=1,\ T_2=1,\ T_3=2$
7	$W(p)=\dfrac{k(T_4 T_3 p^2+T_3 p+1)}{T_2 T_1 p^2+T_1 p+1}$	$k=0.5,\ T_1=1,\ T_2=2,\ T_3=0.5,\ T_4=1$
8		$k=1,\ T_1=1,\ T_2=1,\ T_3=2,\ T_4=1$
9	$W(p)=\dfrac{T_3^2 p^2}{T_2 T_1 p^2+T_1 p+1}$	$k=0.25,\ T_1=1,\ T_2=2,\ T_3=0.75$
10		$k=2,\ T_1=1,\ T_2=1.5,\ T_3=2$
11	$W(p)=k\dfrac{(T_3^2 p^2+1)}{T_2 T_1 p^2+T_1 p+1}$	$k=4,\ T_1=1,\ T_2=0.8,\ T_3=0.5$
12		$k=0.4,\ T_1=1,\ T_2=2,\ T_3=0.5$
13	$W(p)=\dfrac{T_3 p}{T_1^2 p^2+1}$	$T_1=1,\ T_3=1$

7.4 Методичні вказівки та рекомендації

1. Об‘єднайте характеристики у такі групи: 1) ДЧХ та УЧХ; 2) АЧХ та ФЧХ; 3) ЛАЧХ і ЛФЧХ. Графіки першої групи побудуйте в одній системі координат окремого графічного вікна, графіки другої групи – у різних

підвікнах одного графічного вікна, розташованих в рядок, а характеристики третьої групи – у різних підвікнах одного графічного вікна, розташованих одне під іншим.

2. АФЧХ побудуйте у полярних та Декартових координатах.
3. На усі графіки нанесіть координатні сітки, титули, позначення осей та характеристик.

7.5 Контрольні запитання та завдання

1. Що таке частотні характеристики?
2. Які частотні характеристики ви знаєте?
3. Як отримати частотні характеристики із передавальної функції?
4. Які форми представлення комплексних чисел ви знаєте?
5. Як визначити діапазон зміни частоти для побудови *Bode*-діаграм?
6. В яких системах координат можна будувати діаграми Найквіста?

Лабораторна робота 8
ТРИВИМІРНА ГРАФІКА

Ціль роботи – ознайомитися з функціями тривимірної графіки; навчитися будувати графіки функції двох змінних у вигляді кривих, сіток та поверхонь.

8.1 Загальні відомості

3-вимірні графіки у середовищі ML можна будувати за допомогою функцій plot3 (x,y,z), mesh (X,Y,Z) та surf (X,Y,Z) (див. help graph3D).

plot3 (x,y,z) будує у 3-вимірній системі координат криву, з'єднуючи між собою точки з координатами (x_i, y_i, z_i) для $i = 1, 2, ..., n$ відрізками прямих. Вектори x,y,z повинні мати однаковий розмір, наприклад, при виконанні операторів

```
t = linspace (0,8*pi,160);
plot3 (sin(t), cos(t), sqrt(t)),  grid on
xlabel ('sin(t)'),  ylabel ('cos(t)'),  zlabel ('sqrt(t)'),
title ('Спираль')
```

отримуємо графік спіралі, зображений на рис. 8.1.

Функції mesh (X,Y,Z), surf (X,Y,Z) і surfl (X,Y,Z) мають параметрами не вектори, а матриці і будують не лінії, а поверхні.

Щоб скористатися цими функціями, спочатку створюють вектори, що задають діапазони зміни значень абсцис і ординат, наприклад,

$$\mathbf{x} = [x_1 \quad x_2 \quad ... \quad x_n], \qquad \mathbf{y} = [y_1 \quad y_2 \quad ... \quad y_m]; \tag{8.1}$$

потім створюють 2 матриці розміром $m \times n$: матрицю $\mathbf{X}$ із m рядків вектору $\mathbf{x}$ і матрицю $\mathbf{Y}$ із n стовпчиків вектору $\mathbf{y}$:

$$\mathbf{X} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \\ x_1 & x_2 & \dots & x_n \\ \vdots & \vdots & \ddots & \vdots \\ x_1 & x_2 & \dots & x_n \end{bmatrix}, \qquad \mathbf{Y} = \begin{bmatrix} y_1 & y_1 & \dots & y_1 \\ y_2 & y_2 & \dots & y_2 \\ \vdots & \vdots & \ddots & \vdots \\ y_m & y_m & \dots & y_m \end{bmatrix}. \tag{8.2}$$

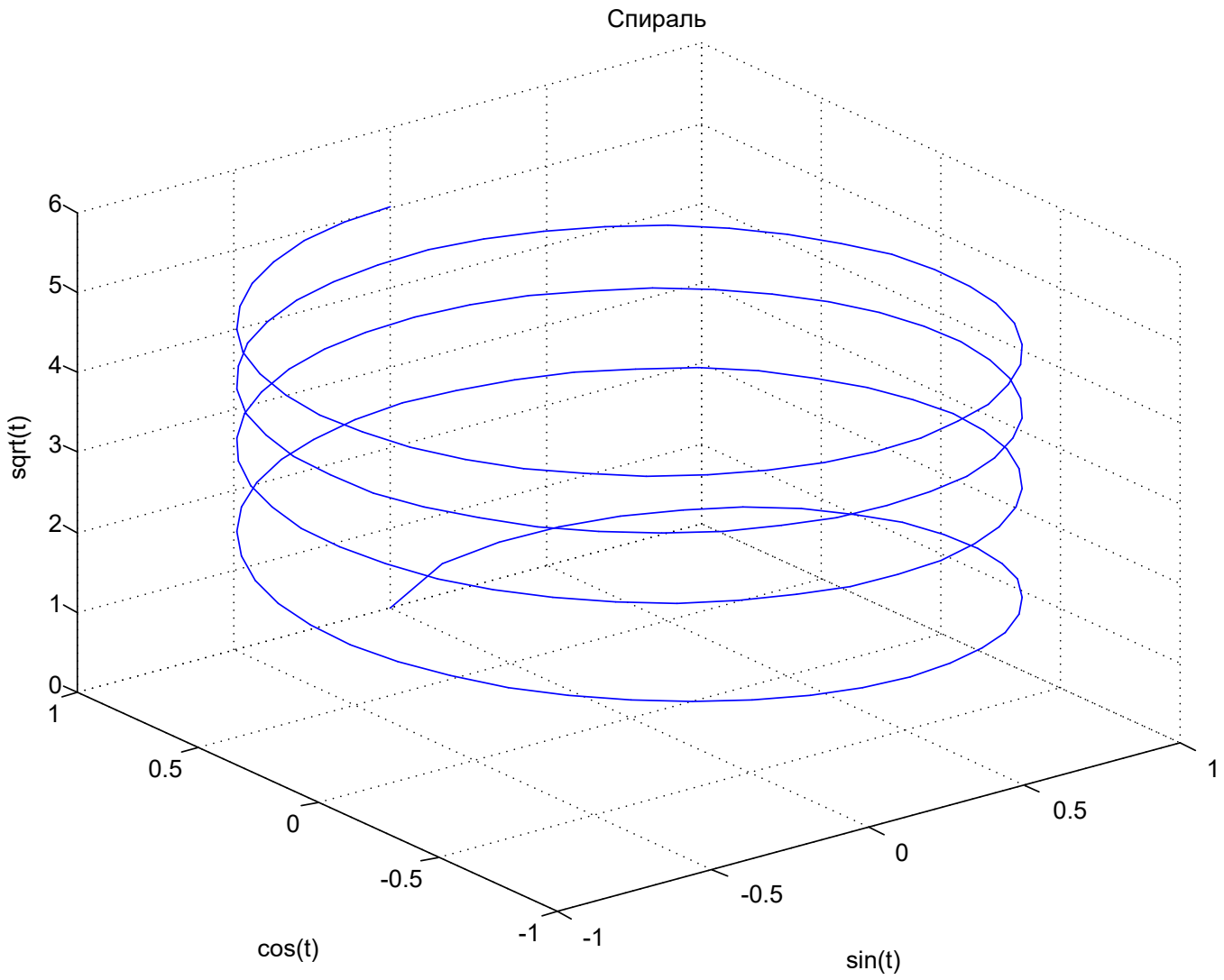


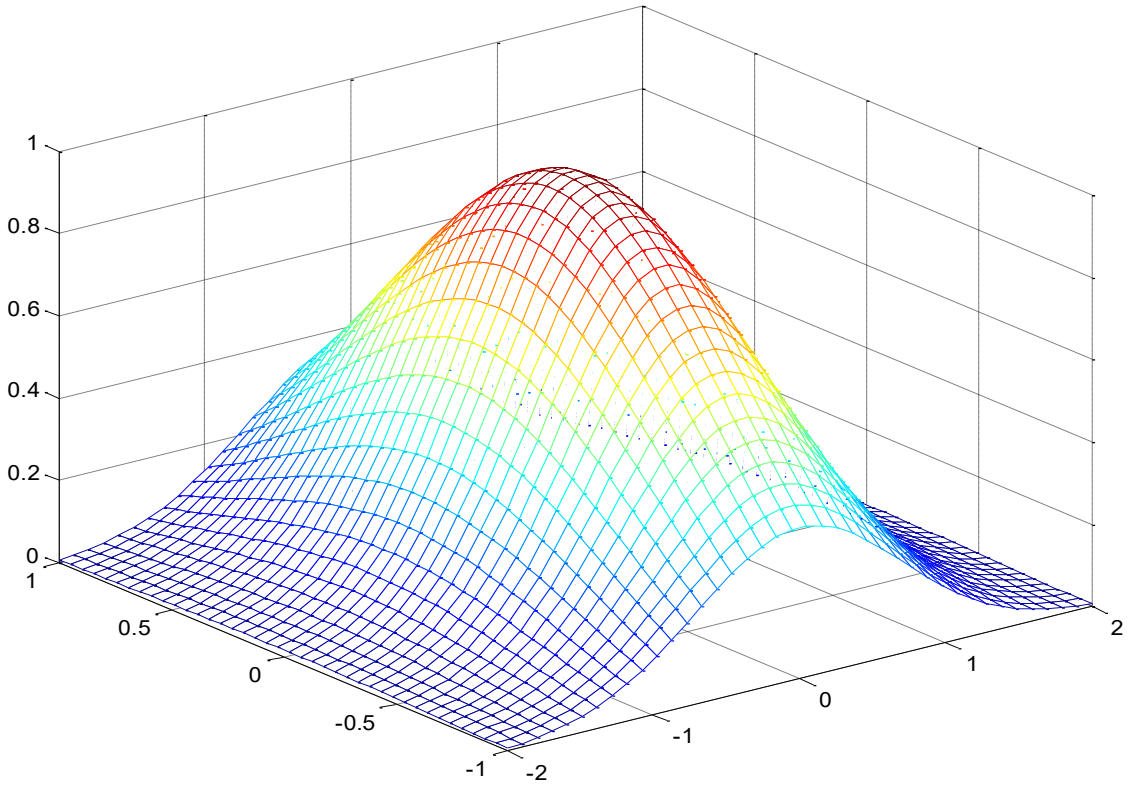
Рис. 8.1. Тривимірний графік спіралі

Після цього розраховують матрицю Z і будують графіки. Наприклад, при виконанні програми

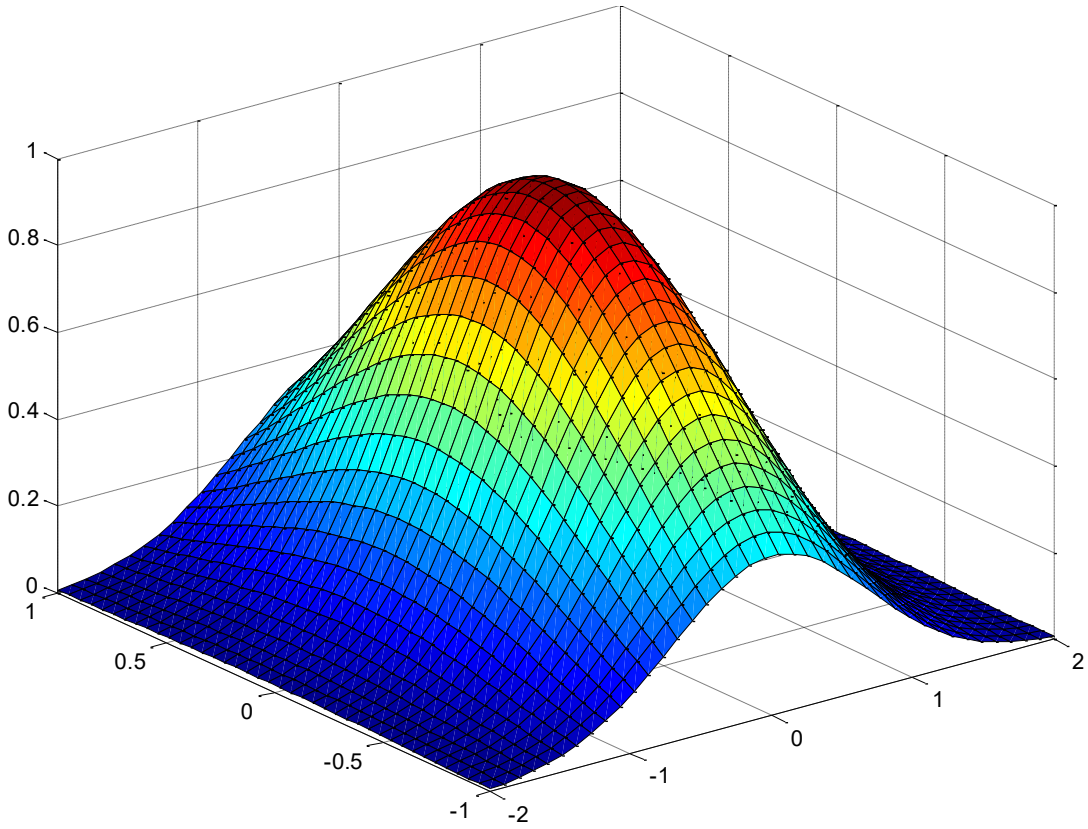
```
x = linspace (-2,2,40);
y = linspace (-1,1,40);
[X,Y] = meshgrid (x,y);
Z = exp(-X.^2-Y.^2);
mesh (X,Y,Z),  grid on
figure,  surf (X,Y,Z),  grid on
```

отримуємо графіки, подані на рис.8.2. Вони демонструють різницю між двома досліджуваними графічними функціями.

Функція mesh будує каркасно-ребристу (*wireframe mesh*) поверхню, а функція surf – непрозору сітчасту поверхню.



а)



б)

Рис. 8.2. Графіки залежності $z = e^{-x^2-y^2}$, побудовані функціями mesh (а) та surf (б)

Для кращого сприйняття "об‘ємності" зображення різних ребер або клітин автоматично фарбується у різні кольори. Крім того, здійснюється видалення невидимих ліній. Щоб створити ефект прозорості каркасно-ребристої поверхні, застосовують команду hidden off. Функція surfl зафарбовує сітчасту поверхню більш природно.

Функція surf може застосовуватися в режимі інтерполяції тіні на гранях графіка shading interp. За замовчанням використовується стиль shading faceted, можна також стилем shading flat.

Також існує можливість міняти кольорову гаму (палітру, карту кольорів) графіка за допомогою функції

colormap (MapName)

Перелік палітр можна подивитися за допомогою команди

help graph3d

Color maps.
hsv - Hue-saturation-value color map (кольори веселки);
hot - Black-red-yellow-white color map (чергування чорного, червоного, жовтого та білого кольорів);
gray - Linear gray-scale color map (відтінки сірого);
bone - Gray-scale with tinge of blue color map (сірі кольори з відтінком голубого);
copper - Linear copper-tone color map (відтінки міді);
pink - Pastel shades of pink color map (розові кольори);
white - All white color map (біла палітра);
flag - Alternating red, white, blue, and black color map (контрастне чергування червоного, білого, синього та чорного кольорів);
lines - Color map with the line colors;
Colorcube - Enhanced color-cube color map;
vga - Windows colormap for 16 colors;
jet - Variant of HSV;
prism - Prism color map;
cool - Shades of cyan and magenta color map (відтінки голубого та фіолетового кольорів);
autumn - Shades of red and yellow color map (відтінки червоного та жовтого кольорів);
spring - Shades of magenta and yellow color map (відтінки пурпурного та жовтого кольорів);
winter - Shades of blue and green color map (відтінки голубого та зеленого);
summer - Shades of green and yellow color map (відтінки зеленого та жовтого).

За замовчанням використовується палітра hsv, у якій кольори плавно змінюються від синього до червоного.

Кожна палітра уявляє собою матрицю інтенсивності rgb-кольорів розміром 64×3, яку можна побачити після введення імені палітри. Графічна інтерпретація інтенсивності трьох змішуваних кольорів виводиться командою

rgbplot (MapName)

8.2 Завдання

Побудувати 3-вимірні графіки фігур за даними табл. 8.1 різними стилями та з використанням декількох різних палітр.

Таблиця 8.1

Вар	Рівняння	Назва поверхні	α	β
1	$\begin{cases} x = \cos\alpha \cdot \cos\beta \\ y = \sin\alpha \cdot \cos\beta \\ z = \sin\beta \end{cases}$	Сфера	-π ÷π	-π/2 ÷π/2
2	$\begin{cases} x = \cos\alpha \cdot (\cos\beta + 3) \\ y = \sin\alpha \cdot (\cos\beta + 3) \\ z = \sin\beta \end{cases}$	Тор	-π ÷π	-π ÷π
3	$\begin{cases} x = \cos\alpha \cdot (\cos\beta + 3) \\ y = \sin\alpha \cdot (\cos\beta + 3) \\ z = \sin\beta + \alpha \end{cases}$	Спіраль	-2π ÷2π	-π ÷π
4	$\begin{cases} x = \alpha \cdot \cos\alpha \cdot (\cos\beta + 1) \\ y = \alpha \cdot \sin\alpha \cdot (\cos\beta + 1) \\ z = \alpha \cdot \sin\beta \end{cases}$	Логариф-мічна спіраль	0÷3π	-π ÷π
5	$\begin{cases} x = \alpha \cdot \cos\alpha \cdot (\cos\beta + 1) \\ y = \alpha \cdot \sin\alpha \cdot (\cos\beta + 1) \\ z = \alpha \cdot \sin\beta - ((\alpha + 3)\pi/8)^2 \end{cases}$	Морська раковина	0÷8π	-π ÷π

6	$\begin{cases} x = \cos\alpha \cdot \cos\beta + 3\cos\alpha \cdot (1.5 + \sin(0.75\alpha)) \\ y = \sin\alpha \cdot \cos\beta + 3\sin\alpha \cdot (1.5 + \sin(0.75\alpha)) \\ z = \sin\beta + 2\cos(1.5\alpha) \end{cases}$	Трилистник	-2π ÷2π $\frac{\pi}{20} \div \pi - \frac{\pi}{20}$	-π ÷π
7	$\begin{cases} x = \cos\alpha \cdot \sin\beta \\ y = \sin\alpha \cdot \sin\beta \\ z = \cos\beta + \lg(\mathrm{tg}(\beta/2)) + 0.2\alpha - 4 \end{cases}$	Поверхня Дини	0÷4π	0.001÷2
8	$\begin{cases} x = (1 + \beta/2 \cdot \cos(\alpha/2)) \cdot \cos\alpha \\ y = (1 + \beta/2 \cdot \cos(\alpha/2)) \cdot \sin\alpha \\ z = \beta/2 \cdot \sin(\alpha/2) \end{cases}$	Стрічка Мьобіуса	0÷2π	-1÷1
	Рівняння		x	y
9	$z = \sin x \cdot \cos y$		-5÷5	-5÷5
10	$z = \frac{a(x-5)}{b(y-2)^3}$		0÷5	0÷1.8
11	$z = (x-1)^5 + (y-1)^5$		-4÷4	-4÷4
12	$z = \sin\left(\sqrt{x^2 + y^2}\right)$		0÷2π	0÷2π
13	$z = x^2 y^3$		-1÷1	0÷3

8.3 Методичні рекомендації до виконання

1. Для варіантів 1-8, у яких рівняння поверхонь задані у параметричному вигляді, функцію meshgrid застосовуйте для параметрів α і β і тільки потім розраховуйте матриці **X**, **Y** та **Z**.
2. Продемонструйте різницю між стилями shading interp, shading faceted та shading flat.

8.4 Контрольні питання та завдання

1. Поясніть різницю між операторами plot3, mesh та surf.
2. Як створити матриці даних для операторів mesh та surf.
3. Які стилі зображення 3-вимірних поверхонь ви знаєте?
4. Як змінити кольорову палітру 3-вимірного графіка?

Лабораторна робота № 9

ЗНАЙОМСТВО З СЕРЕДОВИЩЕМ ПРОГРАМИ СТРУКТУРНОГО МОДЕЛЮВАННЯ *Simulink* СИСТЕМИ ПРОГРАМУВАННЯ *MATLAB*

Мета роботи: ознайомитися з можливостями програми *Simulink* та з основними її функціями

9.1 Запуск *Simulink.* Перелік бібліотек та демонстрацій

Simulink можна запустити з командної строки пакета *MATLAB* однойменною командою

```
>>simulink
```

(>> – запрошення *MATLAB* до вводу команди) або спеціальною кнопкою на панелі інструментів, що показана на рис. 9.1.

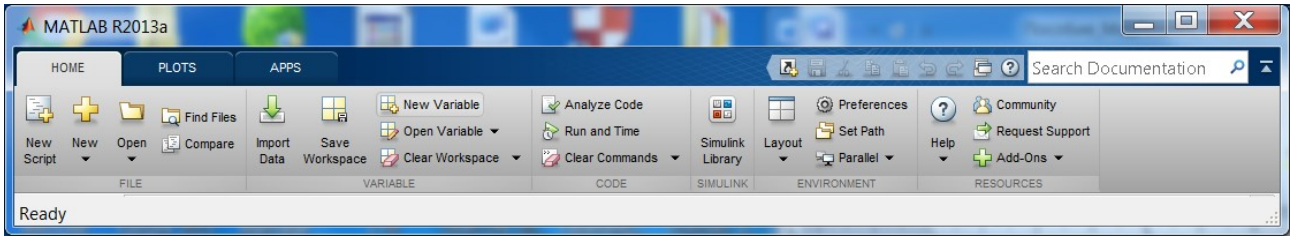


Рис. 9.1. Фрагмент робочого стола системи *MATLAB*

При цьому на екрані відкривається вікно з бібліотеками блоків, представлене на рис. 9.2, яке має назву ***Simulink Library Browser (Навігатор Бібліотек)***.

Кожну бібліотеку можна розкрити подвійним натиском лівої кнопки миші на піктограмі бібліотеки в правій частині вікна або вибором назви бібліотеки в лівій частині вікна. При цьому піктограми блоків обраної бібліотеки заміщають піктограми бібліотек.

Натиском правою кнопкою миші на імені бібліотеки через меню, що випадає, можна відкрити будь-яку бібліотеку у вигляді, звичному для користувачів більш старих версій.

Simulink зі старим інтерфейсом можна також активізувати з командної строки командою simulink3.

При цьому відкривається вікно, зображене на рис. 9.3.

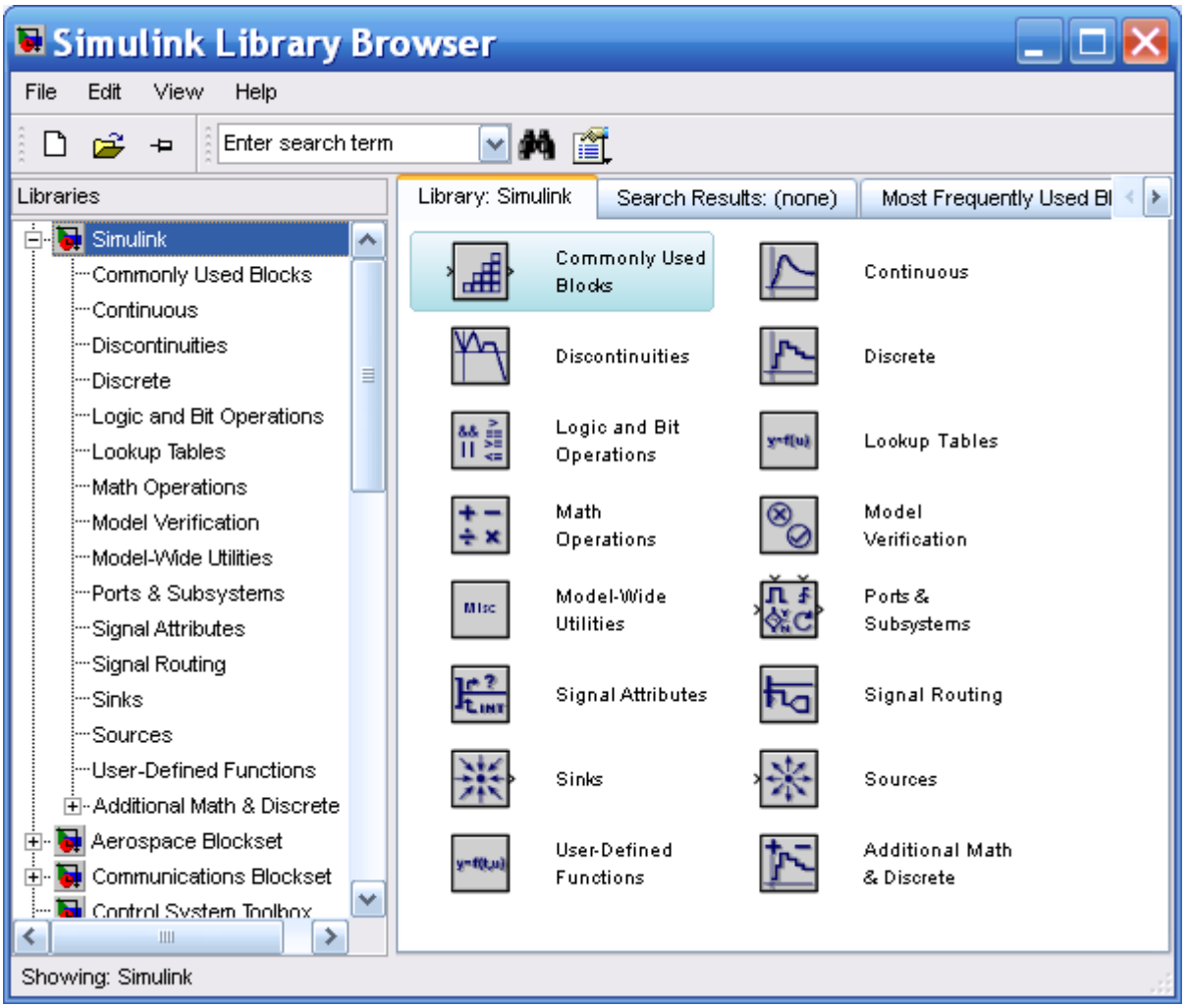


Рис. 9.2. Вікно *Simulink Library Browser*

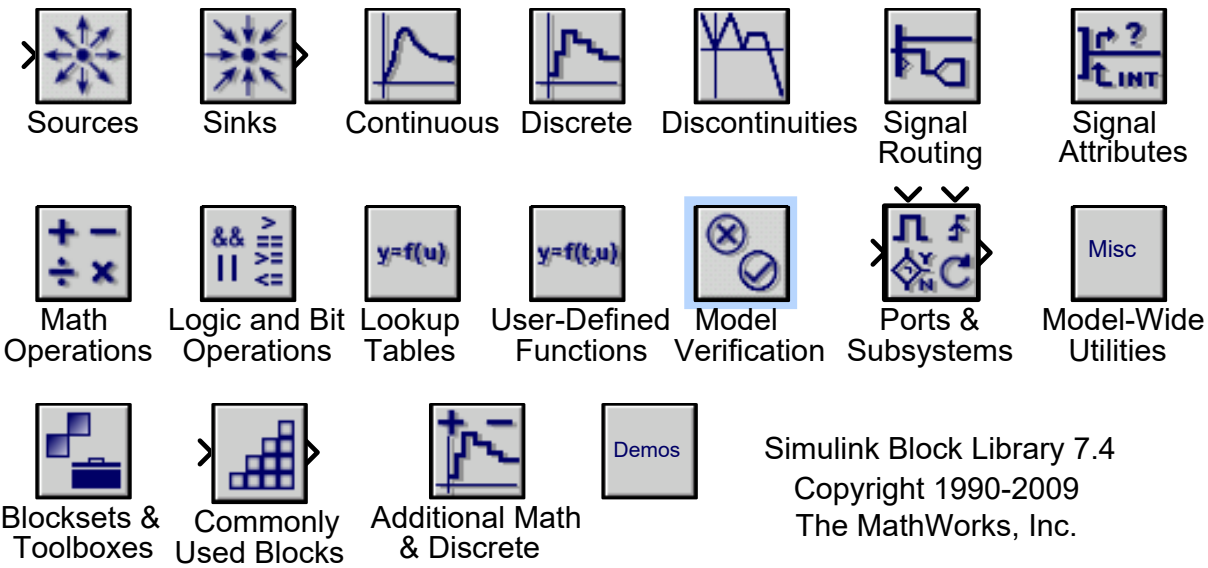


Рис. 9.3. Вікно бібліотек блоків *Simulink*

З цього вікна кожну бібліотеку можна розкрити подвійним натиском миші по її піктограмі. Вікна бібліотек будуть показані нижче (при розгляді їхніх блоків).

До основних *Simulink-бібліотек* відносяться

Source – джерела вхідних сигналів;

Sinks – блоки реєстрації та візуалізації;

Continuous – неперервні лінійні динамічні ланки;

Discontinuities – типові нелінійності;

Math Operations – математичні операції;

Logic & Bit Operations – логічні та бітові операції;

Lookup Tables – кусково-лінійна апроксимація табличних функцій;

User Definde Functions – функції користувача;

Ports & Subsystems – порти та підсистеми;

Signals Routine – блоки маршрутизації моделі;

Commonly Used Blocks – блоки, що використовуються найчастіше.

При подвійному натиску мишею по піктограмі блоку відкривається вікно введення його параметрів, у якому відображуються ім‘я блоку, його тип, короткий опис і рядки введення параметрів із супроводжуючими їх запитами. Більш докладну інформацію про блок можна побачити, звернувшись до функції *Help*. Піктограма блоку часто відображує його динамічні або статичні властивості і реагує на зміну параметрів. Як приклад, на рис. 9.6 наведено вікно введення параметрів блоку *Transfer Function.*

Параметри блоків можуть бути константами, перемінними, функціями або виразами, припустимими в *MATLAB*. Будь-які перемінні, від яких залежить параметр, повинні бути визначені в робочому середовищі до початку процесу моделювання, інакше *Simulink* сигналізує про помилку в цьому блоці.

Клавіша *Apply* дозволяє оцінити результат зміни параметрів, не закриваючи вікна їхнього введення, клавіша *OK* закриває вікно введення

параметрів із запам‘ятовуванням виконаних змін, клавіша *Cancel* закриває вікно введення параметрів зі скасуванням внесених змін.

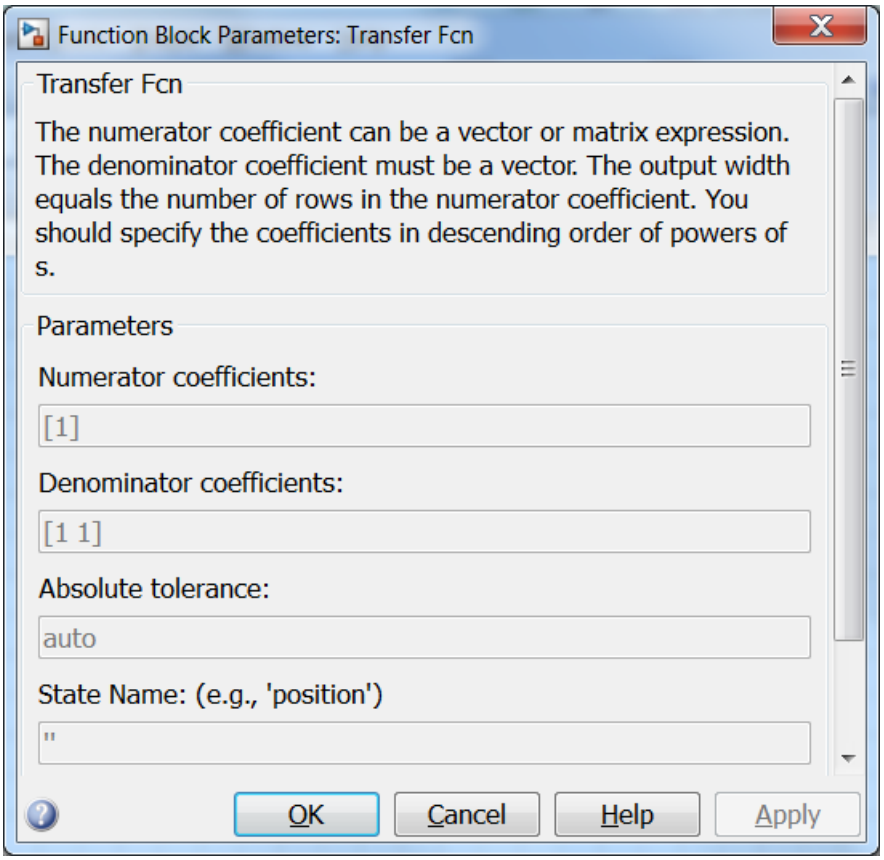


Рис. 9.4. Вікно введення параметрів блока *Transfer Fcn*

Для знайомства з можливостями додатку *Simulink* та особливостями роботи деяких блоків можна скористуватися демонстраціями, які зручно запускати з вікна *MATLAB Demo Window*. Воно з’являється на екрані при виконанні команди

>>demo simulink

Імена демонстрацій співпадають з іменами *mdl*-файлів, що знаходяться відповідно у папках

Matlab\toolbox\simulink\simdemos\ simgeneral

Matlab\toolbox\simulink\simdemos\ simnew

Matlab\toolbox\simulink\simdemos\ simfeatures

Matlab\toolbox\simulink\simdemos\ automotive

Matlab\toolbox\simulink\simdemos\ aerospace

Matlab\toolbox\powersys\powerdemo

Отже, кожну окрему демонстрацію можна виконати з командної строки введенням імені файлу моделі без поширення.

Для спілкування користувача з ЕОМ під час роботи з поширенням *Simulink* у середовищі системи *MATLAB* передбачені такі засоби:

- меню командного вікна *MATLAB*;
- основні меню вікон *Simulink*-моделей, *Simulink*-бібліотек та вікна *Simulink Library Browser*;
- контекстуальні меню цих вікон, що викликаються правою кнопкою миші при виділеному будь-якому елементу вікна;
- кнопки панелі інструментів;
- команди, що набираються та вводяться користувачем безпосередньо у вікні *Matlab Command Window* або у програмному режимі при виконанні програми, що складається з відповідних команд.

9.2 Меню вікна *Simulink Library Browser*

Меню вікна *Simulink Library Browser* містить наступні функції:

File – робота з файлами;

Edit – редагування;

View – вигляд вікна;

Help – допомога.

Кожну функцію цього меню можна вибрати не тільки мишею або клавішами <→>, <←> і <*Enter*>, але й комбінацією клавіш "<*Alt*>+*s*", де *s* – підкреслений символ імені обираної функції.

Функція *File* містить наступні операції:

New... ▸ – створити вікно для введення нової моделі (*Model (^N)*) або бібліотеки (*Library*);

Open... (^O) – відкрити модель;

Close – закрити вікно *Simulink Library Browser*;

Preferences… – переваги (настроювання середовища).

Операції підменю можна вибрати мишею, клавішами <↓>, <↑> та <*Enter*>, комбінацією клавіш, зазначеної в дужках, і клавішею <*s*>, де *s* – підкреслений символ імені обираної операції.

Для збереження моделей і інших файлів користувача в *Windows* передбачена папка *C:\Users\...\Documents\Matlab*. У ній можна створити нову папку, наприклад, *C:\ Users\...\Documents\Matlab\models*. Шлях до файлів-моделей необхідно додати у список доступних директорій за допомогою операції *Set Path→ Add Folder…/Add with Subfolder*. Нову папку можна установити в початок списку (*Move to Top*), у кінець списку (*Move to Bottom*), перемістити на одну позицію вниз (*Move Down*) або догори (*Move Up*). Якщо не виконати запис нового списку у файл, то установка діє тільки протягом поточного сеансу роботи. Для того, щоб використовувати нову установку в майбутніх сеансах, потрібно записати її у файл (*MATLAB \ toolbox \ loc l\ pathdef.m*) за допомогою клавіші *Save.*

Файли моделей і бібліотек додатка *Simulink* повинні мати маску **.mdl*. Вони є текстовими файлами особливої структури і містять інформацію про структуру та параметри моделей, достатню для їхнього графічного відображення і моделювання. З вікон *Simulink* вони відкриваються в графічному вигляді, а з будь-якого текстового редактора (наприклад, вбудованого текстового редактора системи *MATLAB medit.exe*) – у текстовому. Починаючи з версії *MATLAB-2012a*, розширення *mdl* замінено на *slx*.

Текстовий редактор системи *MATLAB* відкривається командою *File → New → M-file* меню командного вікна *MATLAB.*

Відкрити модель можна не тільки з меню *File* будь-якого *Simulink-вікна*, але й з меню *File* пакета *MATLAB* за допомогою функції *Open…*, а також з командного рядка *MATLAB* введенням у ній імені файлу-моделі без розширення. Оскільки в такий же спосіб (з командного рядка) у *MATLAB*

викликаються на виконання *script-файли* (головні програми, написані алгоритмічною мовою системи *MATLAB*) і виводяться значення перемінних [1, 2], тож файлам потрібно давати оригінальні і не занадто короткі імена.

Тут слід зазначити, що при використанні імен перемінних, функцій, файлів, програм, що збігаються з іменами, визначеними в робочому просторі або у файловій системі *MATLAB* раніше, вона не виводить попереджень про збіг імен. При виконанні команди, що містить яке-небудь ім'я, *MATLAB* шукає це ім'я серед перемінних, потім серед файлів з розширеннями *m*, *mex*, *mdl* і серед убудованих функцій в указаному порядку. Перше з виявлених імен сприймається як об'єкт для виконання команди. Отже, використання неоригінальних імен файлів може призвести до казусів.

Якщо при спробі відкрити модель або виконати яку-небудь програму з командного рядка на екран виводиться повідомлення *Undefined function or variable 'name'* (невизначена функція або перемінна), то це означає, що користувач або помилився при наборі імені файлу, або цей файл знаходиться в недоступній для системи *MATLAB* директорії (у більш пізніх версіях *MATLAB* приєднання нових директорій до списку доступних виконується системою програмування автоматично при зверненні до будь-якого файла цієї директорії).

Функція *Preferences* дозволяє виконувати настроювання середовища *MATLAB* і *Simulink*.

Для зображення векторних сигналів стовщеними єднальними лініями слід установити прапорець у поле *Wide nonscalar lines*, а для виводу типів вихідних сигналів – у поле *Port data types* опції *Display*.

У вікні *Preferences* можна установити шрифти (*Simulink Fonts*), що використовуються для позначення блоків (*Blocks*), єднальних ліній (*Lines*) і коментарів (*Annotations*), а також параметри моделювання (*Simulink Simulation Preferences*).

Функція *Edit* містить наступні команди:

Add to the current model (^I)– скопіювати обраний блок або бібліотеку в активне вікно моделі чи бібліотеки;

Find block…(^F) – знайти блок за його ім‘ям;

Find next block (<F3>) – знайти наступний блок.

9.3 Меню вікон *Simulink*-моделей

Меню вікон *Simulink*-моделей відрізняється від меню вікна *Simulink Library Browser* і отримує функції та кнопки, які наведені на рис. 9.8.

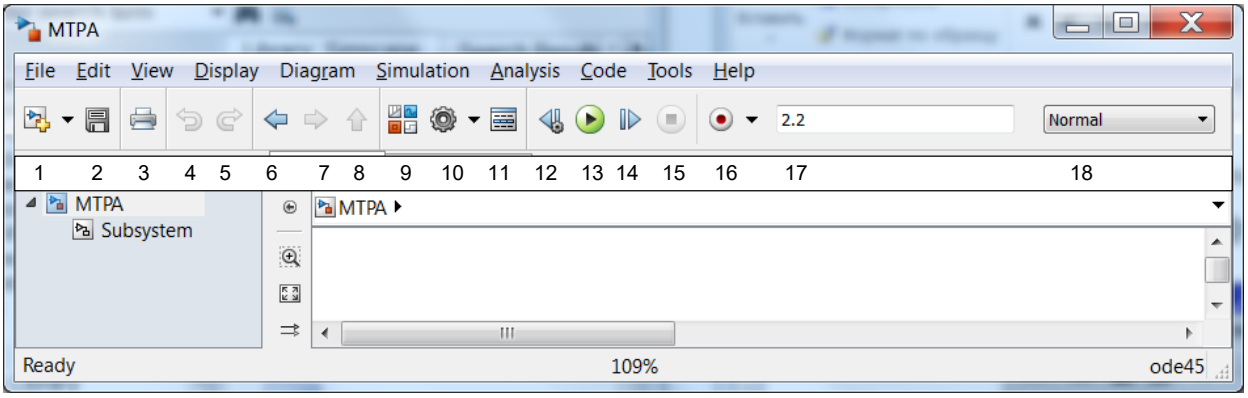


Рис. 9.8 – Меню та кнопки вікна *Simulink*-моделі

Перші п'ять кнопок виконують стандартні операції *Windows:* 1 – *New Model* (відкрити нове *Simulink*-вікно), 2 – *Save* (зберегти), 3 – *Print* (надрукувати), 4,5 – Undo, Redo (скасувати, відновити результат останнього редагування).

Інші клавіші виконують специфічні операції *Simulink*: 6,7,8 – Model Browser (уперед, назад, активізувати батьківську модель); 9 – *Library Browser* (активізувати „навігатор бібліотек”); 10 – *Model Configuration Parameters* (параметри моделювання); 11 – *Model Explorer* (увімкнути „навігатор моделі”); 12 – Stepping Options (вибір параметрів покрокового режиму) 13 – *Run* (почати моделювання); 14 – *Step Forward* (крок уперед); 15 – *Stop simulation* (зупинка моделювання); 17 – *Simulation Stop Time* (час моделювання); 18 – *Simulation Mode* (вибір режиму моделювання).

Установки меню 18 не слід змінювати починаючим користувачам.

У полі заголовка вікна рис. 9.8 відображено шлях до моделі у файловій системі. Символ “*” у кінці шляху означає, що після редагування моделі не виконано її запис у файл.

Функція File містить стандартні операції роботи з файлами:

New ▸ – створити вікно для введення нової моделі (*Model ^N*) або бібліотеки (*Library*);

Open... (^O) – відкрити модель;

Close (^W) – закрити модель;

Save (^S) – зберегти модель у колишньому файлі;

Save As... – зберегти модель у новому файлі;

Source Control… – керування джерелами сигналів;

Model Properties ▸ – властивості моделі;

Preferences… – переваги (настроювання середовища);

Print...(^P) – вивід на принтер;

Print Setup... – установки принтера;

Exit MATLAB – вихід із системи *Matlab*.

Вікно Model Properties, яке відкривається при виборі відповідної операції, містить вкладки *Summary*, *Callbacks* і *History*. Вкладка *Summary* дозволяє увести творця (*Creator*), дату створення (*Created:*) і опис моделі (*Model description:*).

Вкладка Callbacks дозволяє визначити команди, що будуть автоматично виконані перед завантаженням моделі (*Model pre-load function:*), при її ініціалізації (*Model initialization function:*), перед стартом моделювання (*Simulation start function:*), після його закінчення (*Simulation stop function:*) і перед збереженням моделі у файлі (*Model pre-save function:*). Вкладка *Hystory* містить дані про модифікацію моделі.

Функція Edit містить як стандартні *Windows-операції* редагування, так і деякі специфічні операції:

Undo (^Z) – відмінити останню зміну;

Redo (^Y) – відновити останню зміну;

Cut (^X) – переміщення обраних об'єктів у буфер;

Copy (^C) – копіювання обраних об'єктів у буфер;

Paste (^V) – копіювання змісту буфера в обране місце графічного вікна;

Clear (Delete) – вилучення обраних об'єктів;

Select All (^A) – вибір всіх об'єктів в активному вікні

Copy model to clipboard – копіювання моделі в буфер;

Find… – пошук блоків, сигналів, коментарів та інших об'єктів;

Open block – відкрити підсистему;

Mask parameters… – параметри замаскованої підсистеми;

Block parameters… – параметри блоку;

Block properties… – властивості блоку;

Mask subsystem (^M) – маскування підсистеми;

Create subsystem (^G) – об'єднання обраних об'єктів у підсистему (*subsystem*);

Mask subsystem (^M) – маскування підсистеми;

Look undo mask (^U) – заглянути під маску;

Link options ▸ – опції зв'язку підсистеми або замаскованого блоку з бібліотекою;

Update diagram…(^D) – обновити блок-діаграму.

Команду *Update Diagram* необхідно використовувати в наступних випадках:

- після зміни бібліотечних блоків, копії яких використовуються в моделі;
- після зміни кількості вхідних або вихідних аргументів у *S-функції*, що описує модель.

Функція View має звичайні для *Windows* прапорці *Toolbar* і *Statusbar* та команди керування масштабом зображення *Zoom in*, *Zoom out*, *Fit system to view* і *Normal (100%)*.

Найбільш споживаною операцією функції ***Display*** є ***Signals & Ports***, за допомогою якої виконуються операції *Signal Dimensions*, *Wide Nonscalar Lines*, *Port Data Types* та деякі інші.

Важливими операціями функції ***Diagram*** є *Format* та *Rotate & Flip*.

До команд форматування моделі Format належать команди

Font Style – установлення типу та розміру шрифту для текстових написів (для цього треба попередньо виділити блок, а не його ім'я).

Foreground Colour ▸ – установлення кольору переднього плану (піктограма й ім'я блоку),

Background Colour ▸ – установлення кольору заднього плану (фон блоку),

Block shadow – показати/сховати тінь від блоку

Show Block Name – сховати/показати ім'я блоку,

Canvas Colour ▸ – колір екрана.

Якщо фоновий колір блоку не збігається з кольором екрана, то вихідні лінії зв'язку цього блоку приймають колір його фону (*Background*), у противному випадку вони приймають колір піктограми й імені блоку (*Foreground*). Вищесказане відноситься і до контурних ліній блоку.

До команд повороту елементів моделі Rotate & Flip належать команди

Clockwise (^R) – повернути блок на 90° за годинниковою стрілкою,

Counterblockwise (^↑R) – повернути блок на 90° проти годинникової стрілки,

Flip Block (^I) – повернути блок із праворуч на ліворуч або зверху вниз,

Flip Block Name – змінити положення імені блоку (для горизонтально розташованих блоків воно може розташовуватися вгорі або внизу, а для розташованих вертикально – праворуч або ліворуч).

Функція ***Simulation*** керує наступними режимами:

Update diagram…(^D) – обновити блок-діаграму,

Model Configuration Parameters...(^E) – установка і редагування параметрів моделювання,

Start (^T) – старт моделювання.

Вікно параметрів моделювання має декілька вкладок (панелей), з яких найчастіше використовують вкладки *Solver* та *Data Import/Export.*

Ці панелі подані на рис. 9.9 та рис. 9.10 відповідно. Вони визначають метод і параметри симуляції (розв‘язання рівнянь, що описують математичну модель).

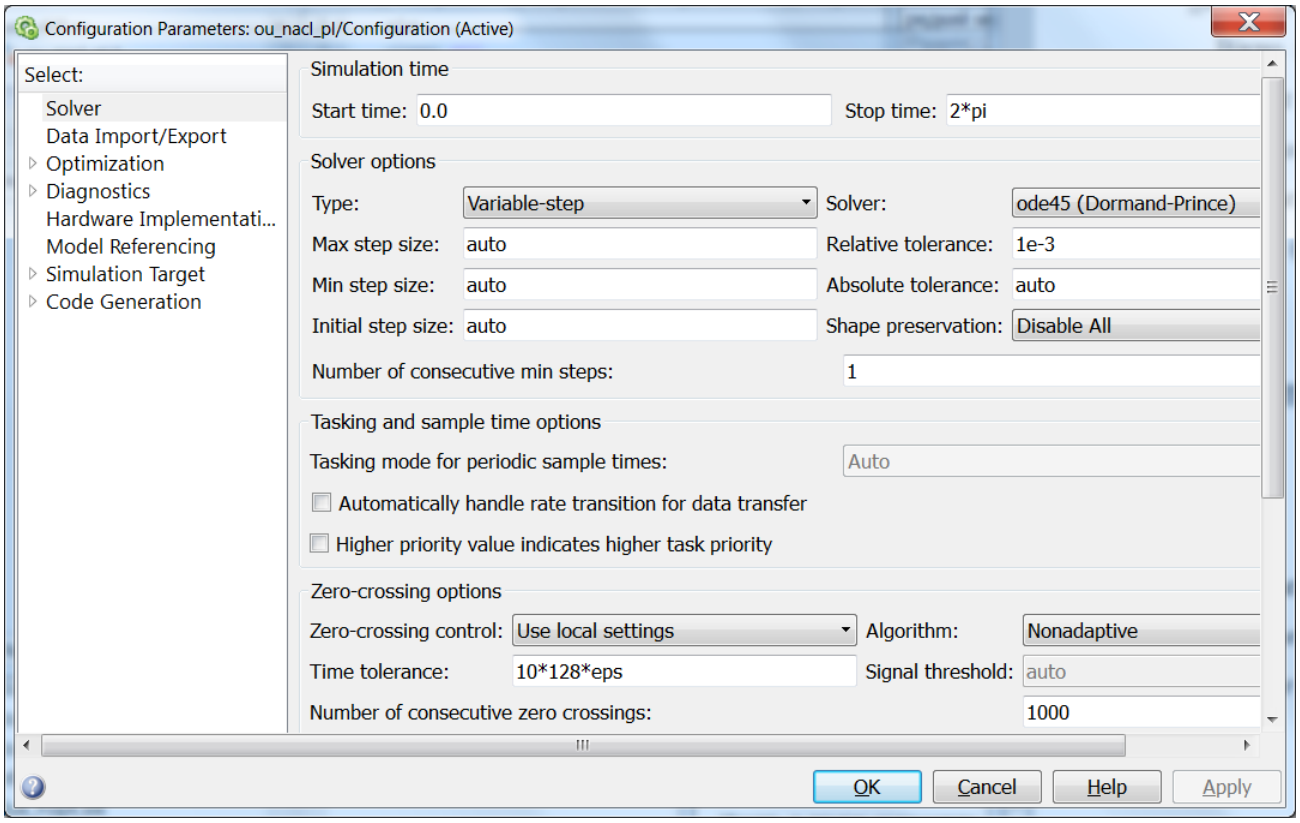


Рис. 9.9. Вкладка *Solver* вікна параметрів симуляції

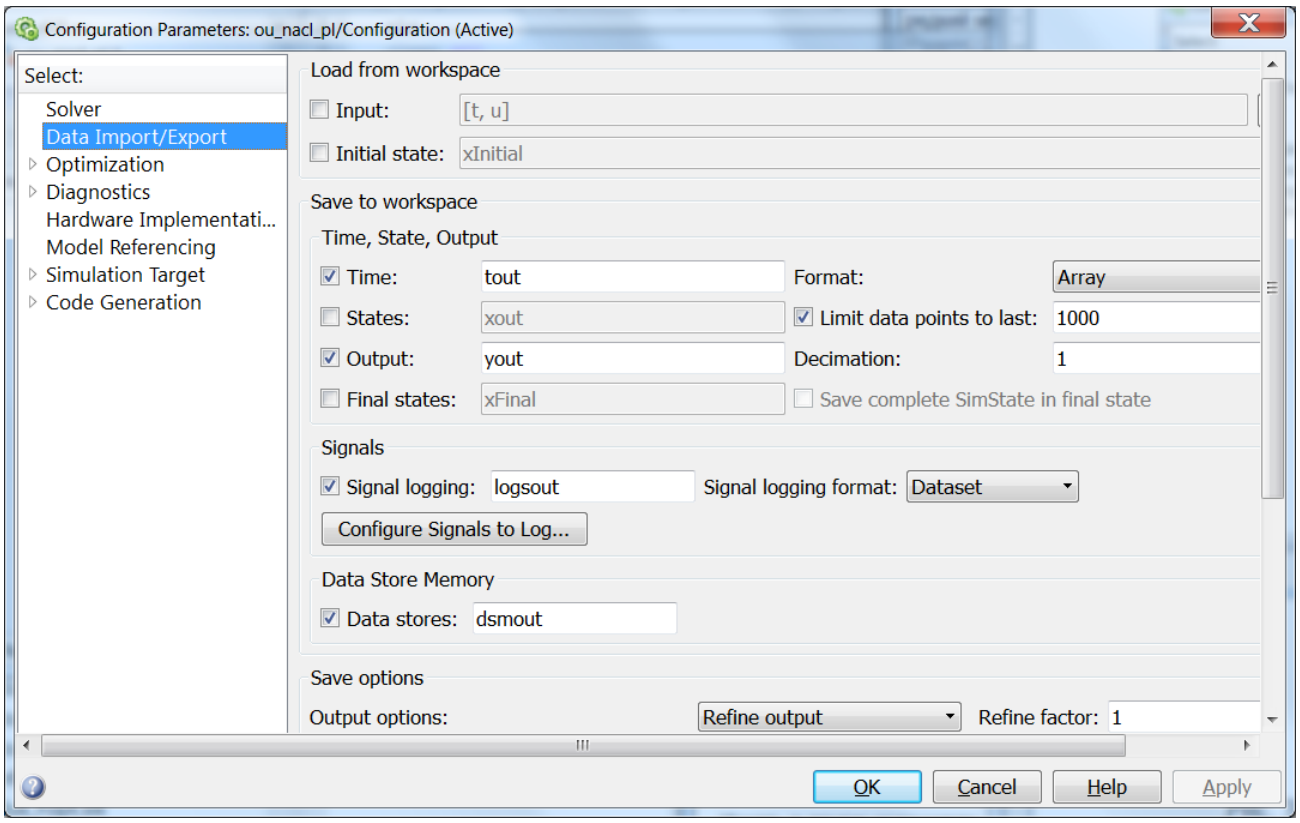


Рис. 9.10. Вкладка *Data Import/Export* вікна параметрів симуляції

Найбільш важливою є панель *Solver* (розв‘язання рівнянь, що описують математичну модель), яка передбачає введення таких параметрів:

Start time – час початку моделювання;

Stop time – час закінчення моделювання;

Type – тип розв‘язання: *Fixed/Variable-Step* (з фіксованим/змінним кроком);

Solver – метод розрахунку перехідних процесів (для неперервних систем – метод розв‘язання звичайних диференційних рівнянь з початковими умовами).

Серед ***методів розв‘язання ДР з фіксованим кроком (Fixed Step)*** в *Simulink* пропонуються ***методи Рунге-Кутта першого-п‘ятого порядків***, а саме:

ode5 – метод Рунге-Кутта 5-го порядку (*Dormand-Prince formula*);

ode4 – метод Рунге-Кутта 4-го порядку (*fourth-order Runge-Kutta formula*);

ode3 – метод Рунге-Кутта 3-го порядку (*Bogacki-Shampine formula*);

*ode*2 – модифікований метод Ейлера (*improved Euler's formula / Heun's method*);

*ode*1 – метод Ейлера (*Euler's method*).

Серед методів розв'язання ДР зі змінним кроком, або, вірніше, методів з заданою точністю, в *Simulink* пропонуються комбіновані методи Рунге-Кутта (4-5)-го (*ode*45) і (2-3)-го (*ode*23) порядків та деякі методи, більш пристосовані для розв'язання жорстких ДР:

*ode*45 – комбінований метод Рунге-Кутта (4-5)-го порядку з автоматичним вибором кроку, призначений для розв'язання нежорстких ДР;

*ode*23 – комбінований метод Рунге-Кутта (2-3)-го порядку з автоматичним вибором кроку, призначений для розв'язання нежорстких та слабко жорстких ДР;

*ode*113 – багатокроковий метод Адамса-Мілтона-Бешфорта (метод прогнозу та корекцій змінного порядку), призначений для розв'язання нежорстких ДР; може бути більш ефективним, ніж *ode*45, при високій точності;

*ode*15*s* – багатокроковий метод (1-5)-го порядку (за замовчанням 5-го), що базується на формулах чисельного диференціювання (*NDFs*), але може використовувати і формули зворотного диференціювання (*BDFs*), у відповідності з методом Гіра, спрямований на розв'язання жорстких ДР;

*ode*23*s* – однокроковий метод, що базується на модифікованій формулі Розенброка 2-го порядку; для деяких жорстких ДР виявляється при невисокій точності виявляється більш ефективним, ніж *ode*15*s*;

*ode*23*t* – метод трапецій з використанням „вільного" інтерполянта для розв'язання помірно жорстких ДР;

*ode*23*tb* – комбінований метод, що використовує на першому етапі формулу трапецій, а на другому –зворотну диференційну формулу другого

порядку; як і метод *ode*23*s*, може бути при невисокій точності більш ефективним, ніж *ode*15*s*, для розв‘язання жорстких ДР.

Якщо модель має тільки дискретні змінні стану, або зовсім не має динамічних ланок, то для моделювання використовують метод з фіксованим або змінним кроком. Змінний крок обирають, коли модель має у своєму складі дискретні динамічні ланки з різними періодами переривання, або коли треба фіксувати моменти часу перетину деякими сигналами нульового рівня.

При використанні методу *ode*15*s* рекомендовано для підвищення стабільності розв‘язання ДР знизити максимальний порядок формули *NDF*, що задається параметром *Maximum order*, з 5 (за замовчанням) до 3.

Для методів з фіксованим кроком треба задати значення цього параметру в полі *Fixed step size*.

Для методів зі змінним кроком задаються такі параметри:

Max step size – максимальний крок; за замовченням він розраховується за формулою $h_{\max} = (t_{stop} - t_{start})/50$;

Min step size – мінімальний крок;

Initial step size – початковий крок;

Relative tolerance – відносна точність (за замовченням 10^{-3});

Absolute tolerance – абсолютна точність (за замовченням 10^{-6});

Refine factor – коефіцієнт збільшення кількості точок моделювання (ціле число); для методу *ode*45 рекомендовано значення 4, для інших методів – 1.

Підвищити якість візуалізації перехідних процесів за рахунок корекції вектору часу шляхом додавання до нього бажаних точок можна установкою параметра *Output options* на панелі *Data Import/Export* у стан *Produce additional output* або *Produce specified output only*. Більшу кількість точок вихідних сигналів моделі можна отримати і зменшенням максимального кроку чисельного інтегрування але таке рішення менш ефективне з точки зору затрат машинного часу.

На панелі *Data Import/Export* у першу чергу зверніть увагу на доцільність наявності «прапорців» перед параметрами *Time* і *Output* та усунення «прапорця» перед параметром *Limit Data Point to Last*. Сенс цих та деяких інших установок буде пояснений пізніше.

9.4 Меню вікон *Simulink*-бібліотек

Меню вікон *Simulink*-бібліотек має багато спільного з меню вікон *Simulink*-моделей.

Між ними існує така різниця:

- у меню бібліотек відсутні функції *Simulation* та *Tools*;
- у заблокованому стані у вікні бібліотек не можливо виконання операцій редагування, тобто усіх операцій функції *Format*, операцій *Clear*, *Cut*, *Paste*, *Create Subsystem*, *Mask Subsystem*, *Mask Parameters*, *Edit mask*, *Link options*, *Update diagram*, *Undo*, *Redo* функції *Edit* та операцій редагування, які здійснюються за допомогою “миші” (пересування блоків, зміна їх розміру, назви, тощо);
- для розблокування бібліотек функція *Edit* їхніх вікон має операцію *Unlock Library*.

Стан *Unlock Library* діє з моменту виконання відповідної операції до закриття вікна бібліотеки і поширюється на вкладені бібліотеки. При наступному відкритті бібліотеки вона буде знову заблокованою.

Користувач може редагувати існуючі бібліотеки (у розблокованому стані) та створювати власні.

Для створення нової бібліотеки треба відкрити її вікно через меню будь-якого вікна *Simulink*-моделі, *Simulink*-бібліотеки або *Simulink Library Browser: File* → *New* → *Library*, занести в нього необхідні блоки і закрити зі зберіганням у файлі.

Файли бібліотек, як і файли моделей, мають поширення *mdl* або *slx* (у нових версіях).

9.5 Типи *Simulink*-блоків

Simulink-блоки можуть створюватися різними способами.

Основу стандартних *Simulink*-бібліотек складають вбудовані блоки, які не доступні для перегляду користувачем у вигляді текстового файлу або структурної схеми.

Деякі стандартні блоки створені за допомогою вже існуючих блоків і являють собою або модифікацію деякого блоку з іншими параметрами і зміненою піктограмою, або комбінацію декількох блоків, об'єднаних в одну підсистему. Решта блоків створені програмно за допомогою апарата *MATLAB*-функцій або *S*-функцій. Після імені такого блоку у вікні введення його параметрів стоять позначки (*mask*) і (*link*), що означають, що блок замаскований і зв'язаний з однією з бібліотек. У такий же спосіб може створювати нові блоки і користувач. Заглянувши під маску (*Edit* → *Look undo Mask* = *^U*), можна побачити в окремому вікні, з чого вони складаються (цю ж операцію можна виконати через контекстуальне меню, що відкривається щигликом правої кнопки миші по піктограмі блоку). Для перегляду і редагування параметрів маскування (*Edit* → *Edit mask…* = *^M*) необхідно попередньо тимчасово вивести з ладу зв'язок з бібліотекою (*Edit* → *Link options* → *Disable link*). Після перегляду і внесення змін в установки цього вікна зв'язок з бібліотекою можна або відновити (*Restore link*) або перервати (*Break link*). При відновленні зв'язку можливі два варіанти: ігнорувати результати редагування (*Use library block*) чи поширити ці результати на бібліотечний блок (*Update library*). При опису імена цих блоків будуть позначені символом “*”.

S-функції можуть бути написані за особливими правилами на мовах *MATLAB (*.m)*, або *C++ (*.cpp)*, або *C (*.c)*, або *Ada (*.adb)*, або *Fortran (*.f)*, а потім конвертовані в блоки *Simulink*.

9.6 Створення та редагування моделей

Перед початком формування структурної схеми необхідно відкрити для неї нове вікно (*Simulink* → *File* → *New* → *Model (^N)* або *MATLAB* → *File* → *New* → *Model*). При цьому відкривається вікно з заголовком "*Untitled*". При запису в файл (*File* → *Save / Save as…*) у якості заголовку вікна буде фігурувати призначене користувачем ім'я файлу, яке за замовченням отримає поширення *mdl*.

Вже існуючу модель можна завантажити з меню *Simulink* або *MATLAB* (*File* → *Open*) та із командного рядка *MATLAB* введенням в ній імені файла, в якому збережена модель, без поширення.

В основу створення та редагування моделей покладено, як і в більшості графічних *Windows*-додатків, ***принцип drag-and-drop (перетягни та покинь)***.

Копіювання блоків із бібліотек або з будь-якої вже існуючої моделі у вікно створеного файлу після їх відкриття виконується мишею за допомогою операції *drag* (тягнути при натиснутій лівій клавіші миші). Аналогічно здійснюється переміщення блоків всередині вікна. Копіювання блоків всередині вікна виконується операцією *drag right* (тягнути при натиснутій правій клавіші миші). При цьому до імені нового блока додається цифра, що відображає порядковий номер копіювання, чим забезпечується унікальність імені кожного блока однієї моделі.

Імена блоків можна змінювати безпосереднім редагуванням. При цьому не можна їх дублювати або залишати блок без імені (ім'я – пустий рядок), але можна сховати ім'я (*Format* → *Hide Name*). Зворотна операція здійснюється командою *Format* → *Show Name*. Для завершення редагування імені треба зробити щиглик мишею зовні поля введення тексту. Після цього ім'я аналізується системою та чи приймається, чи відкидається нею з виведенням повідомлення. Для зміни шрифту імені (*Format* → *Font…*) необхідно попередньо виділити сам блок, а не його ім'я.

В довільній точці активного *Simulink*-вікна, що виділяється щигликом лівої клавіші миші, можна вставити будь-які коментарі (*Annotations*). Введення та редагування коментарю виконуються точно так, як відповідні операції з ім‘ям блока. В результаті утворюється новий специфічний блок *Note*, який відрізняється від інших блоків тим, що він не має ні піктограми, ні портів, ні вікна введення параметрів, а складається з одного імені, яке є текстом анотації. Рештою цей блок схожий на інші: його можна копіювати, пересувати, знищувати, виділяти і т.п.

Іменами можна відмічати і лінії зв‘язку. Для цього необхідно клацнути двічі лівою кнопкою миші по обраній з‘єднувальній лінії та ввести текст в утворене поле. Отриманий у такий спосіб надпис буде, на відміну від блока *Note*, прив‘язаний до лінії зв‘язку.

При пересуванні блока, до якого вже приєднані лінії зв‘язку, останні витягуються або скорочуються для збереження зв‘язків; кінці зв‘язків, не приєднані до блоку, що пересовується не змінюють своє положення. Перемістити блок в межах одного вікна без ліній зв‘язку (висунути блок з моделі) можна операцією <*Shift*>+*drag*.

Для виконання якої-небудь дії з будь-яким об‘єктом моделі (блоком, лінією зв‘язку, сукупністю блоків та/або зв‘язків, тобто фрагментом моделі, усією моделлю) його треба спочатку відмітити. Поодинокий об‘єкт виділяється щигликом лівої клавіші миші (*click*). Фрагмент моделі можна виділити такими способами:

- помітити вибірково блоки або зв‘язки, не відпускаючи клавішу *Shift* (<*Shift*> + *click*);
- помітити підряд, тобто. заключити за допомогою мишки у прямокутник;
- помітити все (*Edit* → *Select All*=^*A*).

Про те, що виділення відбулося, свідчать маленькі незафарбовані квадратики, розташовані в кутах обраних блоків та поблизу кінців обраних зв‘язків.

З поміченим блоком можна виконувати такі операції:

- зміна розміру – *drag* за вугол;
- поворот на 90^{O} – *Options* → *Rotate* або ^*R*;
- поворот на 180^{O} – *Options* → *Flip Horizontal* або ^*I*;
- знищення – <*Delete*>;
- всі операції функцій *Edit, Options, Style* меню *Simulink*.

Подвійний щиглик по піктограмі блока розкриває текстове вікно для введення його параметрів.

Параметри блока можуть бути подані як в чисельному вигляді, так і у вигляді імен перемінних або математичних виразів. В останньому випадку до початку моделювання перемінним повинні бути присвоєні значення. Це можна зробити з командного рядка *Matlab* або виконанням командного файлу.

Зв'язки встановлюються між вхідними та вихідними портами блоків. Стрілка на зв'язку показує напрямок потоку даних. Утворюються зв'язки у такі способи:

- довільно кусочно-неперервно (*drag* від порту з перериванням цієї операції в бажаних точках зламу);
- з автоматичним розташуванням точок зламу.

Автоматичне розташування точок зламу забезпечується при виконанні операції *drag* від порту до порту без переривання, а також при швидкому з'єднанні блоків, яке виконується у такий спосіб: помічається початковий блок або блоки, натискається клавіша *Ctrl* і помічається кінцевий блок.

Для ***розгалуження лінії зв'язку*** використовують операцію *drag right* або ^ *drag left* або *drag* будь-якою клавішею але у зворотному напрямку, тобто від вхідного порту до точки розгалуження.

При проведенні ліній зв'язку не варто намагатися влучити точно в порт; лінія приєднається до порту, якщо відпустити клавішу миші при находженні графічного курсора всередині блока або поблизу порту (на відстані менше 5 пікселів).

З поміченими лініями зв‘язку можна виконувати такі операції:

- зміна напрямку – *Options* → *Rerout Lines,* або ^*L*;
- паралельне пересування – *drag,* ухопившись за середину сегменту зв‘язку;
- зміна кута між сегментами зв‘язку – *drag*, ухопившись за вузол;
- додатковий злам сегмента зв‘язку – <*Shift*>+ *drag*, ухопившись за бажану точку зламу;
- ділення зв‘язку – *drag*, ухопившись за першу ліву мітку зв‘язку або *drag right* від будь-якої точки зв‘язку;
- знищення – <*Delete*>.

З поміченими фрагментами моделі або з усією моделлю можна виконувати всі ті операції, що і з окремими блоками та/або зв‘язками.

Для того, щоб зробити розташування об‘єктів моделі більш зручним, вікна *Simulink* мають невидиму сітку 5х5 пікселів, до вузлів або до ліній якої прив‘язуються всі об‘єкти. Дискретне пересування виділених об‘єктів можна виконати клавішами <→>, <←>, <↑>, <↓>.

При редагуванні моделі ***графічний курсор змінює свою форму***, сигналізуючи користувачу про ту дію, до виконання якої підготовлена система:

 – готовий для наступних дій;

 – готовий для нанесення обмежувального прямокутника (виділення групи поруч розташованих об‘єктів);

 – готовий для пересування блока, сегмента лінії, або виділеного фрагмента;

 – готовий до проведення лінії зв‘язку;

 – готовий для перенесення або для створення нової точки зламу лінії;

 – готовий до зміни розміру блока.

9.7 Завдання

1. Ознайомитися з основними та спеціальними бібліотеками *Simulink* на рівні їхніх можливостей;
2. Ознайомитися з основними демонстраціями *Simulink* та демонстраціями поширення *Power System Blockset*, не вдаючись до зайвих подробиць;
3. Створити задану викладачем папку в директорії *C:\Пользователи\ Мои документы\MATLAB* та забезпечити доступ до неї з середовища пакета *MATLAB*;

9.8 Контрольні питання

1. Поясніть призначення команд меню середовища *MATLAB*, кнопок панелі інструментів та окремих вікон (*Command Window*, *Command History*, *Current Directory* та ін.).
2. Яким чином можна виконати встановлення за власним бажанням поточного каталогу та доступних каталогів *MATLAB*? Навіщо це робити?
3. Поясність призначення команд меню та кнопок панелей інструментів вікон *Simulink Library Browser* та *Simulink*.
4. Дайте загальну характеристику бібліотек та блоків *Simulink*.
5. Перелічите основні прийоми створення та редагування моделей. Як виконується встановлення потрібних параметрів моделювання, яка мета цього встановлення?

Лабораторна робота № 10

ОСНОВНІ ЗАСОБИ РЕЄСТРАЦІЇ ТА ВІЗУАЛІЗАЦІЇ СИГНАЛІВ У СЕРЕДОВИЩІ *Simulink*

Мета роботи: навчитися користуватися засобами реєстрації та візуалізації сигналів у середовищі *Simulink.*

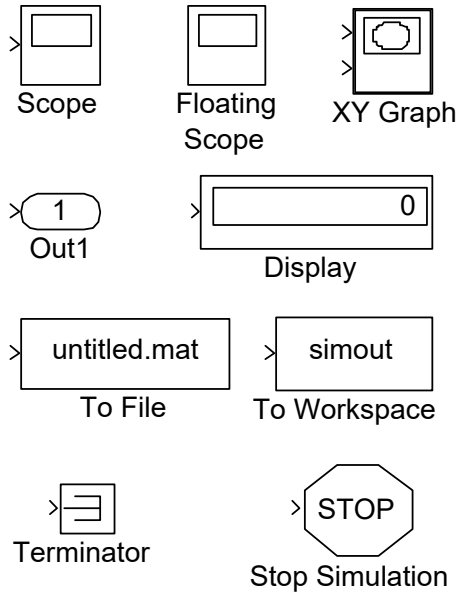


Рис. 10.1. Бібліотека вихідних блоків *Sinks*

Для реєстрації та візуалізації сигналів у *Simulink* існує бібліотека *Sinks*, блоки якої подані на рис. 10.1.

Більшість з них призначені для запам'ятання результатів моделювання *(To File, To Workspace, Out)*, а також для їх графічного *(Scope, Floating Scope, XY Graph)* або чисельного *(Display)* відображення. Усі вони мають тільки входи і не мають виходів.

Блок ***Terminator* (*Заглушка)*** приєднується до виходів блоків, не зв'язаних з іншими блоками для запобігання висновку попереджувального повідомлення про наявність у моделі не приєднаних виходів.

Розглянемо параметри, які є спільними для декількох блоків бібліотеки *Sinks*.

Параметр Decimation (Проріджування) у блоках *Scope*, *To Workspace*, *Display* визначає дискретність запам'ятовування або відображення інформації і може приймати тільки додатні цілочисельні значення *d*, що вказують через скількох кроків чисельного інтегрування варто фіксувати результати моделювання. При *d*=1 (значення за замовчуванням) блоком запам'ятовується інформація, отримана на кожнім кроці ЧІ, а при *d*=5 – тільки інформація,

отримана на кожному п'ятому кроці. При інтегруванні з постійним кроком параметр d – це відношення кроку фіксації до кроку інтегрування: $h_{out} = dh_{int}$.

Якщо в процесі моделювання для рішення диференціальних рівнянь обрано метод ЧІ зі змінним кроком, а крок фіксації повинний бути постійним, то для дискретизації виведення замість параметра *Decimation* варто використовувати ***параметр Sample Time (Період дискретності)***, обираючи його значення рівним бажаному кроку реєстрації: $T_s = h_{out}$. При $T_s = -1$ блок висновку успадковує період дискретності від блоку, вихідні сигнали якого він реєструє.

Параметр Limit data point to last у блоках *Scope*, *To Workspace*, *Out* визначає максимальну кількість точок для збереження сигналу (відлік ведеться від останньої розрахованої точки). Для того, щоб не втратити інформацію, значення параметра *Limit data point to last* можна установити рівним ∞ (константа *Inf*) або прибрати прапорець, що робить цю опцію активною.

Параметр Save format у блоках *Scope*, *To Workspace*, *Out* визначає формат запам'ятовування даних.

Інформація може бути збережена в наступних форматах: *Structure with time* (*Структура з часом моделювання*), *Structure* (*Структура без часу моделювання*) і *Array* (*Масив*).

Найпростішим типом даних є ***Array***. У цьому випадку кожен сигнал записується в окремий стовпець матриці. Один рядок матриці містить стан усіх сигналів у конкретний момент часу. Кількість рядків при d=1 дорівнює числу кроків моделювання. Якщо дані збережені в матриці y, то виділити з неї j-ий сигнал можна операцією y(:,j), а всі сигнали в i-й зареєстрований момент часу – операцією y(i,:).

Якщо дані збережені у змінній y, що має формат *Structure with time*, то вектор-стовпець часу визначається звертанням до поля *time* змінної y (y.time), звертання y.blockName формує шлях до блоку (ім'я-моделі / ім'я підсистеми / ім'я-блоку), звертання y.signals.dimensions – кількість вхідних сигналів,

звертання y.signals.label – мітку лінії зв'язку, а звертання y.signals.values – чисельні значення матриці вхідних сигналів, організованої так само, як при використанні формату *Array*. Структура без часу (*Structure*), відрізняється від структури з часом тим, що її поле *time* має значення порожньої матриці.

10.1 Блоки візуалізації

Блок ***Scope*** **(*Осцилограф*)** у процесі моделювання відображає вихідні сигнали приєднаних до нього блоків. Для того, щоб побачити графіки перехідних процесів, необхідно відкрити його вікно (див. рис. 10.2), що поряд з полем виведення графіків містить такі «кнопки» (рис. 10.3):

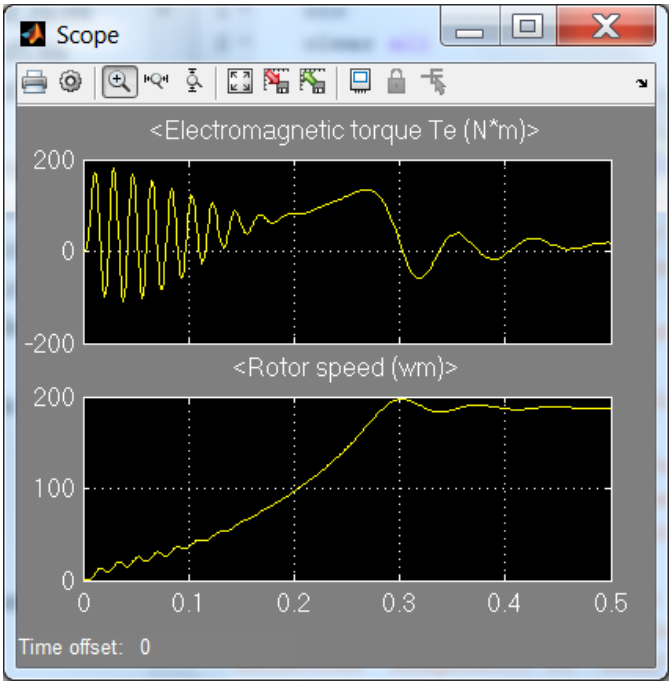


Рис. 10.2. Вікно блоку *Scope*

Рис. 10.3. Кнопки вікна блоку ***Scope***

1 – *Print* (друкування);

2 – *Parameters* (настроювання параметрів);

3 – *Zoom* (рівномірна зміна масштабу по обох осях);

4 – *Zoom X-axis* (зміна масштабу по осі *X*);

5 – *Zoom Y-axis* (зміна масштабу по осі *Y*);

6 – *Autoscale* (автоматичне масштабування);

7 – *Save current axes settings* (запам'ятовування системи координат);

8 – *Restore saved axes settings* (відновлення системи координат, наприклад, після зміни масштабу однією з клавіш *Zoom*);

9 – *Floating Scope* (плаваючий осцилограф;

10 – *Lock/Unlock axes selection* (заблокувати/розблокувати вибір ліній зв'язку для плаваючого осцилографа);

11 – *Signal Selection* (вибір сигналів для плаваючого осцилографа);

12 – *Dock Scope* (осцилограф виводить графіки без кнопкової панелі).

Звичайно після закінчення процесу моделювання для того, щоб побачити графіки в нормальному масштабі, натискають кнопку *Autoscale*, а потім запам'ятовують діапазони систем координат кнопкою *Save current axes settings*.

Якщо результати автоматичного масштабування за якимись причинами не влаштовують користувача, то він може змінити межі координатних осей графіків вручну за допомогою кнопок 3-5 візуально або установкою границь системи координат у чисельному вигляді.

Масштабування зображення в обраному виміру кнопками можна виконувати двома способами. При першому способі після активізації відповідної кнопки курсор миші підводять до бажаної ділянки графіка і клацають на ньому лівою клавішею. При кожнім натисканні масштаб буде збільшуватися, що приведе до відображення у вікні всі меншого і меншого фрагмента графіка. При другому способі фрагмент графіка, що хочуть збільшити, виділяють за допомогою курсору. Повернення до результатів попереднього масштабування виконується командою контекстуального меню *Zoom out*, а повернення до вихідного масштабу – кнопками *Autoscale* або *Restore saved axes settings*.

Для установки меж графіка по осі ординат у чисельному вигляді необхідно викликати контекстуальне меню щигликом правої кнопки миші в площині системи координат, вибрати з нього функцію *Axes Properties* і у вікні, що відкрилося, установити *параметри* Y-min *і* Y-max. У цьому ж вікні можна установити заголовок системи координат (параметр *Title*), замінивши коментар *%<SignalLabel>* бажаним текстом або вставивши цей текст перед коментарем.

Діапазон часу, що є спільним для всіх систем координат, можна змінити у ***вікні Parameters***, що відкривається однойменною кнопкою й отримує 2

вкладки: *General* і *Data history*.

За допомогою ***вкладки General*** (див. рис. 10.4) установлюються наступні параметри:

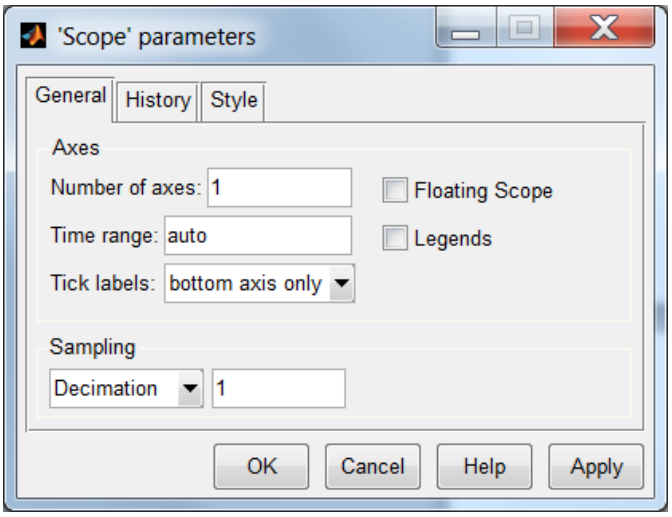


Рис. 10.4. Вкладка *General* вікна параметрів блоку *Scope*

- *Number of axes* – кількість систем координат, що визначає і кількість вхідних портів;
- *Time range* – діапазон часу;
- *Tick labels* – режим візуалізації чисельних значень міток координатної сітки, що може приймати значення *all* – значення часу виводяться в усіх підвікнах (системах координат), *bottom axis only* – значення часу виводяться тільки в самій нижній системі координат) і *none* (значення міток не виводяться, і графіки займають максимально можливу площу вікна);
- прапорці *Floating Scope* та *Legend*;
- параметри *Decimation* або *Sample Time*, що визначають дискретність відображення інформації (див. опис спільних параметрів).

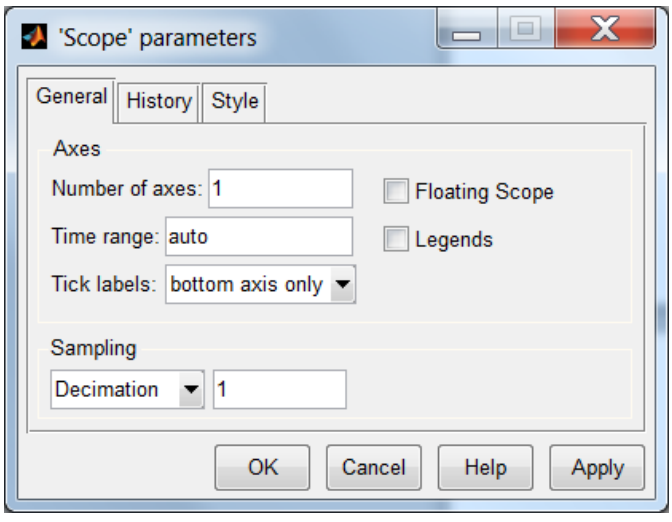


Рис. 10.5. Вкладка *History* вікна параметрів блоку *Scope*

За допомогою ***вкладки History*** (див. рис. 10.5) назначаються параметри *Limit data point to last*, *Variable Name* і *Save format*. Два останніх параметри активізуються тільки при встановленому прапорці *Save data to workspace* (зберегти дані в оперативній пам'яті). При установці цього прапорця блок

Scope починає виконувати, крім власних функцій, функції ланок *To Workspace* або *Out*. За замовчуванням для запису даних пропонується змінна з ім‘ям *ScopeData* у форматі *Structure with time*.

При виборі формату *Array* у перший стовпець матриці *ScopeData* записується час моделювання, а у інші – значення сигналів, приєднаних до осцилографа. Тому при побудові усіх графіків перехідних процесів в одному вікні в цьому випадку можна застосувати оператор

```
plot(ScopeData(:,1),ScopeData(:,2:end))
```

Формат *Array* не можна використовувати для збереження інформації багатоканальним *Осцилографом*. Поле *signals* змінної *ScopeData* у цьому випадку є вектором структур, з кількістю елементів, рівним числу портів (каналів) *Осцилографа*. На додаток до полів *dimensions*, *values* і *label* кожен елемент змінної *signals* має ще поля *title* і *plotStyle*, які містять інформацію про заголовки графіків і стилі їхнього зображення.

За допомогою ***вкладки Style*** (рис. 10.6) назначаються такі параметри:

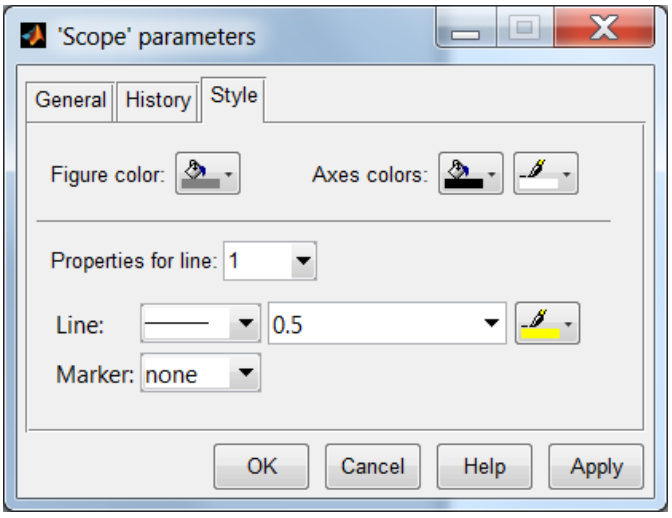


Рис. 10.6. Вкладка *Style* вікна параметрів блоку *Scope*

- *Figure Colour* – колір графічної фігури;
- *Axes Colour* – колір системи координат (колір фону і колір ліній градуювання);
- *Line* – стиль зображення ліній, їх товщина та колір;
- *Marker* – стиль зображення маркерів точок графіка.

Останні три параметри встановлюються окремо для кожного графіка, вибір якого здійснюється параметром *Parameters for line:*.

Параметри *Осцилографа* можна редагувати в процесі моделювання.

Розмір і пропорції вікна *Scope* можна змінювати довільно.

Скалярний сигнал зображується завжди жовтим кольором, а для складових векторного сигналу використовуються за замовчанням повторювані циклічно 6 кольорів: *жовтий, малиновий, блакитний, червоний, зелений, синій*. Фон зображення за замовчанням *чорний*.

Приклади використання блоку *Scope* для візуалізації двох синусоїдальних сигналів різної амплітуди і частоти показано на рис. 10.7.

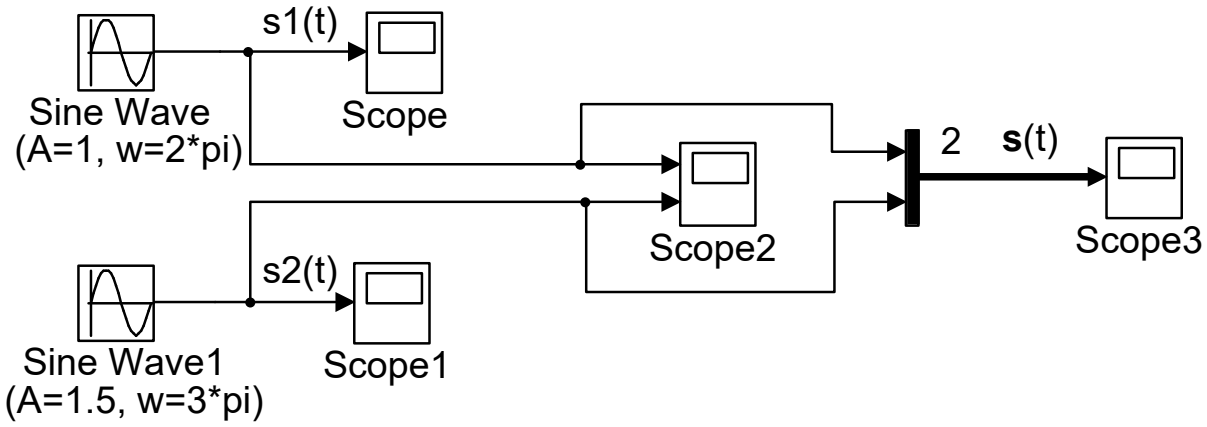


Рис.10.7. Модель візуалізації двох сигналів блоками *Scope*

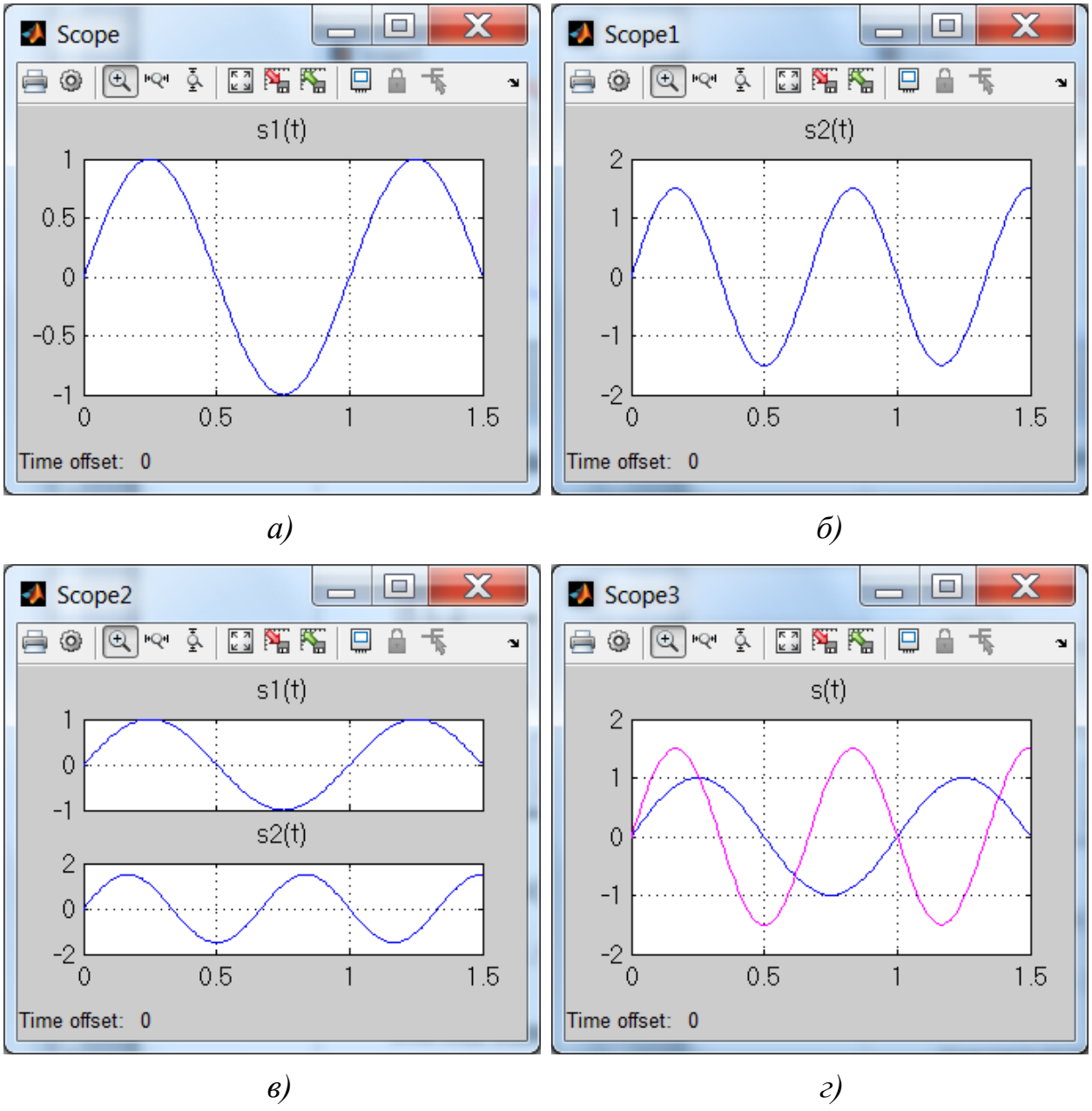
Вигляд відкритих після симуляції осцилографів показано на рис. 10.8.

Блоки *Scope* та *Scope1* відображують синусоїди у різних вікнах, а блоки *Scope1* та *Scope3* – в одному вікні. Різниця між останніми полягає в тому, що *Scope2* відкриває 2 системи координат, кожна у своєму підвікні, що досягається установленням у полі параметру *Number of axes* вкладки *General* вікна *Parameters* значення 2, а *Scope3* зображує обидва сигнали, попередньо об‘єднані блоком *Mux* (Мультіплексор) в один векторний сигнал, в одній системі координат.

***Блок Floating Scope* (*Плаваючий Осцилограф*)** можна взяти з бібліотеки або переключити звичайний осцилограф *Scope* у режим, що плаває, відповідною кнопкою або установкою відповідного прапорця. *Floating Scope* не має вхідних портів. Підключення до нього реєстрованих сигналів можна виконати у *вікні Signal Selector*, що відкривається кнопкою 11 (див. рис. 10.1). *Signal Selector*, завдяки наявності в ньому засобів навігації, дозволяє вибрати

будь-які сигнали системи, включаючи внутрішні сигнали закритих підсистем. На відміну від блоку *Scope*, ***плаваючий осцилограф не має буфера для збереження зареєстрованих даних. Тому для нього не можливі режими масштабування графіків (кнопки 3-6 не працюють)***.

Єдиною перевагою плаваючого осцилографа є економія оперативної пам'яті.



а) *б)*

в) *г)*

Рис. 10.8. Результати візуалізації синусоїд блоками *Scope* (а), *Scope1* (б), *Scope2* (в), *Scope3* (г)

***Блок XY Graph* (*XY-графобудівник*)** призначено для побудови фазових портретів, тобто для зображення одного зі збережених у пам‘яті сигналів у функції іншого (а не у функції часу). Його вхідними параметрами є координати меж графіка : *x-min, x-max, y-min* і *y-max*, а також параметр дискретизації *Sample time* (див. рис. 10.9).

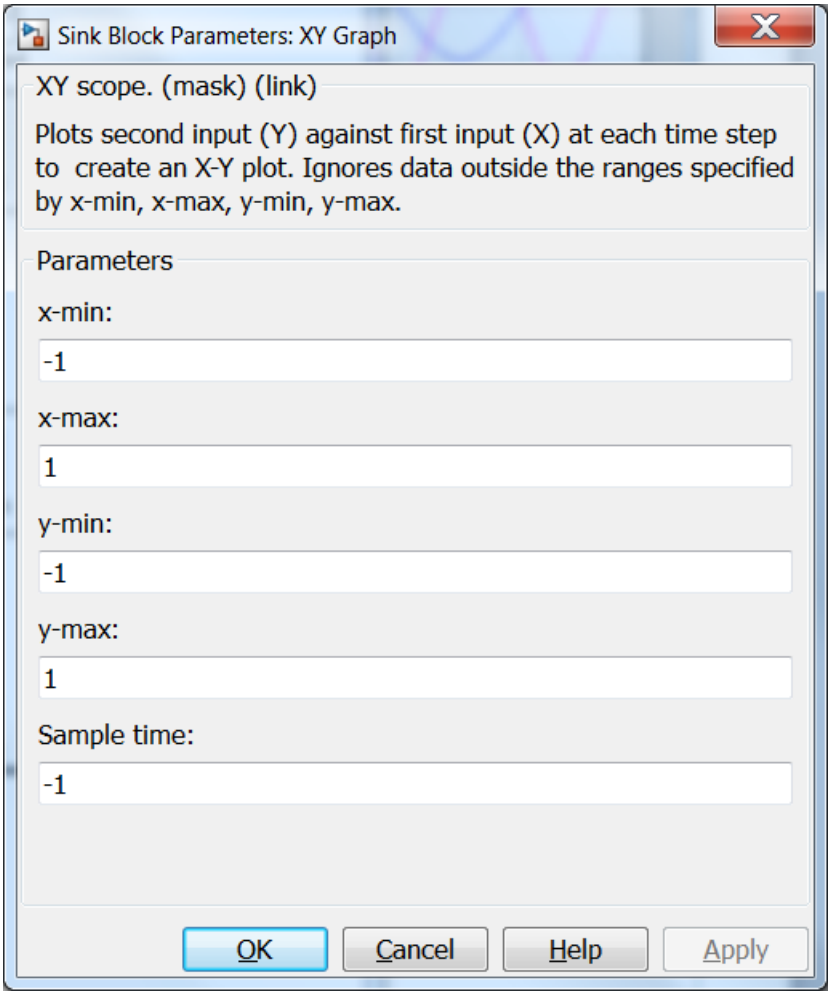


Рис. 10.9. Вікно параметрів блоку *XY Graph*

Обидва входи блоку є скалярними. При моделюванні *Simulink* автоматично відкриває графічне вікно *MATLAB* і динамічно відображає в ньому задану користувачем графічну залежність.

Приклад та результат використання блоку показано на рис. 10.10.

До перегляду графіка треба встановити діапазони зміни його параметрів у вікні рис. 10.9.

Більш якісні графіки залежності одного сигналу від іншого можна побудувати оператором *plot* за результатами вимірювання сигналів.

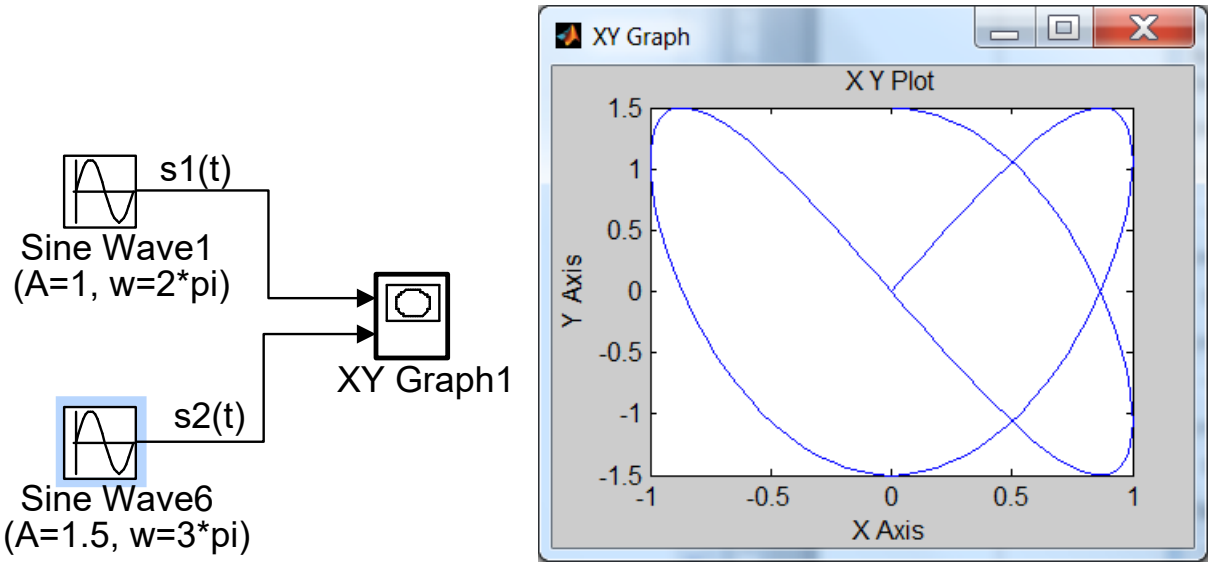


Рис. 10.10. Приклад використання блоку *XY Graph*

Блок ***Display* (*Дисплей*)** застосовується для виведення чисельних значень сигналів у заданому форматі, що обирається за допомогою параметра *Format* і може приймати наступні значення:

short – 4 значущі цифри у форматі з фіксованою крапкою;

long – 14 значущих цифр у форматі з фіксованою крапкою;

short-e – 5 значущих цифр у мантисі чисел із плаваючою крапкою;

long-e – 16 значущих цифр у мантисі чисел із плаваючою крапкою.

Скалярні і векторні сигнали, що надходять до блоку, можуть мати як дійсні, так і комплексні значення.

Якщо розмір блоку *Display* малий для відображення всієї інформації, то в його правому нижньому куті з'являється чорний трикутник, що вказує, у якому напрямку варто розтягти піктограму блоку.

Так само, як і *Осцилограф*, *Дисплей* може працювати в плаваючому режимі.

Приклад використання блоку наведено на рис. 10.11.

Оскільки зазвичай моделювання здійснюється дуже швидко і з малими проміжками часу між розрахунками, то користувач може побачити тільки останнє значення змінної після закінчення симуляції.

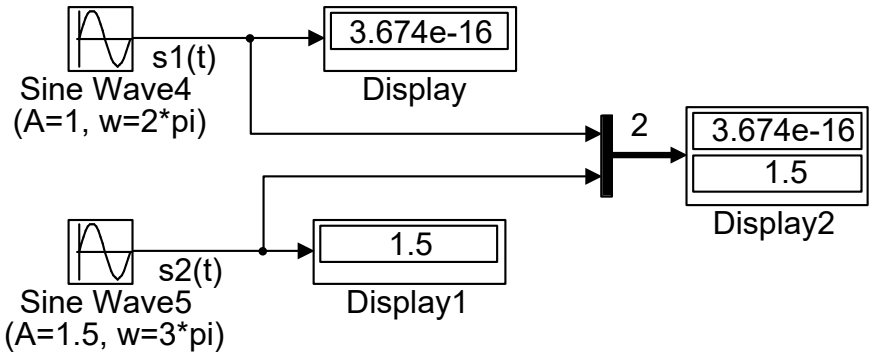


Рис. 10.11. Приклад використання блоку *Display*

10.2 Блоки запам‘ятовування сигналів

Блоками ***Out* (*Вихідний Порт*)** відзначають виходи моделей і підсистем.

У підсистемах вони необхідні для зв'язку підсистеми з моделлю більш високого рівня, а в моделях вищого рівня – для запам‘ятовування приєднаних до них сигналів з метою подальшого використання для побудови графіків або для аналізу чисельних даних.

Для виконання останньої операції необхідно відкрити через функцію *Simulation* вікно *Model Configuration Parameters*, вибрати в ньому вкладку *Data Import/Export* та в розділі *Save to workspace* встановити прапорці в полях *Output* та *Time* і визначити імена змінних, у яких зберігатиметься інформація. За замовчанням у полі *Output* прописано ім‘я yout , а у полі *Time* – ім‘я tout, які, за бажанням користувача, можна змінити на будь-які інші.

Цей блок зручно використовувати для фіксації векторних сигналів.

Пізніше буде показано, що блоки *Out* використовують також для інформування додатків, призначених для аналізу і синтезу систем автоматичного керування (наприклад, *Control Toolbox*) про можливі точки знімання вихідних сигналів.

Скалярні і векторні сигнали, що надходять на вихідні порти можуть мати як дійсні, так і комплексні значення.

Параметри *Output when disabled* і *Initial output* мають значення тільки в підсистемах, виконуваних за умовою.

Блок ***To Workspace* (*У Пам'ять*)** запам'ятовує дані, що надходять на його вхідний порт у перемінній з ім'ям, завданим у поле параметра *Variable name.* Його зручно застосовувати для фіксації змінних з різними іменами.

Для запису інформації у форматі *Array* (*Масив*) треба не забути змінити значення за замовченням параметру *Save Format* (див. рис. 10.12).

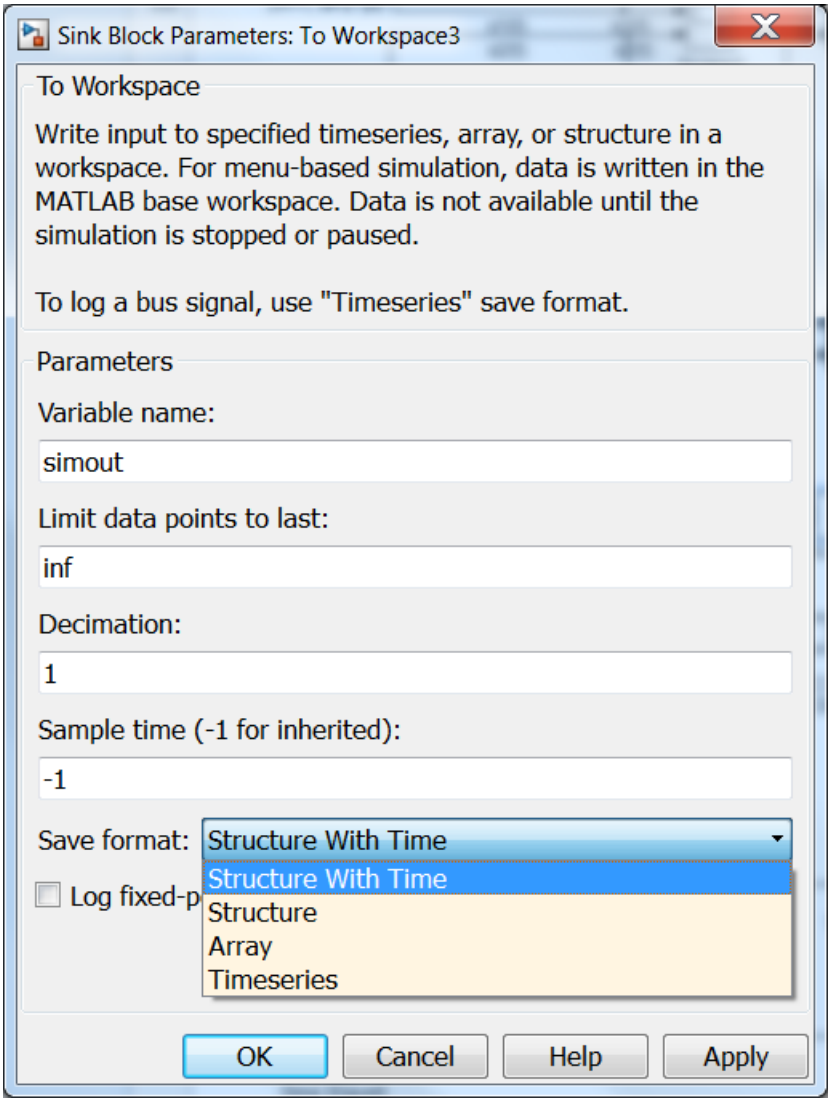


Рис. 10.12. Вікно параметрів блоку *To Workspace*

Усякий раз ***при старті процесу моделювання дані, що зберігаються, обновляються***, так що цей блок не може об'єднати результати декількох послідовних розрахунків. Результати не доступні в робочому середовищі доти, поки не закінчено або не зупинено розрахунок перехідних процесів.

Як бачимо, за замовченням параметр *Variable Name* має значення simout.

***Реєстратор To File* (*У Файл*)** записує часи моделювання і вхідні сигнали у файл з ім'ям *File Name* та розширенням *mat (*.mat)*, який має структуру матриці, що містить у першому рядку монотонно зростаючий вектор часу, а в інших рядках відповідні йому вектори вихідних сигналів:

$$\begin{bmatrix} t_1 & t_2 & \dots & t_k \\ y_1 & y_2 & \dots & y_k \\ z_1 & z_2 & \dots & z_k \\ \dots & \dots & \dots & \dots \end{bmatrix}. \qquad (10.1)$$

Ім'я матриці задається параметром *Variable name*. Для дискретизації сигналів так само, як у блоці *To Workspace*, використовуються параметри *Decimation* і/або *Sample time*.

При кожнім старті процесу моделювання файл даних обновляється.

Якщо якісь частини системи (наприклад, різні пристрої, що задають) працюють незалежно один від одного, то результати моделювання цих підсистем можна записати у файли і зчитувати ці данні з файлів при моделюванні частини системи, що залишилася. При багаторазовому моделюванні це дозволить скоротити час розрахунку перехідних процесів.

Цей блок також рекомендується використовувати в тому випадку, коли для запам'ятовування результатів моделювання не вистачає оперативної пам'яті або коли моделювання займає занадто багато часу. Отриманий файл результатів моделювання можна потім використовувати для побудови графіків за допомогою послідовно з'єднаних блоків *From File* і одного з пристроїв, що реєструють, наприклад, *Scope*. Піктограма блоку відображає ім'я файлу, у який записуються матричні дані.

Матриця, записана у файл, стає доступної в середовищі *Matlab* після завантаження файлу командою load FileName у вигляді значення змінної, ім'я якої визначено в полі параметру *Variable name*.

На рис. 10.13 наведено приклад використання блоків *Out, To Workspace* та *To File* для реєстрації вихідних сигналів двох синусоїдальних сигналів, застосованих вище у моделях рис. 4.7, 4.10 та 4.11.

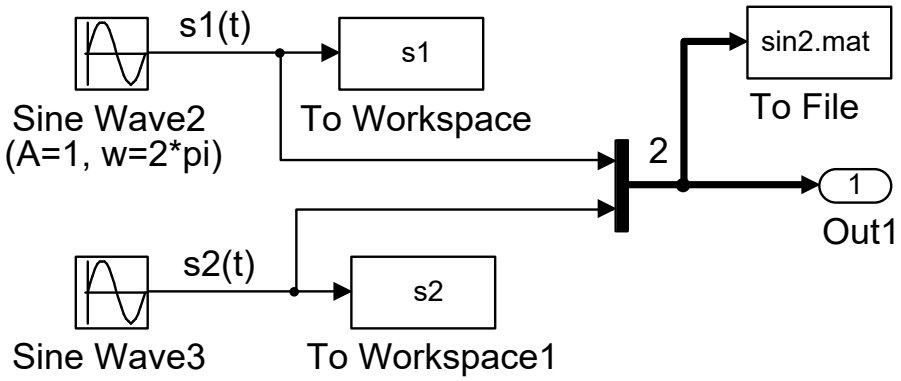


Рис. 10.13. Приклад реєстрації сигналів блоками *Out, To Workspace* та *To File*

Для того, щоб отримати графіки, аналогічні графікам, побудованим за допомогою блоків *Scope* (рис. 10.8) та *XY Graph* (рис. 4.10), використовуючи інформацію, записану у змінні tout, yout, з застосуванням блоку *Out*, треба виконати таку послідовність операторів алгоритмічної мови *MATLAB*:

```
figure (1), plot (tout, yout (:,1)), title (‘s1(t)’), grid on
figure (2), plot (tout, yout (:,2)), title (‘s2(t)’), grid on
figure (3)
subplot (2,1,1), plot (tout, yout (:,1)), title (‘s1(t)’), grid on
subplot (2,1,2), plot (tout, yout (:,2)), title (‘s2(t)’), grid on
figure (4), plot (tout, yout), legend (‘s1(t)’, ‘s2(t)’), grid on
figure (5), plot (yout (:,1), yout (:,2)), title (‘s2(s1)’), grid on
```

Результати виконання цієї програми показані на рис. 4.14.

Такі ж само результати отримаємо, якщо у наведеній вище програмі замінимо yout (:,1) на s1, yout (:,2) на s2 та yout на [s1 s2].

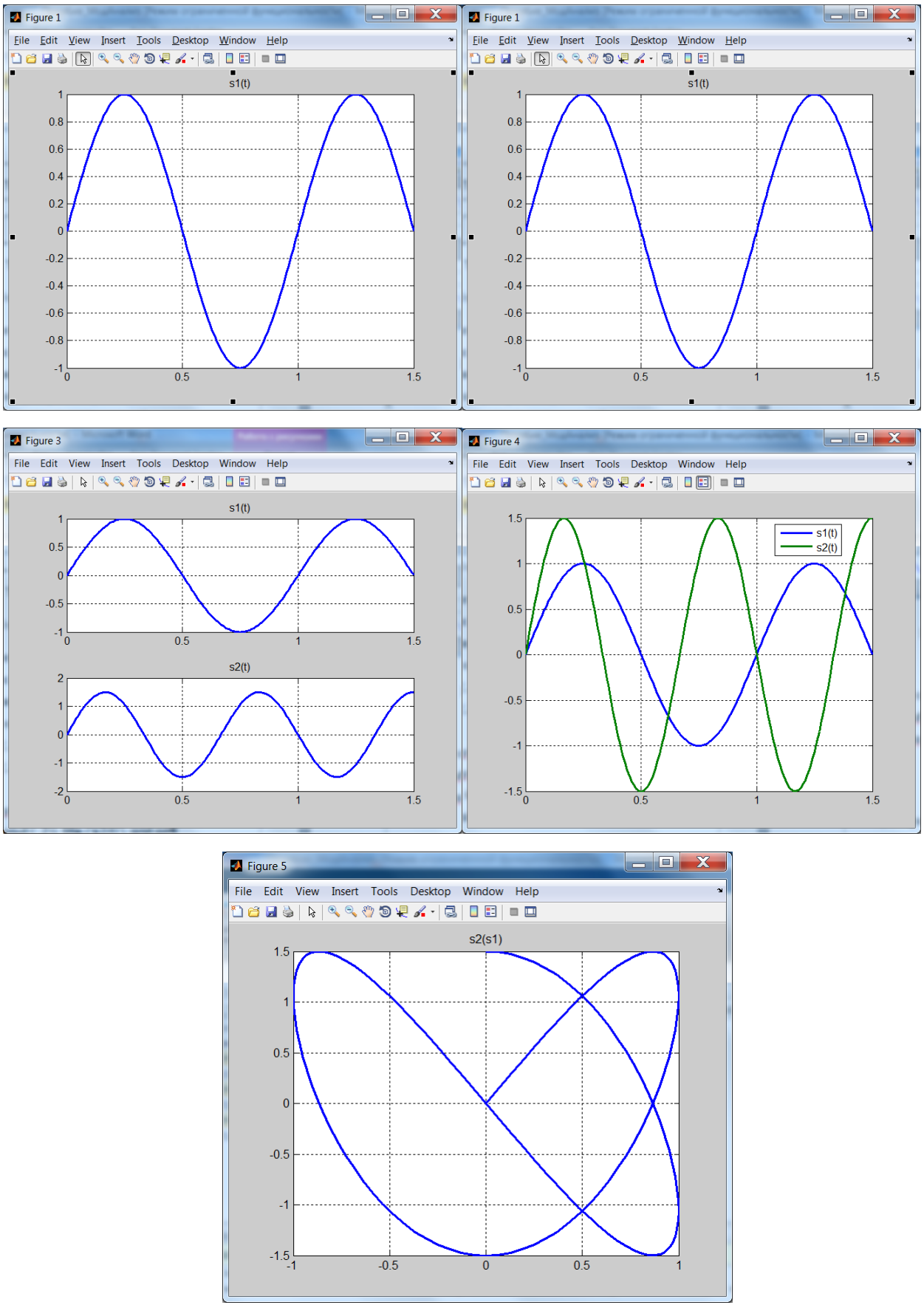


Рис. 10.14. Результати симуляції моделі рис. 4.13
та виконання наведеної вище програми

10.3 Завдання

Створити модель, яка формує синусоїду *s(t)* з одиничною амплітудою та одиничним періодом блоком *Sine Wave*, прямокутний періодичний сигнал *p(t)* з амплітудою 0.8, періодом 1.2с та відносною тривалістю імпульсу 70% блоком *Pulse Generator* і сигнал $k(t) = \sqrt{t}$ послідовним з‘єднанням блоків *Clock* та *Sqrt*. Зафіксувати ці сигнали блоками бібліотеки *Sinks* у такі способи:

- трьома блоками *Scope*,
- одним блоком *Scope* з трьома системами координат,
- одним блоком *Scope* з однією системами координат,
- блоком *Floating Scope*,
- одним (спільним) та трьома (окремими для кожного сигналу) блоками *Out*;
- одним та трьома блоками *To Workspace*,
- блоком *XY Graph*,
- блоками *Display*;

використати одержану інформацію для побудови графіків *s(t), p(t), k(t)* та *s(k).*

10.4 Методичні вказівки та рекомендації

1 Без зайвої потреби не використовуйте вікна моделей, бібліотек, осцилографів у повноекранному режимі.

2 Для швидкого з‘єднання двох блоків виділіть щигликом миші вхідний блок, натисніть на клавішу <*Ctrl*> та виділіть мишею вихідний блок.

3 Деякі операції з блоками зручно робити через меню, що випадає при щиглику по блоку правою клавішею миші.

4 Слідкуйте за наочністю моделі, для чого притримуйтесь таких правил:

- намагайтеся використовувати мінімальну кількість зломів та перетинів ліній зв‘язку;
- не робіть блоки занадто великими (модель повинна бути компактною);
- приховуйте імена блоків, призначення яких зрозуміло з їхніх піктограм;

- якщо це сприяє кращому порозумінню моделі, замінюйте імена стандартних блоків іншими іменами, іменуйте лінії зв‘язку та доповнюйте моделі текстовими коментарями;
- при використанні різних кольорів ретельно продумайте, що саме ви бажаєте виділити тим чи іншим кольором;

6 Для реєстрації часу моделювання застосовуйте один з таких засобів:

- блок *Clock* приєднайте до блоку *To Workspace*, визначивши у полі *Variable name* цього блоку будь-яку змінну, наприклад, *t*.;
- у вкладці *Workspace I/O* вікна *Configuration Simulation Parameters* установіть прапорець *Time* функції *Save to workspace*.

7. При реєстрації даних блоками *To Workspace* і *Scope*, а також за допомогою функції *Save to workspace* вікна *Simulation Parameters* використовуйте формат *Array*. При великій кількості точок у векторі часу моделювання не забувайте збільшити або зробити безкінечним (*Inf*) параметр *Limit data point to last*.
8. Для об‘єднання декількох скалярних сигналів в один векторний використовуйте блок *Mux* бібліотеки *Signals & Systems*. Це дасть змогу скоротити кількість блоків реєстрації.

10.5 Контрольні питання та завдання

1 Охарактеризуйте бібліотечні блоки реєстрації вихідних сигналів, перелічите їхні недоліки та переваги.

2 Як створити багатоканальний блок *Scope* (осцилограф) та багатоканальний блок *Display* (дисплей)?

3 Як масштабуються графіки, відображені блоком *Scope*? Як змінити межі графіків за часом та за вихідною координатою, кольори фону і ліній та інші властивості?

4 Як отримати інформацію про сигнали, приєднані до вихідних портів?

5 Опишіть структуру даних блоків *To Workspace*.

6 Як залишити у пам‘яті сигнали, що візуалізуються блоком *Scope*? У чому полягає різниця між даними у форматі *Array*, записаними за допомогою блоків *Scope* та *To Workspace*?

7 Які імена за замовчанням мають дані, записані у пам‘ять за допомогою блоків *Out*, *To Workspace* та *Scope*? Чи можна їх змінювати?

8 Поясніть призначення параметру *Decimation* блоків *To Workspace*, *To File* і *Scope.*

9 Поясніть призначення параметру *Limit data points to last* блоків *Scope.*

10 Які способи одночасної візуалізації декількох сигналів ви знаєте? Як їх реалізувати?

11 Поясніть призначення блоку *To File* і правила його використання.

Лабораторна робота № 11

ФОРМУВАННЯ ВХІДНИХ СИГНАЛІВ У СЕРЕДОВИЩІ *Simulink*

Мета роботи: навчитися формувати у середовищі різноманітні вхідні сигнали.

Вхідні сигнали, що діють на системи автоматичного керування, здебільш є функціями часу. Їх можна розподілити на такі групи:

1) постійні;

2) ступеневі;

3) лінійні;

4) нелінійні неперіодичні, що описуються аналітичними виразами;

5) гармонічні;

6) періодичні;

7) випадкові.

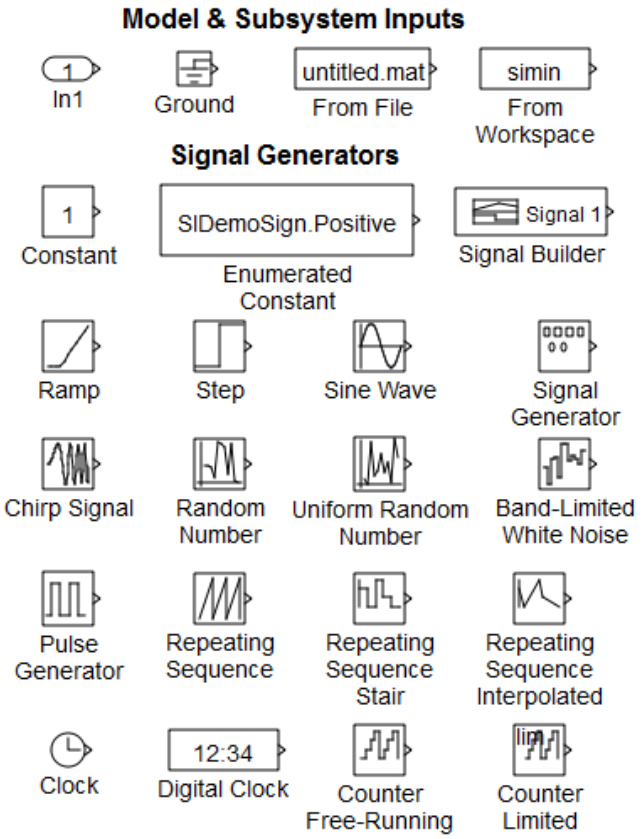


Рис. 11.1. Бібліотека джерел вхідних сигналів *Sources*

11.1 Загальна характеристика блоків бібліотеки *Sources*

Основу формування вхідних сигналів складають блоки бібліотеки джерел *Sources*. У цю бібліотеку входить група блоків, що не мають вхідних портів, а мають тільки виходи. Їхні піктограми подані на рис. 11.1.

Блоки цієї бібліотеки можна розділити на 3 групи:

1) блоки, що не формують вхідних сигналів, а тільки помічають входи (блок *In*) або

заземляють їх (блок *Ground*), щоб при старті моделювання в командному вікні *MATLAB* не виводилося попередження про наявність у моделі неприєднаного вхідного порту (Warning: Input port 1 of block … is not connected);

2) блоки, що можуть формувати тільки один вихідний сигнал (*Ramp, Clock, Digital Clock* і *Repeating Sequence*);

3) блоки, що можуть формувати як один, так і декілька вихідних сигналів (*Const, Step, Signal Generator, Sine Wave, Pulse Generator, Chirp Signal, Random Number, Uniform Random Number* і *Band Limited White Noise*).

Для того, щоб блоки 3-ої групи формували матрицю, складену з одновимірний векторів-стовпців, кількість яких відповідає кількості компонент у векторах параметрів, у вікні настроювання блоку повинен бути встановленим прапорець *Interpret vector parameters as 1-D*. Якщо цей прапорець не встановлено, то блок на кожному кроці чисельного інтегрування формує сигнали розмірність яких повторює розмірність параметрів. Для останнього випадку зберігати дані можна тільки у форматі *Structure with Time*. Візуалізація даних не залежить від стану параметру *Interpret vector parameters as 1-D*.

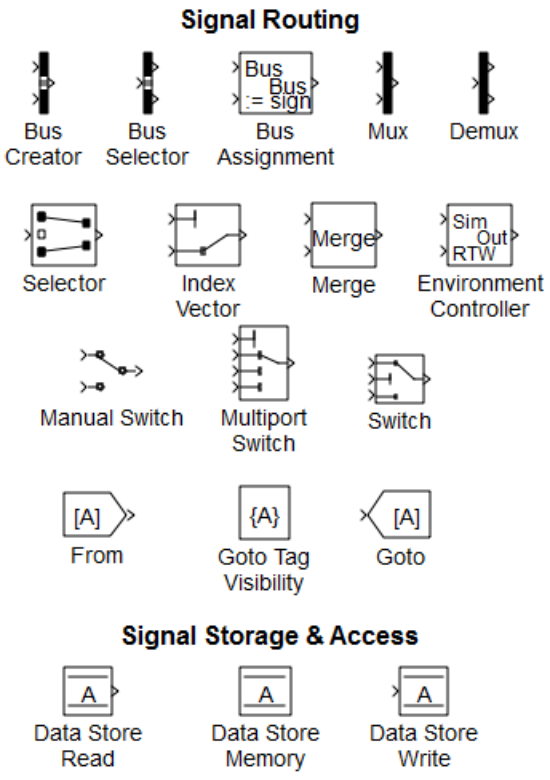


Рис. 11.2. Бібліотека *Signals Routing* (*Маршрутизація сигналів*)

Векторні параметри джерел можуть бути задані у вікнах настроювання блоків як рядками, так і стовпцями за правилами алгоритмічної мови системи *MATLAB*. Векторний сигнал може бути розділений на скалярні сигнали або на векторні сигнали з меншою кількістю компонентів за

допомогою блока *Demux* бібліотеки *Signal Routing* (див. рис. 11.2). Зворотну операцію виконує блок *Mux*.

Такі джерела як *Step, Sine Wave, Pulse Generator, From File, From Workspace, Random Number* і *Uniform Random Number* можуть змінювати значення своїх вихідних сигналів як неперервно, так і дискретно, тобто в моменти часу, кратні періоду дискретності, що задається параметром *Sample Time* (T_S). Для того, щоб джерело працювало у неперервному режимі, значення цього параметра необхідно покласти рівним нулю (T_S=0).

Якщо джерело повинне формувати дискретний сигнал, то в поле параметра *Sample Time* може вводитися одне додатне значення, що інтерпретується як період дискретності *Ts*, або два додатних значення, перше з який сприймається як період дискретності *Ts,* а друге – як зсув (запізнення) *offset.* У цьому випадку зміна вихідних сигналів ланок будуть здійснюватися в дискретні моменти часу

$$t_d = nT_s + offset; \qquad |offset| \le T_s. \tag{11.1}$$

У дискретному режимі кожен блок працює так, як працював би цей же блок у неперервному режимі з приєднаним до його виходу екстраполятора нульового порядку (блок *Zero-Order Hold* бібліотеки *Discrete*) з тим же значенням параметра *Sample Time*. Якщо установити T_S=-1, то джерело працює в тому ж режимі, що і блок, приєднаний до його виходу.

11.2 Основні засоби формування вхідних сигналів

Виходом блоку ***Constant*** **(*Константа*)** є скалярне постійне значення *c*, визначене параметром *Constant value* і не залежне від часу $y = c = const$ або вектор констант у вигляді рядка $\mathbf{y} = [c_1 \quad c_2 \quad ... \quad c_n]$ або стовпця $\mathbf{y} = [c_1 \quad c_2 \quad ... \quad c_n]^{\mathrm{T}}$.

Джерело ***Step*** **(*Стрибок, Сходинка*)** забезпечує стрибкоподібну зміну вихідного сигналу між двома постійними рівнями y_{In} *(Initial value)* і y_{Fin} *(Final value)* у заданий момент часу t_s *(Step time)*:

$$y_{Step}(t) = \begin{cases} y_{In} \text{ при } t \le t_s, \\ y_{Fin} \text{ при } t > t_s. \end{cases} \quad (11.2)$$

Загальний вигляд скалярного ступінчатого сигналу показано на рис. 11.3

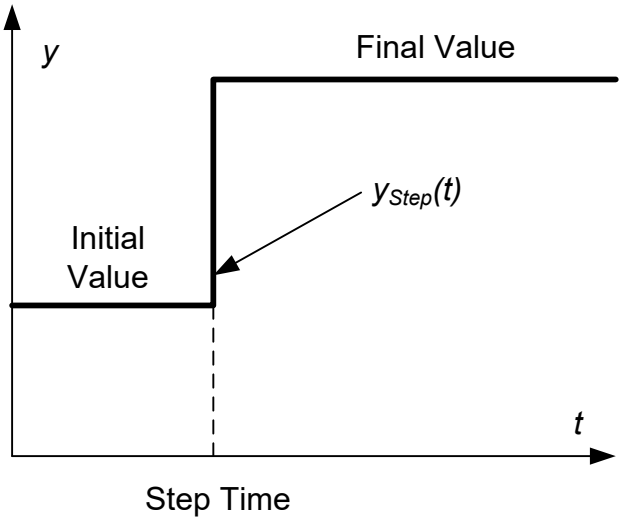


Рис. 11.3. Загальний вигляд скалярного ступінчатого сигналу

Декілька блоків *Step*, приєднаних до входів алгебраїчного суматора *Sum* з математичної бібліотеки *Math operations* створюють багатосходинковий сигнал. Враховуючи можливість блоку *Step* формувати векторні сигнали, та можливість блоків *Sum* та *Add* сумувати усі елементи векторного сигналу, багатосходинковий сигнал можна створити послідовним з‘єднанням одного блоку *Step* із суматором.

Блоки ***Sum* (*Суматор*)** та ***Add (Додавання)*** можуть мати круглу (*round*) або прямокутну (*rectangular*) форму, яка визначається параметром *Icon shape*.

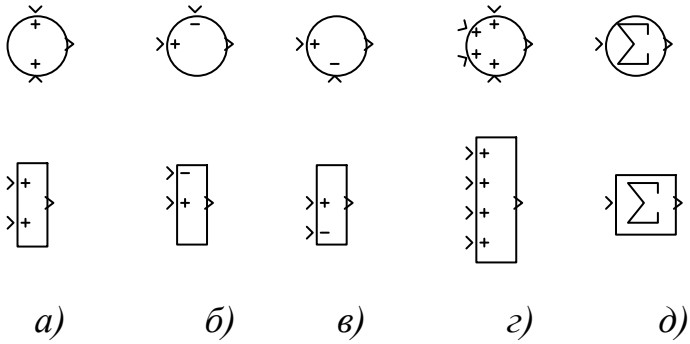
Вони підсумовують вхідні сигнали з відповідними знаками, які встановлюються в полі параметра *List of signs* (входи нумеруються зверху вниз для прямокутного блоку та проти годинної стрілки для круглого блоку). Комбінація знаків “+”, ”–“ описує параметри кожного конкретного порту, де кількість портів відповідає кількості знаків, використаних у комбінації. Для пропуску позиції чергового порту у списку знаків використовують символ „|”.

Якщо замість списку знаків визначити кількість входів, то усі вони будуть підсумовуючими. При одному вході блок підсумовує з відповідним знаком елементи вхідного вектору:

$$y = sum(u) = \sum_{i=1}^{k} u_i \ . \qquad (11.3)$$

При цьому в піктограмі блоку відображається не знак вхідного сигналу, а символ Σ. Блоки *Sum* та *Add* з одним входом можна замінити блоком *Sum of Elements*.

Можливі різновиди блоків *Sum, Add,* в залежності від значень їх параметрів наведені на рис. 11.4.



а) *б)* *в)* *г)* *д)*

Рис. 11.4. Блоки *Sum* та *Add* при таких значеннях параметру *List of signs*:
а) ++; *б)* –+|; *в)* |+–; *г)* 4; *д)* 1 або +

На рис. 11.5 показано багатоступінчатий сигнал, а на рис. 11.6 – варіанти його формування. Параметри блоків *Step* з рис. 11.6 наведені у табл. 11.1.

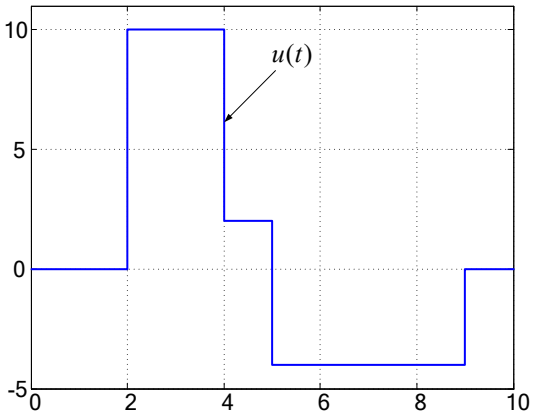


Рис. 11.5. Багатоступінчатий неперіодичний сигнал

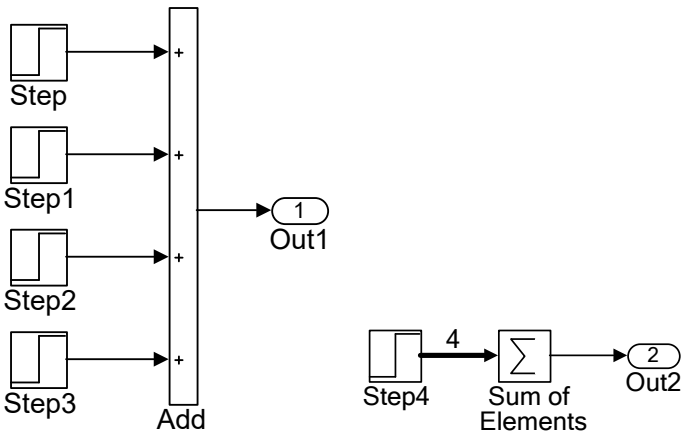


Рис. 11.6. Варіанти формування багатоступінчатого сигналу

Таблиця 11.1

Ім‘я блоку	*Step Time*	*Initial Value*	*Final Value*
Step	2	0	10
Step1	4	0	-8
Step2	5	0	-6
Step3	9	0	4
Step4	[2 4 5 9]	[0 0 0 0]	[10 -8 -6 4]

Блок ***Clock*** **(*Годинник*)** забезпечує відлік і відображення часу моделювання: $y(t)=t$. Параметр *Decimation*, що може приймати цілочисленні додатні значення *d*, має сенс тільки при включеному прапорці *Display time* і при використанні для рішення диференціальних рівнянь методів чисельного інтегрування з постійним кроком *h*. У цьому випадку в піктограмі блоку, що здобуває прямокутну форму, відображаються по черзі чисельні значення часу моделювання, кратні *d*h*.

Блок ***Digital clock*** **(*Цифровий Годинник*)** формує сигнал, який збігається з часом моделювання у моменти часу, кратні періоду дискретності *Sample Time*. Між цими моментами вихідний сигнал має постійне значення, що дорівнює останньому обмірюваному часу моделювання.

Джерело ***Ramp*** **(*Рампа, Похила Площина*)** формує сигнал, що лінійно наростає або спадає від значення y_0 (*Initial output*), починаючи з моменту часу *t* (*Start time*) з коефіцієнтом пропорційності *k* (*Slope*):

$$y(t)=\begin{cases} y_0 \text{ при } t \le t_S, \\ y_0 + kt \text{ при } t > t_S. \end{cases} \qquad (11.4)$$

Загальний вигляд скалярного ступінчатого сигналу показано на рис. 11.7.

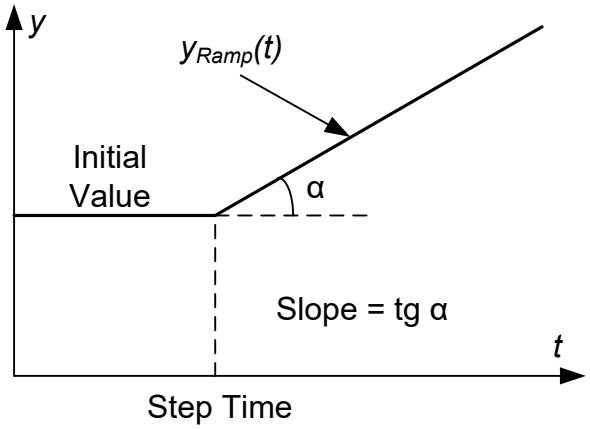


Рис. 11.7. Загальний вигляд скалярного сигналу *Ramp*

Блок ***Sine Wave* (*Синусоїда*)** може працювати в двох режимах (*Time-Based Mode* і *Sample-Based Mode*), котрі встановлюються у вікні настроювання блоку за допомогою випадаючого меню (*Parameters*, *Sine type*). У режимі *Time-Based* можливе формування як неперервного, так і дискретного вихідного сигналу. У неперервному режимі (T_S =0) блок генерує синусоїду у функції часу відповідно до рівняння:

$$y(t) = A\sin(\omega t + \varphi_0) + y_0, \tag{11.5}$$

де

A – амплітуда (*Amplitude*);

ω – кругова частота, рад/с (*Frequency*);

φ_0 – фазовий зсув, рад. (*Phase*);

y_0 – вертикальний зсув (*Bias*).

У дискретному *Time-Based* режимі (T_S >0) при обчисленні вихідного сигналу використовуються формули:

$$\begin{cases} \sin(t + T_S) = \sin(t)\cos(T_S) + \sin(T_S)\cos(t), \\ \cos(t + T_S) = \cos(t)\cos(T_S) - \sin(t)\sin(T_S), \end{cases} \tag{11.6}$$

які дозволяють формувати вихідне значення значно швидше, ніж у неперервному режимі. Недоліком цього методу є нагромадження похибок округлення, тому що для обчислення вихідного сигналу в кожний наступний момент часу використовується значення вихідного сигналу в попередній

момент часу. Для ліквідації цього недоліку ланка *Sine Wave* передбачає можливість роботи в режимі *Sample-Based*. При цьому значення вихідного сигналу обчислюється за формулою

$$y = A\sin\left(2\pi\left(k + o\right)/\, p\right) + y_0, \qquad (11.7)$$

де

p – кількість розрахункових крапок на періоді хвилі синусоїди (*Samples per period*);

o – зсув (відносне фазове зрушення) сигналу (*Number of offset Samples*);

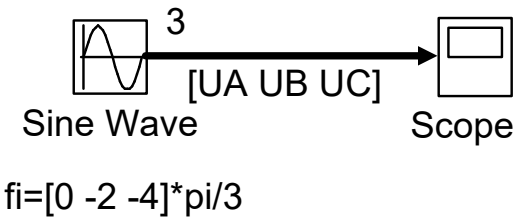
k – цілочисельний параметр, що приймає на періоді синусоїди значення [0, 1, 2, …, p-1]...

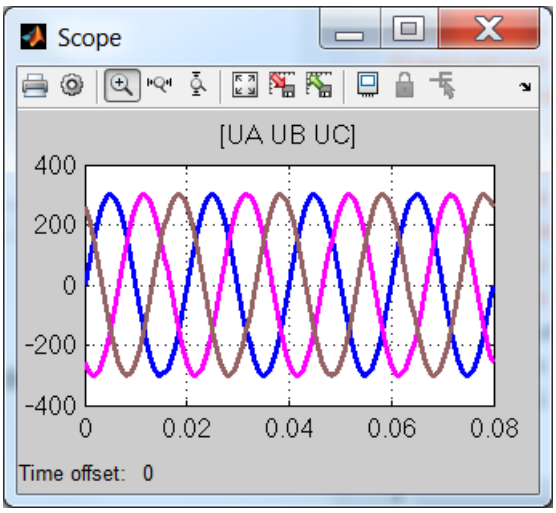
Для узгодження методів *Time-Based discrete* і *Sample-Based* необхідно забезпечити наступні співвідношення між їхніми параметрами:

$$p = \frac{2\pi}{\omega T_S}; \qquad o = \frac{p\varphi_0}{2\pi}. \qquad (11.8)$$

Ліквідуючи нагромадження похибок округлення, формула (11.7) має схований недолік, що виявляється при використанні цього блоку в підсистемі, що виконується за умовою (*Conditionally Executed Subsystem*). Цей недолік полягає в можливості неузгодженої роботи блоку *Sine Wave* з іншими блоками моделі після виконання підсистемою паузи з наступним поновленням її роботи. Отже, при виникненні потреби формування дискретної синусоїди в складі підсистеми, виконуваної за умовою, блок *Sine Wave* необхідно використовувати в режимі *Time-Based*.

Не варто забувати, що блок *Sine Wave* здатний формувати векторні сигнали. Цю властивість зручно використовувати при моделюванні трифазних кіл змінного струму. Для прикладу на рис. 11.8 наведено модель формування трифазних напруг змінного струму частотою 50 Гц і амплітудою 200 В, зсунутих одна від одної на кут 120 градусів і результат її симуляції.





а) *б)*

Рис. 11.8. Модель (а) формування трифазних напруг та результати її симуляції (б)

Блок ***Pulse Generator* (*Генератор Прямокутних Імпульсів*)** також може працювати в неперервному (*Time-Based*) або дискретному (*Sample-Based*) режимах, вибір одного з яких здійснюється за допомогою параметра *Pulse Type*. В обох режимах блок формує періодичний сигнал, що складається з однополярних (додатних) прямокутних імпульсів. В неперервному режимі він має наступні параметри:

ht – висота імпульсу (*Amplitude*);

T – період в одиницях виміру часу (*Period(secs)*);

du – ширина імпульсу у відсотках від періоду (*Pulse width (in % of period)*);

stt – час початку пульсацій (*Phase delay (secs)*).

У дискретному режимі у вікні настроювання блоку з'являється додатковий параметр *Sample Time* , а період і час початку пульсацій задаються у відносних одиницях (частках періоду дискретності – *Number of samples*).

Якщо при моделюванні обрано метод рішення диференціальних рівнянь з фіксованим кроком, то *Simulink* автоматично використовує блок *Pulse Generator* у режимі *Sample-Based* з періодом дискретності, рівним кроку інтегрування. У цьому випадку крок інтегрування повинний бути обраний таким, щоб параметри *T, du*T/100* і *stt* були кратні йому.

Блок ***Repeating Sequence*** **(*Повторювана Послідовність*)** виконує повторення циклу, заданого таблицею двох параметрів: *Time values* (t_v)– вектор часу і *Output values* (y_v) – вектор вихідних сигналів. Вектор часу повинен бути монотонно зростаючим. Різниця між значеннями його останнього і першого елементів визначає період вихідного сигналу: *T=tv*(*end*)-*tv*(1). Блок перемальовує свою піктограму відповідно до вигляду вихідного сигналу. Він реалізований за допомогою замаскованої підсистеми зображеної на рис. 11.9.

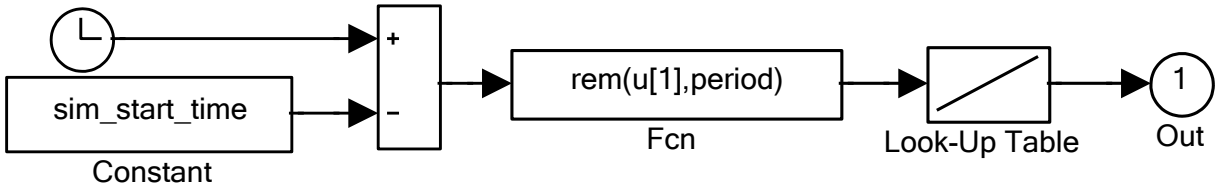


Рис. 11.9. Модель замаскованого блоку *Repeating Sequence*

Блок *Fcn* моделі рис. 11.10 визначає залишок від цілочисельного ділення часу моделювання на період, а блок *Look-Up Table* виконує кусочно-лінійну апроксимацію табличної функції, заданої параметрами *Vector of input values* і *Table data*.

Джерело ***Chirp Signal*** **(*Синусоїда зі Змінною Частотою*)**, реалізоване за допомогою замаскованої підсистеми рис. 11.10, формує синусоїду з одиничною амплітудою і частотою, що змінюється за лінійним законом:

$$y = \sin(At^2 + Bt), \tag{11.9}$$

де

$$B=2\,\pi f_1\ , \quad A=2\,\pi\,(f_2 - f_1)\,/\,T \tag{11.10}$$

за вхідними параметрами:

f_1 (*Initial frequency* [*Hz*]) – початкова частота [Гц];

T (*Target time* [*secs*]) – заданий час [с];

f_2 (*Frequency at target time* [*Hz*]) –частота в заданий момент часу [Гц].

Піктограма джерела зображує графік функції $y=sin(t^2)$.

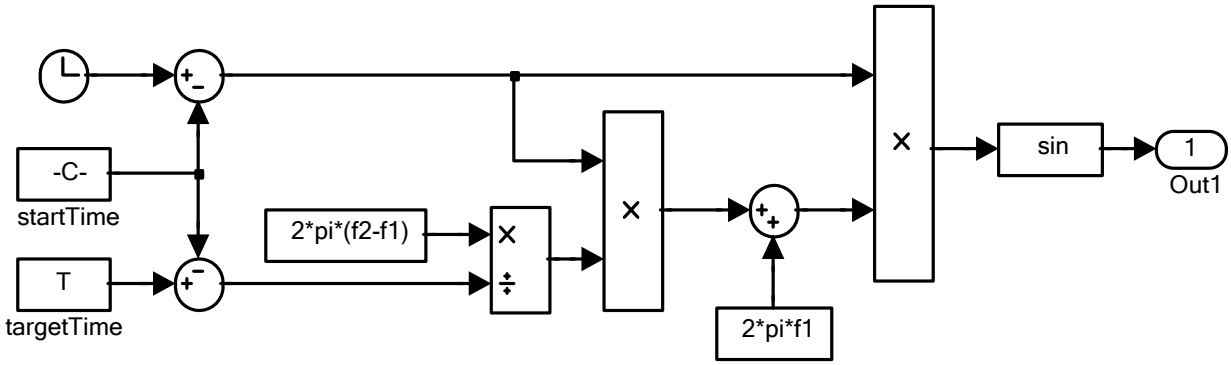


Рис. 11.10 – Модель блоку *Chirp Signal*

Комплексне джерело ***Signal Generator*** **(*Генератор Сигналів*)** генерує один з 4-х сигналів: синусоїду (*Sine*), прямокутний (*Square*) періодичний сигнал зі шпаруватістю 50%, пилкоподібний (*Sawtooth*) періодичний сигнал, а також випадковий сигнал (*Random*).

Тип сигналу вибирається за допомогою випадаючого меню (*Parameters*, *Wave form*). Чисельними параметрами являються амплітуда *A* (*Amplitude*) і частота (*Frequency*). Частота може задаватися як у Гц (*Hertz*), так і в рад/с (*rad/sec*) за допомогою параметра *Units*. Значення всіх сигналів змінюються від *+A* до *–A*. Вихідні установки блоку можуть бути змінені протягом процесу моделювання, що особливо ефективно у сполученні з фіксацією результатів блоком *Scope* (*Осцилограф*), коли реакцію системи на різні типи вхідного сигналу потрібно одержати швидко.

Випадкові сигнали створюються генераторами випадкових чисел.

Джерела випадкових чисел з нормальним **(*Random Number*)** та рівномірним **(*Uniform Random Number*)** розподілами генерують відповідно нормально розподілений (Гаусовський) і рівномірно розподілений випадкові сигнали для деяких заданих стартових чисел *Initial seed*, що ініціалізують генератори випадкових чисел.

Специфічними параметрами ланки *Random Number* є середнє значення сигналу *Mean* і середньоквадратичне відхилення *Variance*, а ланки *Uniform Random Number* – мінімальне *Minimum* і максимальне *Maximum* значення випадкового сигналу. Обидва блоки можуть працювати як в неперервному, так

і у дискретному режимах.

Джерело ***Band-Limited White Noise*** **(*Білий Шум з Екстраполяцією*)** забезпечує формування “білого шуму” для неперервних систем за допомогою послідовно з'єднаних блоків *Random Number*, що працює в дискретному режимі, та *Gain* (*Пропорційна ланка*) на основі таких параметрів:

Cov (Noise Power) – потужність шуму (коваріація);

T_s *(Sample Time)* – період дискретності;

Seed – стартове число генератора випадкових чисел.

Коефіцієнт підсилення ланки *Gain* обчислюється за формулою

$$k = \sqrt{Cov} / \sqrt{T_s} \,. \qquad (11.11)$$

Вектори *Noise Power* і *Seed* можуть бути однієї довжини. Для більш швидкого протікання процесу моделювання необхідно параметр *Sample Time* установлювати максимально великим, узгоджуючи однак його величину з мінімальної сталою часу системи.

Блок ***From Workspace*** **(*З пам'яті*)** забезпечує читання даних з матриці, що повинна мати два або більш стовпців. Перший стовпець повинний містити монотонно зростаючі значення часу, а кожен додатковий стовпець – табличні значення функцій часу:

$$\begin{bmatrix} t_1 & y_1 & z_1 & \dots & w_1 \\ t_2 & y_2 & z_2 & \dots & w_2 \\ \dots & \dots & \dots & \dots & \dots \\ t_k & y_k & z_k & \dots & w_k \end{bmatrix}. \qquad (11.12)$$

Наприклад, для того, щоб блок *From Workspace* сформував на інтервалі $t \in [0,\, 2\pi]$ сигнал $y(t) = sin(t) + cos(t^2)$, у середовищі *MATLAB* до моделювання необхідно виконати послідовність операцій

```
Time = (0 : pi/50 : 2*pi)';
Out = sin (Time) + cos (Time.^2);
D = [Time  Out];
```

і установити в поле параметра *Data* у вікні настроювання блоку *From Workspace* ім'я матриці D.

Значення часу використовуються блоком для обчислення вихідного сигналу за допомогою лінійної інтерполяції або екстраполяції.

Піктограма блоку відображає ім'я матриці, з якої зчитуються його дані.

За допомогою випадаючого меню *Form output after final data value by* можна вибрати один з чотирьох способів зміни вихідного сигналу після перевищення часом моделювання значення останнього елемента першого стовпця матриці даних (максимального значення аргументу табличної функції):

Extrapolation – лінійна екстраполяція;

Setting To Zero – установка в нуль;

Holding Final Value – затримка останнього табличного значення;

Cyclic Repetition – циклічне повторення.

Для використання першого способу прапорець інтерполяції *Interpolate data* повинний бути обов'язково встановлений, а для останнього способу – скинутий. Для інших способів стан цього прапорця не має значення. При останньому способі (*Cyclic Repetition*) дані, що зчитуються, повинні мати формат не матриці, а структури. Наприклад, щоб розглянутий у попередньому прикладі сигнал циклічно повторювався при $t > 2\pi$, необхідно сформувати структуру даних у такий спосіб:

```
D.Time = (0 : pi/50 : 2*pi)';
D.signals(1).Out=sin (Time)+cos (Time.^2);
D.signals(1).dimensions = 1;
```

Останній параметр визначає розмірність вихідного сигналу.

Джерело ***From File*** **(*З Файлу*)** забезпечує читання даних з файлу, чиє ім'я занесено в поле параметра *Filename* і відображається в піктограмі блоку.

Цей формат ідентичний формату блоку *To File*, що забезпечує сумісність файлів.

Запис даних у *mat*-файл у пакеті *MATLAB* виконує команда

```
save FileName var1 var2 …
```

Наприклад,

```
save data1 D
```

збереже значення змінної D у файлі *data1.mat.* Для визначення вихідних сигналів у моменти часу, відсутні у файлі даних, використовується лінійна інтерполяція або екстраполяція. Файл даних для блоку *From File* може бути сформований блоком *To File* при моделюванні іншої системи.

Блок ***In* (*Вхідний Порт*)** помічає входи моделі. Якщо у вкладці *Workspace I/O* вікна *Simulation Parameters* установити прапорець *Input* функції *Load from Workspace*, то зовнішній вхідний сигнал можна задавати в полі параметру *Input* у такий же спосіб, як за допомогою блоку *From Workspace*.

Основним параметром блоку *In* є номер порту (*Port number*), що відображається в його піктограмі. Усі порти в межах одного вікна повинні мати послідовну нумерацію. Номера портів формуються автоматично в порядку їхньої появи у вікні. При видаленні порту з вікна порти, що залишилися в ньому, перенумеровуються таким чином, щоб у послідовності номерів не було пропусків. Номера портів можна змінювати і вручну.

Параметром *Port dimensions* можна визначити розмірність вхідного сигналу. Значення за замовчуванням (-1) відповідає автоматичній установці цього параметра.

Тип сигналу (*real, complex*), тип представлення даних (*double, single, int8, …, boolean*), прапорець *Interpolate data*, а також параметр *Sample time* мають сенс тільки при використанні зовнішніх джерел у моделях верхнього рівня.

Додаткові засоби формування нелінійних вхідних сигналів будуть розглянуті при знайомстві з нелінійними блоками і функціональними перетворювачами бібліотек *Simulink*.

11.3 Використання функціональних блоків при формуванні вхідних сигналів

Якщо треба сформувати функцію часу, не передбачену блоками бібліотеки Sourses, використовують послідовне з‘єднання блоку Clock з одним із функціональних блоків, що будуть розглянуті нижче.

Нелінійні статичні характеристики, задані аналітичними виразами, в *Simulink* можна реалізувати функціональними блоками *Abs, Sign, Sqrt, Signed Sqrt, Reciprocal Sqrt, Math Function, Trigonometric Function* і *Polynomial* з бібліотеки *Math Operations* і блоками бібліотеки *User defined functions*.

Блоки бібліотеки *Math Operations* подані на рис. 11.11.

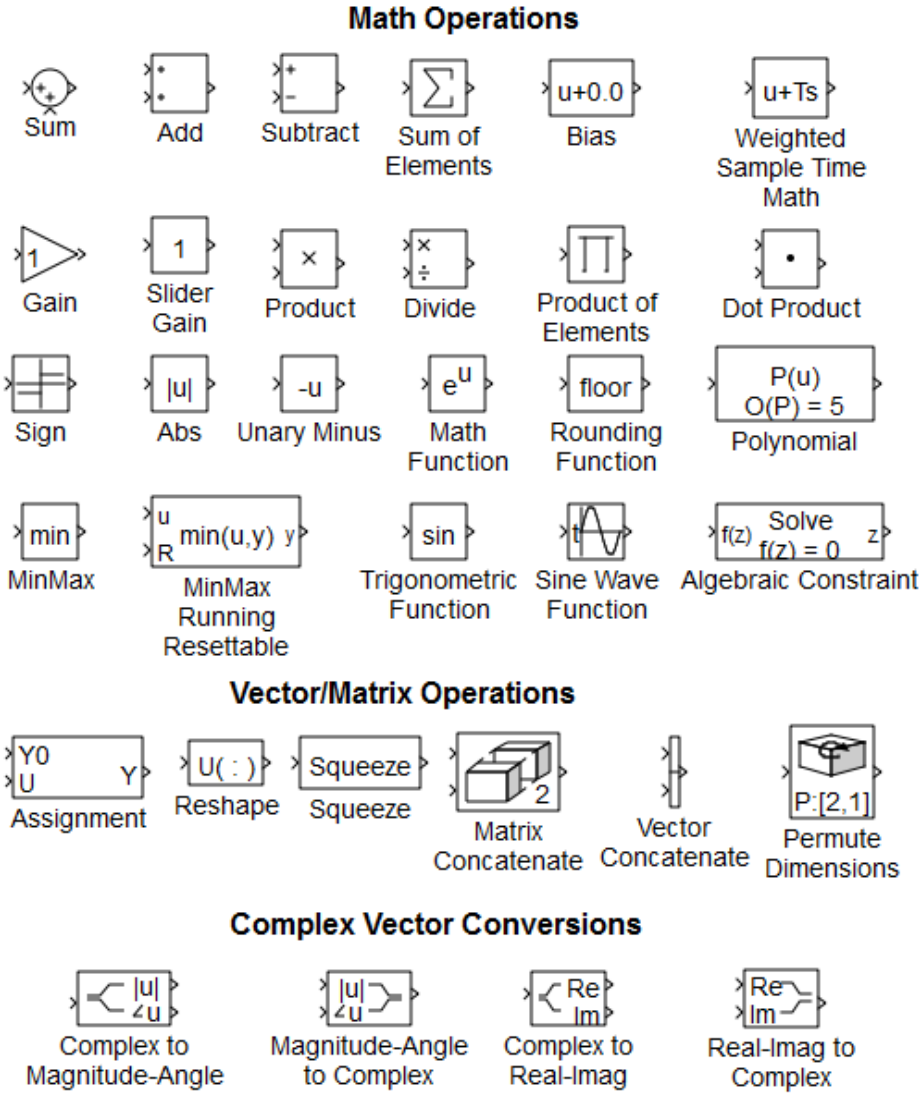


Рис. 11.11. Бібліотека математичних блоків *Math Operations*

Виходом ланки ***Abs* (*Модуль*)** є абсолютне значення входу: $y = |u|$.

Блоки ***Sqrt, Signed Sqrt,*** та ***Reciprocal Sqrt*** – це однакові блоки, які в залежності від вибору параметру *Function*, який може приймати значення *Sqrt, ReSqrt* та *SignSqrt*, виконують такі операції:

$$y = \sqrt{u}\,, \tag{11.13}$$

$$y = 1/\sqrt{u}\,. \tag{11.14}$$

та

$$y = \sqrt{|u|}\cdot sign(u)\,. \tag{11.15}$$

Блок ***MinMax* (*Мінімум* / *Максимум*)** визначає мінімальний або максимальний (за вибором користувача) із вхідних сигналів, кількість яких задається **параметром *Number of input ports*.**

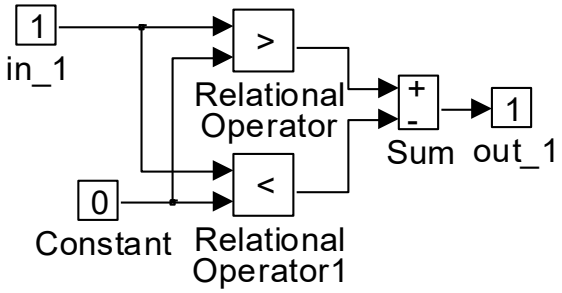


Рис. 11.12 – Структура блоку *Sign*

Блок ***Sign* (*Знакова Функція*)** формує вихідний сигнал за структурою рис. 11.12, величина якого визначається знаком вхідного сигналу:

$$y = \begin{cases} 1 & \text{при} \quad u > 0, \\ 0 & \text{при} \quad u = 0, \\ -1 & \text{при} \quad u < 0. \end{cases} \tag{11.16}$$

Блок ***Trigonometric Function*** відтворює одну із елементарних тригонометричних (*sin*, *cos*, *tan*), зворотних тригонометричних (*asin*, *acos*, *atan*, *atan2*), гіперболічних (*sinh*, *cosh*, *tanh*), та зворотних гіперболічних функцій (*asinh*, *acosh*, *atanh*) від вхідних сигналів шляхом вибору значення параметра *Function*.

Блок ***Polynomial* (*Поліном*)** обчислює значення степеневого полінома, в якому незалежною змінною є вхідний сигнал u, а вектор коефіцієнтів $\mathbf{A} = [a_n, a_{n-1}, ..., a_1, a_0]$ задається у полі параметра *Polynomial coefficients*:

$$P_n(u, \mathbf{A}) = a_n u^n + a_{n-1}u^{n-1} + ... + a_1 u + a_0\,, \tag{11.17}$$

що відповідає виконання в MATLAB оператора y=polyval(A,u).

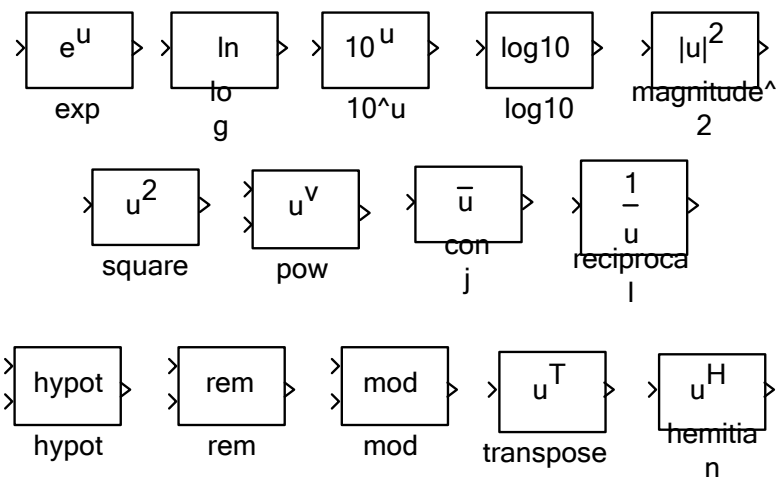


Рис. 11.13. Блоки *Math Function*

Блок ***Math Function*** може відтворювати елементарні одно- та двоаргументні функції від вхідних сигналів. Вибір функції здійснюється через меню. Він впливає на вигляд піктограми блоку. На рис. 11.13 наведені можливі різновидності цього блоку. Для наочності імена блоків замінені іменами функцій меню. Двоаргументна функція *hypot (u,v)* здійснює операцію

$$y_i = \sqrt{u_i^2 + v_i^2}\,, \qquad i = 1, 2, ..., n\,, \tag{11.18}$$

функція *mod (u,v)* округляє результат ділення першого аргументу на другий до найближчого цілого з недостачею, а функція *rem (u,v)* – визначає остачу від цілочисленного ділення цих аргументів. Інші функції не потребують пояснень.

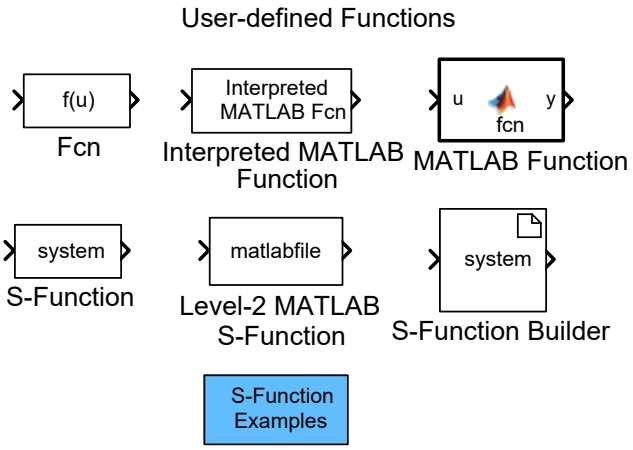


Рис. 11.14 – Бібліотека *User defined functions*

Блоки бібліотеки показані на рис.11.14.

Вихідний сигнал блоку ***Fcn*** **(*Функція*)** є аналітичною функцією, при побудові якої можуть бути використані такі компоненти:

- *u*[*i*] або *u*(*i*) – значення і-го вхідного сигналу (*u* без індексу еквівалентно *u*(1));
- числа;
- операції +, -, *, /, ^;

- дужки ( );
- елементарні функції *sin*, *cos*, *tan*, *asin*, *acos*, *atan, sinh, cosh, tanh, exp, ln, log10, sqrt, floor, ceil, abs*, *atan2, rem*;
- операції порівняння ==, ~=, >, <, >=, <=;
- скалярні неіндексовані змінні.

Наприклад, *sin*(4**u*).**exp*(-*u*/*k*), *rem*(*u*(1), *u*(2)), *u*(1)**u*(2)>=*u*(3)+*u*(4).

Блок ***Interpreted MATLAB Fcn*** **(*Інтерпретована МАТЛАВ-Функція*)** обчислює задану *MATLAB*-функцію (стандартну або створену користувачем) від вхідного сигналу *u* (скалярного або векторного). Цей блок є пасивним двобічним інтерфейсом між середовищами *Simulink* і *MATLAB*. Наприклад, якщо задана функція *sin*, то вихідний сигнал *y* буде обчислюватися по формулі *y*=*sin*(*u*). Якщо вхідних сигналів декілька, то вони позначаються як *u*(*1*), *u*(*2*), *u*(*3*) і т.д.

Блок *Interpreted MATLAB Fcn* підтримує векторні входи і виходи, які перетворюються у скалярні блоком *Demux*, а навпаки – блоком *Mux*. Кількість входів повинна співпадати з кількістю вхідних аргументів *MATLAB*-функції, а розмірність виходу (*Output width*) – з кількістю її вихідних аргументів. Якщо користувач задає параметр *Output width* рівним -1, то *Simulink* установлює розмірність вихідного сигналу рівною розмірності вхідного сигналу.

Блок ***MATLAB Fcn*** **(*МАТЛАВ-Функція*)** включає в *Simulink*-модель *C*-код сформованої користувачем *MATLAB*-функції, текстовий редактор для формування якої з заданим шаблоном відкривається при подвійному щиглику мишею на іконці блоку:

```
1    function y = fcn(u)
2    %#codegen
3
4    y = u;
```

11.4 Використання табличних функціональних перетворювачів при формуванні вхідних сигналів

Якщо треба сформувати функцію часу, задану у вигляді таблиці, можна використати послідовне з‘єднання блоку Clock з блоком ***Look-Up Table*** із бібліотеки ***Look-Up Tables***.

Блок ***1D-Look-Up Table*** **(*Функціональний Перетворювач*)** забезпечує кусочно-лінійну апроксимацію табличної функції, заданої векторами аргументів *(Breakpoints)* і значень *(Table data)* однакової довжини: $y = f(u)$;

$$y = \frac{y_i - y_{i-1}}{u_i - u_{i-1}} \cdot (u - u_{i-1}) + y_{i-1}, \qquad (11.19)$$

де i=1...n – номера вузлових точок таблиці.

Вектор аргументів повинен бути монотонно зростаючим. Вихідні значення для значень входу, що попадають за межі заданих, знаходять за допомогою екстраполювання за першими (або останніми) двома точками.

Піктограма блоку *Look-Up Table* відображає графік залежності вхід-вихід. При зміні значень параметрів у діалоговому вікні блоку графік автоматично перемальовується. Якщо ж параметр заданий перемінною, значення якої змінюється в робочому середовищі пакета *MATLAB*, то характеристика блоку змінюється відразу, але перемальовується тільки після відкриття його діалогового вікна.

11.5 Завдання

1. Сформуйте багатоступеневі сигнали, зображені на рис. 11.15.
2. Ознайомтеся з блоком *Look-up Table*. Сформуйте довільну кусково-лінійну функцію часу із 4-6 ділянок.
3. Створити модель, що складається з блоків формування періодичних сигналів бібліотеки *Sources*, приєднаних через ключовий елемент *Multiport Switch* до блоків бібліотеки *Sinks*; ознайомитися з принципами роботи перелічених блоків, змінюючи їхні параметри.

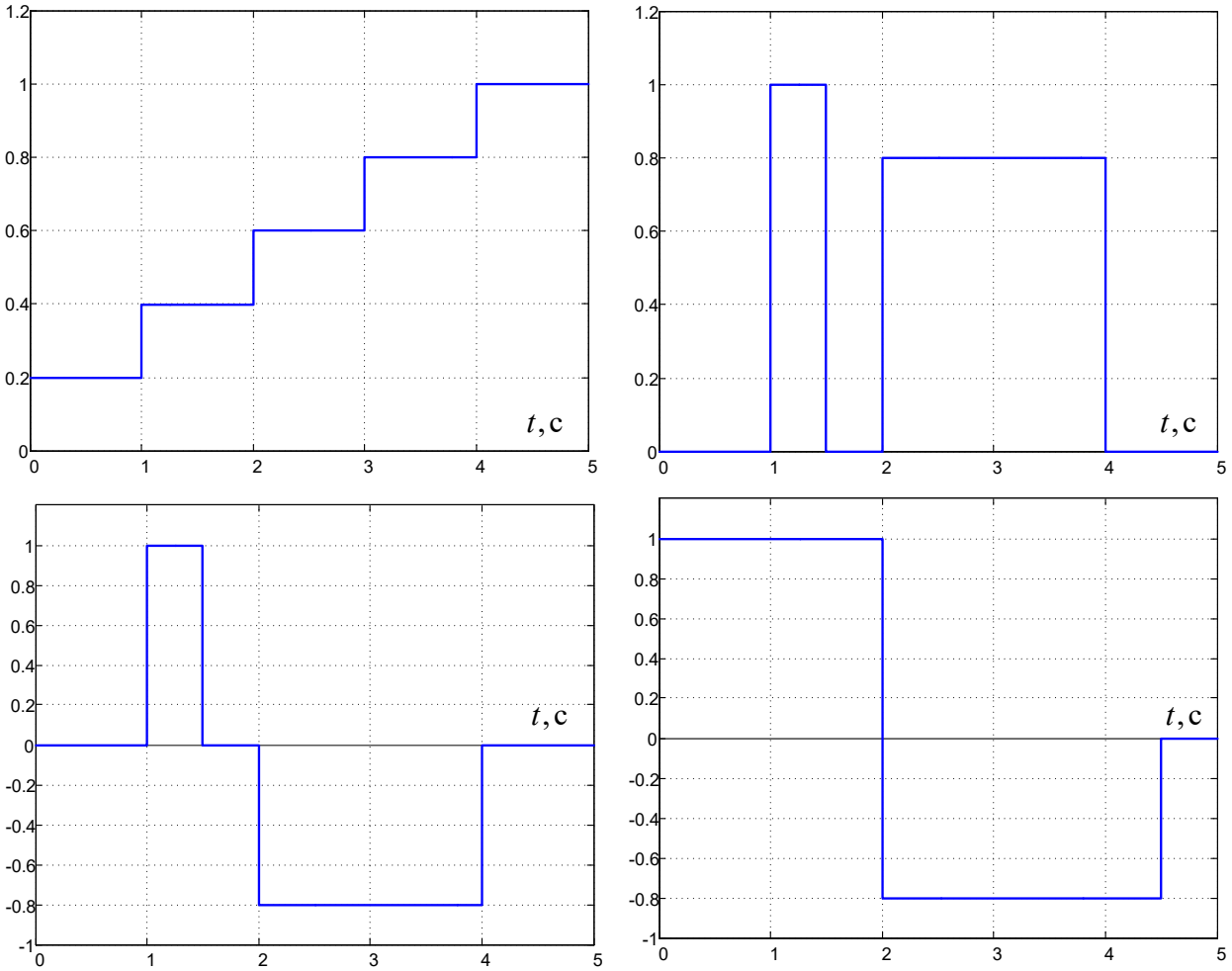


Рис. 11.15. Багатоступінчаті сигнали до завдання 1

4. Сформувати вхідні сигнали, подані на рис. 11.16, за даними табл. 11.1.
5. Сформувати вхідні сигнали за формулами, поданими у табл. 11.1. ($U = f(t)$).

11.6 Методичні вказівки та рекомендації

1. При формуванні першого із багатоступінчатих сигналів пункту 1 застосовуйте не *Step*, а *Quantizer*.
2. При використанні блоку *Look-up Table* не забувайте видалити в параметрах функцію atan.
3. При виконанні пункту 3 підбирайте параметри блоків так, щоб для їх симуляції можна було б використати однаковий час перехідного процесу. Час перехідного процесу оберіть таким, щоб він дорівнював 3÷5 періодам досліджуваних функцій.

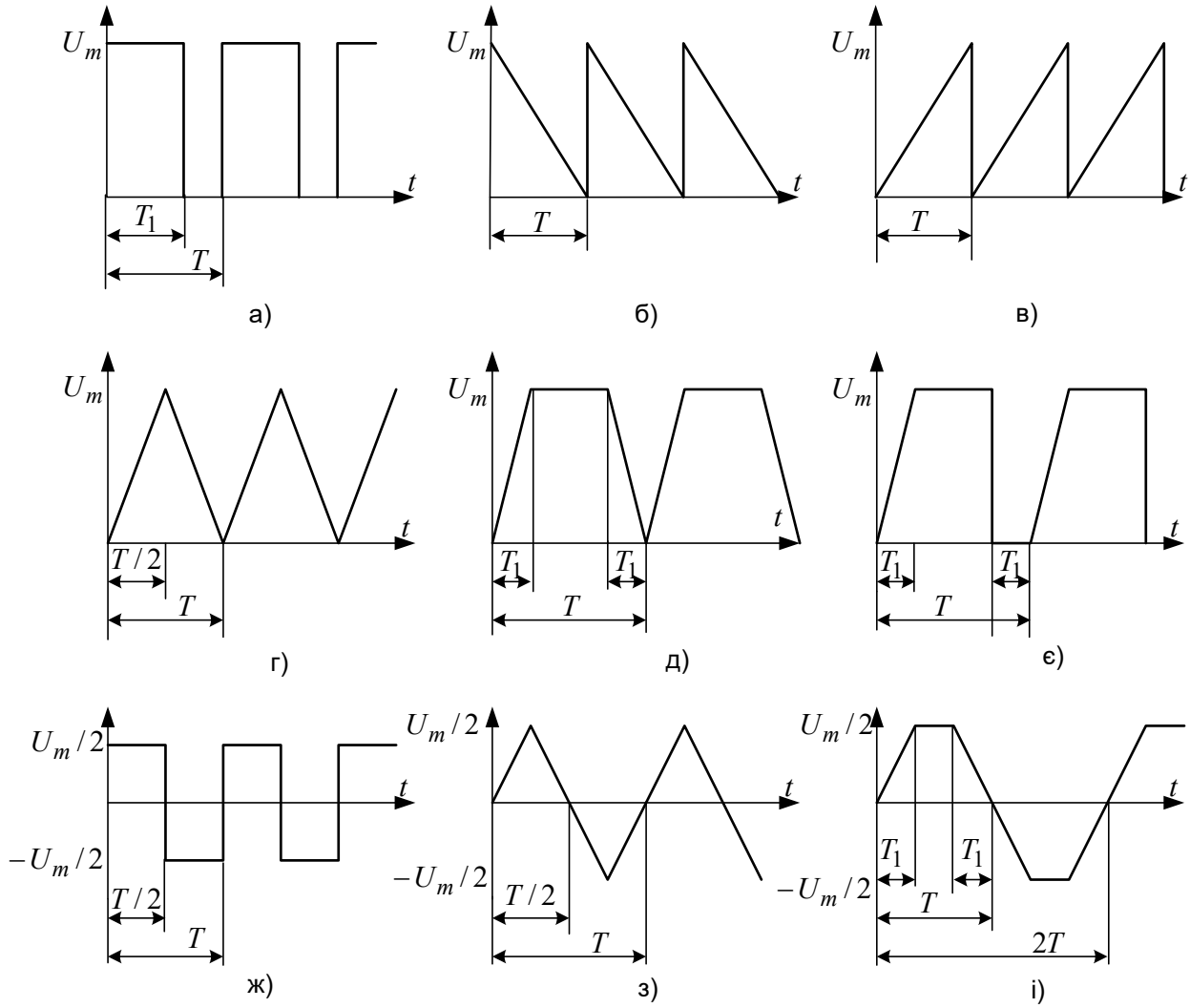


Рисунок 11.16 – Графіки вхідних сигналів до лабораторної роботи № 11

11.7 Контрольні питання та завдання

1. Які типи вхідних сигналів ви знаєте?
2. Дайте стислу характеристику блоків бібліотеки Sources. Які з них можуть формувати векторні сигнали?
3. Як утворити багатосступеневий неперіодичний сигнал?
4. Яким параметром змінюється кут нахилу лінійної характеристики, створюваної блоком *Ramp*?
5. Як сформувати одним блоком трифазну гармонічну напругу
6. Поясніть принцип роботи блоку *Repeating Sequence*.
7. Поясніть принцип роботи блоку *Chirp Signal*.

Таблиця 11.1

№ вар	U_m	T	T_1	№ графіка	$U = f(t),\ t \in [0, 10]$
1	100	1	0,2	а, б, д	$U = 150\sin(60t + \pi/10)$
2	100	1	0,4	а, в, є	$U = 125\cos(50t - \pi/12)$
3	100	1	0,6	а, г, і	$U = 100\sin(10t)e^{-t/4}$
4	150	2	0,5	а, д, з	$U = 100e^{-t/4}$
5	150	2	0,8	ж, б, д	$U = 200\sqrt{t}$
6	150	2	0,75	ж, б, є	$U = 100\sqrt[3]{t}$
7	200	0,5	0,1	ж, в, і	$U = \dfrac{100}{t + 0.1}$
8	200	0,5	0,2	ж, г, д	$U = \dfrac{150t}{t + 1}$
9	200	0,5	0,15	ж, г, є	$U = 10t^2 + 50t + 4$
10	220	0,1	0,01	ж, г, і	$U = 100\sin(\lg(t + 1))$
11	220	0,1	0,02	а, б, є	$U = 200\cos(\ln(t + 1))$
12	220	0,1	0,04	а, б, і	$U = 150\dfrac{1}{1 - 0.5\sin t}$
13	250	10	1	а, г, д	$U = 100\dfrac{t^2}{1 + t^3}$
14	250	10	2	а, г, є	$U = 200\cos^2(t)\cos(2t)$
15	250	10	4	а, з, і	$U = 200\sin^2(t)\sin(2t)$
16	75	5	1	а, з, є	$U = e^{\sin 2t}\sin t$
17	75	5	0,5	а, д, і	$U = 200\sqrt[3]{t}\cdot\sin(3t)$
18	75	5	1,5	4, є, ж	$U = 100\sqrt[2]{t}\cdot\cos(2t)$
19	110	2	0,1	а, г, і	$U = 5\cos(2t)\cdot e^{-t/2}$
20	110	2	0,15	б, д, ж	$U = 3\sin(4t)\cdot e^{-t/4}$

Лабораторна робота №12

МОДЕЛЮВАННЯ НЕПЕРЕРВНИХ ЛІНІЙНИХ ДИНАМІЧНИХ СИСТЕМ У СЕРЕДОВИЩІ *Simulink*

Мета роботи: навчитися конструювати структурні математичні моделі динамічних систем за їх математичним описом.

12.1 Форми математичного опису неперервних лінійних систем

У більшості випадків на першому етапі досліджень роблять такі припущення, які дозволяють вважати САУ лінійними та стаціонарними (*LTI-systems*). Методи аналізу та синтезу таких системи є найпростішими та добре розвинутими. Неперервні *LTI*-системи описуються звичайними лінійними диференційними рівняннями з постійними коефіцієнтами.

За кількістю вхідних та вихідних величин системи розподіляються на такі групи:

- *SISO (Single Input Single Output)* – системи, що мають один вхід та один вихід;
- *SIMO (Single Input Many Output)* – системи, що мають один вхід та багато виходів;
- *MIMO (Many Input Many Output)* – системи, що мають багато входів та багато виходів.

В середовищі *Simulink* застосовують три форми подання звичайних ДР, що описують неперервні *LTI*-системи:

- ***Transfer Function Polynomial*** – передавальні функції у поліноміальній формі;
- ***Zero-Pole-Gain*** – передавальні функції, розкладені за нулями та полюсами;
- ***State Space*** – математичний опис у просторі станів.

Перші 2 форми в теорії автоматичного керування (ТАК) відносяться до класичних форму запису, які застосовують для опису *SISO*-систем.

Третя форма використовується у сучасній теорії керування і є найбільш універсальною, тому що може описати як *SISO*-, так і *SIMO*- та *MIMO*-системи.

Система з одним входом $u(t)$ і одним виходом $y(t)$ в загальному випадку описується ***звичайним диференційним рівнянням*** (ДР) n-го порядку

$$A_n\frac{d^n y(t)}{dt^n}+A_{n-1}\frac{d^{n-1}y(t)}{dt^{n-1}}+...+A_1\frac{dy(t)}{dt}+A_0 y(t)=$$

$$=B_m\frac{d^m u(t)}{dt^m}+B_{m-1}\frac{d^{m-1}u(t)}{dt^{m-1}}+...+B_1\frac{du(t)}{dt}+B_0 u(t) \qquad (12.1)$$

з початковими умовами

$$y(0)=y_0,\ y'(0)=y'_0,\ ...,\ y^{(n-1)}(0)=y_0^{(n-1)}\ . \qquad (12.2)$$

Для того, щоб отримати з ДР (12.1) передавальну функцію (ПФ) запишемо ***ДР в операторній формі*** шляхом заміни оператору диференціювання *d/dt* оператором Лапласа *s*, а сигналів у функції часу *x(t)* їх зображеннями *x(s)*:

$$\frac{d}{dt}\to s\,,\qquad u(t)\to u(s),\qquad y(t)\to y(s)\,. \qquad (12.3)$$

У такий спосіб отримаємо

$$A_n s^n y(s)+A_{n-1}s^{n-1}y(s)+...+A_1 s y(s)+A_0 y(s)=$$

$$=B_n s^m u(s)+B_{n-1}s^{m-1}u(s)+...+B_1 s u(s)+B_0 u(s)\,. \qquad (12.4)$$

Оскільки тепер операцію диференціювання замінено операцією множення на оператор Лапласа, тобто диференційне рівняння перетворилося на алгебраїчне, зображення вихідного і вхідного сигналів можна винести за дужки:

$$(A_n s^n+A_{n-1}s^{n-1}+...+A_1 s+A_0)y(s)=(B_n s^m+B_{n-1}s^{m-1}+...+B_1 s+B_0)u(s)\,. \qquad (12.5)$$

З останнього рівняння знаходимо ***передавальну функцію (Transfer Function)*** досліджуваної системи як відношення зображень вихідного сигналу до вхідного при нульових початкових умовах:

$$W(s)=\frac{y(s)}{u(s)}=\frac{B_n s^m+B_{n-1}s^{m-1}+...+B_1 s+B_0}{A_n s^n+A_{n-1}s^{n-1}+...+A_1 s+A_0}\,. \qquad (12.6)$$

Як бачимо, і чисельних, і знаменних передавальної функції (12.6) є степеневими поліномами. Тому математичний опис лінійної динамічної системи у формі (12.6) називають ***Transfer Function Polynomial.*** У багатьох

випадках використання такої форми математичного опису цілком достатньо для структурного математичного моделювання неперервних систем у середовищі *Simulink*.

Порядки поліномів у чисельнику та у знаменнику передавальної функції пов‘язані між собою умовою можливості фізичної реалізації:

$$m \le n . \tag{12.7}$$

Знаменник передавальної функції звуть характеристичним поліномом (ХП) системи

$$G(s) = A_n s^n + A_{n-1} s^{n-1} + ... + A_1 s + A_0 . \tag{12.8}$$

Порядок динамічної ланки визначається порядком характеристичного поліному, що збігається з порядком відповідного диференційного рівняння.

Ланки з різними передавальними функціями мають спеціальні назви, наприклад,

- пропорційна $W_{\text{п}}(s) = k$;
- інтегральна $W_{\text{і}}(s) = \frac{1}{s}$;
- аперіодична $W_{\text{а}}(s) = \frac{k}{Ts+1}$;
- коливальна $W_{\text{к}}(s) = \frac{k}{T^2 s^2 + 2\xi Ts + 1}$.

12.2 Моделювання лінійних динамічних систем в середовищі *Simulink*

Simulink є середовищем для структурного математичного моделювання. Це означає, що математичний опис досліджуваних об‘єктів має бути поданим у вигляді структурних схем, в яких динамічні властивості систем відображаються за допомогою передавальних функцій, отриманих із диференційних рівнянь.

Структурні схеми уявляють собою графічне відображення математичного опису у вигляді з‘єднаних між собою односпрямованими лініями зв‘язку окремих блоків. Структурні схеми лінійних неперервних динамічних систем складаються з передавальних функцій, пропорційних

(масштабних) ланок та алгебраїчних суматорів.

В *Simulink* неперервні динамічні блоки знаходяться в бібліотеці *Continuous*. До них належать блоки аналогового інтегрування () і диференціювання (*Derivative*), лінійні неперервні динамічні ланки загального вигляду в різній формі (*Transfer Fcn*, *Zero-Pole* і *State-Space*) і ланки запізнювання (*Memory*, *Transport Delay* і *Variable Transport Delay*). Із них найчастіше споживаними є блоки *Integrator* та *Transfer Fcn*.

Блок ***Transfer Fcn*** реалізує передавальну функцію загального вигляду в поліноміальній формі (12.6). Його вхідними параметрами є вектори коефіцієнтів степеневих поліномів чисельника (*Numerator*) і знаменника (*Denominator*), упорядковані за зменшенням степенів оператора Лапласа *s*. Вектор з (*n*+1) елементів задає поліном степені *n*. Порядок знаменника повинний бути більшим від порядку чисельника або дорівнювати йому. Блок має нульові початкові умови.

Коефіцієнти можуть бути введені трьома способами:

1) константами і скалярними змінними, об'єднаними квадратними дужками у вектор;

2) змінною без дужок;

3) змінною або виразом у круглих дужках.

При першому способі в піктограмі блоку відображаються поліноми з коефіцієнтами у вигляді введених констант і змінних при відповідних степенях оператора Лапласа *s*. Наприклад, введення вектору [1 -2 *T* 4] відобразиться в блоці як

$$s^3 - 2s^2 + Ts + 4 \ ,$$

а уведення вектору [*b*0, *b*1, *b*2] – як

$$b0s^2 + b1s + b2 \, .$$

Змінні, що використані в цих векторах, повинні одержати значення в робочому середовищі *MATLAB* до початку процесу моделювання.

При другому способі поліном відображається у вигляді *змінна(s)*.

Наприклад, введення *num* відображається як *num(s)*, введення *An* – як *An(s)* та т.і.

При третьому способі *Simulink* шукає значення змінних, які входять до виразу, у робочому просторі *MATLAB,* обчислює значення виразу і відображає його так само, як і при першому способі введення. Наприклад, якщо поліном визначено у діалоговому вікні як (*G*+*B*), а перемінні *G* і *B* визначено в *MATLAB* як *G*=[4 1 1], *B*=[0 1 0], то відображення полінома в блоці має вид:

$$4s^2+2s+1.$$

Значення змінних повинне існувати в робочому середовищі в будь-який момент зображення блоку, інакше в піктограмі блоку відобразяться три знаки питання: *???*.

Блок ***Integrator (Інтегратор)*** у найпростішому випадку інтегрує вхідний скалярний u або векторний $\mathbf{u}=[u_1, u_2, ..., u_k]$ сигнал із заданими початковими умовами (*Initial condition*), тобто

$$\mathbf{x}(t)=\mathbf{x}_0+\int_0^t \mathbf{u}(t)dt\,. \qquad (12.9)$$

Він може також працювати у режимах обмеження вихідного сигналу на заданому рівні (*Limited Integrator*) та скидання його у заданий початковий стан (*Reset Integrator*).

Вікно введення параметрів блоку зображено на рис. 12.1.

Для того, щоб обмежити вихідний сигнал аналогового інтегратора, необхідно установити прапорець *Limit output* і ввести значення параметрів y_{max} (*Upper saturation limit*) і y_{min} (*Lower saturation limit*). Припустимим значенням цих параметрів є константа *Inf* (∞), що дозволяє обмежувати блок тільки знизу або тільки зверху.

Початкова умова може задаватися константою в параметрах блоку і може визначатися зовнішнім сигналом. Вибір здійснюється за допомогою меню опції *Initial condition source*. При виборі режиму *internal* діє початкова умова, що встановлюється в полі параметра *Initial condition*. При виборі режиму *external*

блок одержує додатковий вхідний порт, до якого можна приєднати блок *IC* (*Початкова умова*) бібліотеки *Signal Attributes*. Задавати зовнішні початкові умови має сенс тільки для інтегратора зі скиданням.

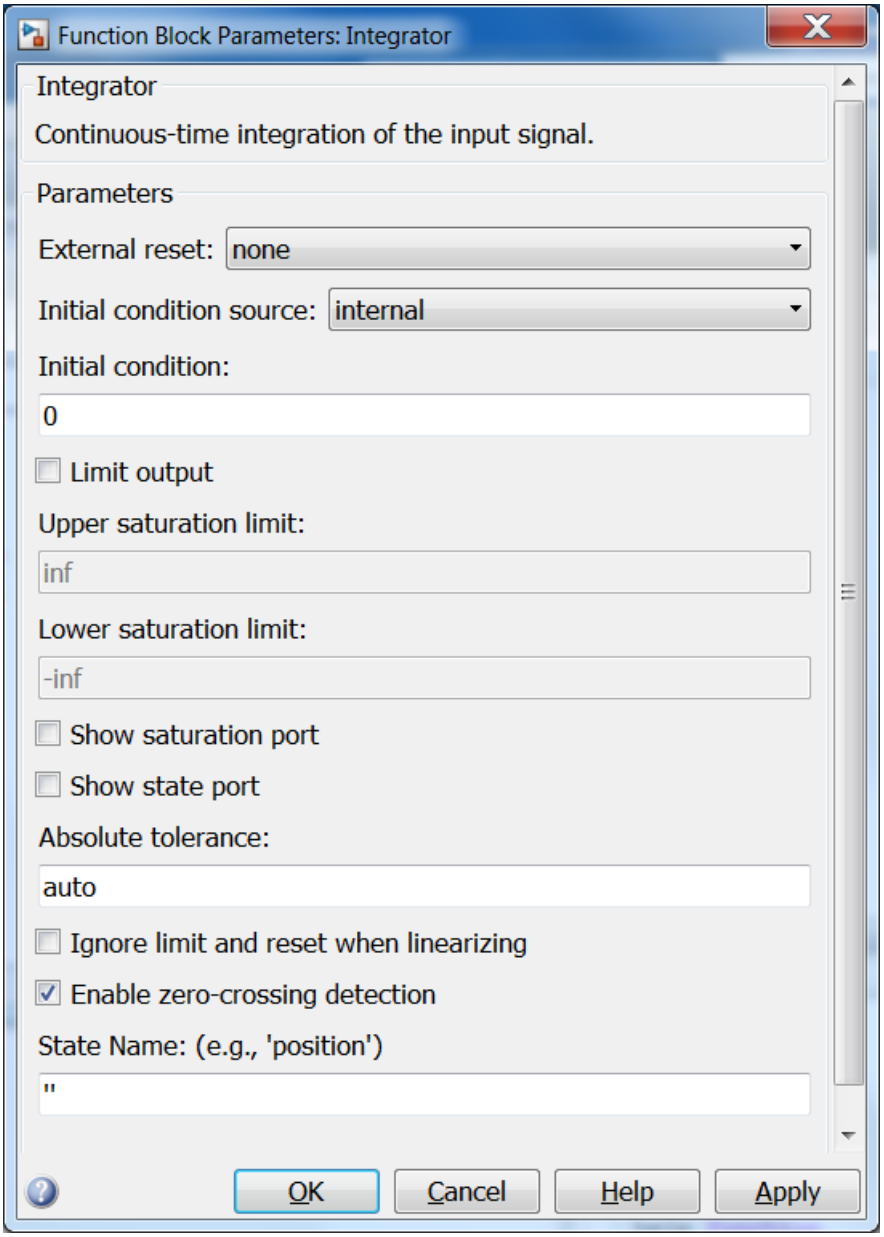


Рис. 12.1. Вікно введення параметрів блоку *Integrator*

Для організації скидання або зміни початкових умов у функції вихідного сигналу інтегратора необхідно, щоб уникнути утворення алгебраїчної петлі, замість вихідного порту використовувати порт стану, для чого потрібно візуалізувати його установкою прапорця *Show state port*. Сигнал стану

інтегратора формується раніш, ніж вихідний сигнал, хоча і має з ним однакові значення.

На рис. 12.2 зображені піктограми блоку *Integrator* з різними установками режимів.

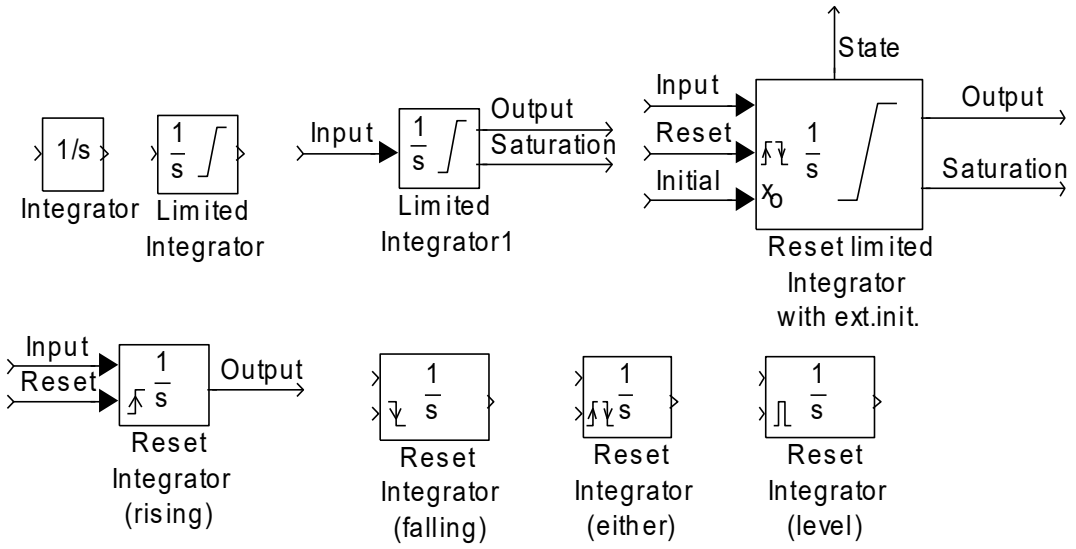


Рис. 12.2. Піктограми інтеграторів, які працюють у різних режимах

Пропорційна ланка реалізується блоком ***Gain* (*Підсилення*, *Коефіцієнт передачі*)** бібліотеки *Math Operations*.

Піктограма блоку відображає скалярний коефіцієнт підсилення в тій же формі, у якій він визначений при введенні (змінна або константа). Якщо коефіцієнт заданий у вигляді змінної в круглих дужках, то усередині блоку відображається її значення. Якщо значення змінної або її ім'я, або вираз занадто великі для того, щоб відобразити їх у межах блоку, то піктограма відображає символи "-*K*-". Щоб побачити вираз або значення коефіцієнта підсилення, необхідно збільшити розміри блоку.

Алгебраїчний суматор реалізується одним з блоків *Sum (Суматор), Add (Додавання), Subtract (Віднімання)* або *Sum of Elements (Сума елементів)*, як це описано в попередньому розділі (див. рис. 11.4 та 11.11).

Структурна модель, складена із описаних вище блоків, доповнюється джерелами вхідних сигналів та блоками візуалізації або реєстрації вихідних сигналів.

12.3 Завдання

1. Отримати реакцію на стрибкоподібний сигнал лінійної динамічної системи з передавальною функцією в поліноміальній формі (12.6). Коефіцієнти поліномів у чисельнику та у знаменнику передавальної функції наведені у табл. 12.1.
2. Виконати демонстрацію Bouncing ball. Розібратися з роботою інтеграторів у різних режимах.

3. Таблиця 12.1

№ вар.	A_5	A_4	A_3	A_2	A_1	A_0	B_2	B_1	B_0
1	1	5	10	10	5	1	0	2	1
2	1	5	10	10	5	1	3	3	1
3	1	5	10	10	5	1	0	4	0
4	1	3.2	5.2	5.2	3.2	1	0	0	1
5	1	3.2	5.2	5.2	3.2	1	0	2	0
6	1	3.2	5.2	5.2	3.2	1	0	1	1
7	1	2.8	5	5.5	3	1	0	0	1
8	1	2.8	5	5.5	3	1	0	2	0
9	1	2.8	5	5.5	3	1	1.4	1.8	1
10	1	3.8	6.8	6.9	3.9	1	0	3	0
11	1	3.8	6.8	6.9	3.9	1	0	2	1
12	1	3.8	6.8	6.9	3.9	1	1.3	2.1	1

12.4 Методичні рекомендації

1. Стрибкоподібний сигнал з одиничною амплітудою сформуйте блоком *Step*.
2. Час моделювання підбирайте таким, щоб вихідна координата встигла досягти свого усталеного значення $y_{\text{уст}} = B_0 / A_0$.
3. Перше наближення часу перехідного процесу оберіть рівним 20.

12.5 Контрольні питання та завдпння

4. Який математичний опис мають лінійні неперервні системи?
5. Що таке передавальна функція? Як її отримати з диференціального рівняння?
6. Які *Simulink*-блоки використовують при моделюванні лінійних неперервних систем?
7. Які параметри має блок *Transfer Fn*? Як вони задаються та відображуються в піктограмі блоку?
8. Яку операцію виконує блок *Integrator*? Які параметри він має? У яких режимах може працювати?
9. Що таке пропорційна ланка? Яким блоком вона реалізується в середовищі *Simulink*?
10. Яким блоками в можна реалізувати алгебраїчні суматори? Як можна змінювати їх зовнішній вигляд?

Лабораторна робота №13

СТРУКТУРНЕ МОДЕЛЮВАННЯ ЕЛЕКТРИЧНИХ КІЛ

13.1 Методика структурного математичного моделювання

Для структурного моделювання на основі математичного опису треба утворити ***структурну схему*** досліджуваного об‘єкту.

Неперервні стаціонарні динамічні системи описуються сукупністю диференційних та алгебраїчних лінійних та нелінійних рівнянь. Одним із універсальних способів розробки структурних схем таких об‘єктів полягає у попередньому перетворенню диференційних рівнянь довільного виду і порядку у систему диференційних рівнянь першого порядку у нормальній формі Коші, у яких перші похідні від змінних стану виражені через вхідні, вихідні сигнали та змінні стану:

$$\begin{cases} \dfrac{dx_1(t)}{dt} = f_1(x_1, x_2, ..., x_n, u_1, u_2, ..., u_k, y_1, y_2, ..., y_r), \\ \dfrac{dx_2(t)}{dt} = f_2(x_1, x_2, ..., x_n, u_1, u_2, ..., u_k, y_1, y_2, ..., y_r), \\ \qquad\qquad ..., \\ \dfrac{dx_n(t)}{dt} = f_n(x_1, x_2, ..., x_n, u_1, u_2, ..., u_k, y_1, y_2, ..., y_r), \end{cases} \tag{13.1}$$

де

$$\begin{cases} y_1 = g_1(x_1, x_2, ..., x_n, u_1, u_2, ..., u_k), \\ y_2 = g_1(x_1, x_2, ..., x_n, u_1, u_2, ..., u_k), \\ \qquad\qquad ..., \\ y_3 = g_1(x_1, x_2, ..., x_n, u_1, u_2, ..., u_k), \end{cases} \tag{13.2}$$

$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & ... & x_n \end{bmatrix}^T$ – вектор змінних стану;

$\mathbf{y} = \begin{bmatrix} y_1 & y_2 & ... & y_r \end{bmatrix}^T$ – вектор вихідних сигналів;

$\mathbf{u} = \begin{bmatrix} u_1 & u_2 & ... & u_k \end{bmatrix}^T$ – вектор вхідних сигналів;

з початковими умовами

$$\mathbf{x}(t_0) = \mathbf{x}_0 = \begin{bmatrix} x_{10} & x_{20} & ... & x_{n0} \end{bmatrix}^T. \tag{13.3}$$

Рівняння (13.2) називають рівняннями виходу або рівнянням зв‘язку. На відміну від диференційних рівнянь (13.1) рівняння (13.2) є алгебраїчними.

Для складання структурної схеми диференційні рівняння (11.1) треба записати в ***операторній формі***. Для цього оператор диференціювання $\frac{d}{dt}$ замінюють оператором Лапласа s, і переходять від сигналів у просторі часу $x_i(t), u_i(t), y_i(t)$ до зображень цих сигналів у просторі змінної Лапласа $x_i(s), u_i(s), y_i(s)$:

$$\begin{cases} sx_1(s) = f_1(x_1, x_2, ..., x_n, u_1, u_2, ..., u_k, y_1, y_2, ..., y_r), \\ sx_2(s) = f_2(x_1, x_2, ..., x_n, u_1, u_2, ..., u_k, y_1, y_2, ..., y_r), \\ \qquad\qquad\qquad ..., \\ sx_n(s) = f_n(x_1, x_2, ..., x_n, u_1, u_2, ..., u_k, y_1, y_2, ..., y_r). \end{cases} \tag{13.4}$$

У такий спосіб операція диференціювання у просторі часу замінюється операцією множення зображення сигналу на змінну Лапласа. Відповідно операції інтегрування відповідає операція ділення зображення сигналу на оператор Лапласа.

Структурну схему за математичним описом складають ***методом неявних сигналів***, тобто перші похідні від змінних стану формують за рівняннями (13.1), не звертаючи уваги на те, що деякі з них є невідомими. Отримані похідні пропускають через інтегратори

$$W_I(s) = \frac{1}{s}, \tag{13.5}$$

отримуючи на їхніх виходах змінні стану. На останнє з відомих вхідних сигналів та змінних стану за рівняннями (13.2) формують вихідні сигнали.

Надалі діють у такій послідовності:

- структурну схему набирають із блоків *Simulink*, доповнюючи її блоками формування вхідних сигналів та реєстрації вихідних;
- виставляють параметри усіх блоків (краще задавати їх у вигляді змінних, а не констант);

- створюють, записують та виконують файл даних для ініціалізації моделі;
- розробляють план модельного експерименту і створюють програмні файли для його реалізації;
- налаштовують модель, обираючи метод розв‘язання диференційних рівнянь та його параметри;
- виконують власне симуляцію, фіксують, візуалізують та аналізують отримані результати.

13.2 Приклад структурного моделювання лінійного електричного кола

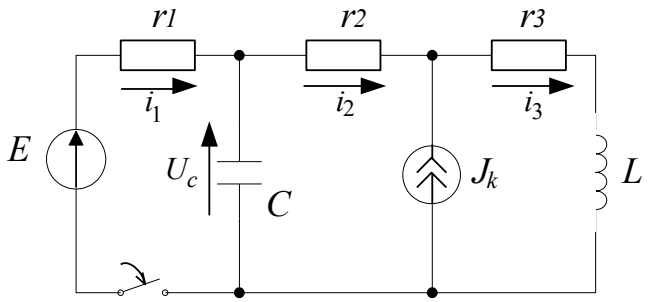


Рис. 13.1. Досліджуване електричне коло

Як приклад, розглянемо розгалужену схему, зображену на рис. 13.1, для якої треба розрахувати перехідні процеси, тобто знайти $i_1(t)$, $i_2(t)$, $i_3(t)$, $U_C(t)$ при замиканні ключа при таких параметрах: E = 100 В, J_k = 5 А, C = 200 мкФ, L = 0,3 Гн, r_1 = 20 Ом, r_2 = 50 Ом, r_3 = 70 Ом.

Математичний опис лінійних електричних кіл створюють, використовуючи закони Ома та Кірхгофа з урахуванням тієї обставини, що напруги і струми на резисторі індуктивності та конденсаторі пов‘язані між собою співвідношеннями:

$$U_R(t) = RI_R(t), \qquad U_L(t) = L\frac{dI_L(t)}{dt}, \qquad I_c(t) = c\frac{dU_c(t)}{dt}. \tag{13.6}$$

Математичний опис схеми при замкненому стані ключа, згідно з законами Кірхгофа, має вигляд:

$$E = i_1 r_1 - U_c, \tag{13.7}$$

$$E = i_1 r_1 + i_2 r_2 + i_3 r_3 + L\frac{di_3}{dt}, \tag{13.8}$$

$$i_1 + C\frac{dU_c}{dt} = i_2, \tag{13.9}$$

$$i_2 + J_k = i_3. \tag{13.10}$$

Рівняння (13.8), (13.9) – диференційні; (13.7), (13.10) – алгебраїчні рівняння, які часто називають рівняннями зв'язку.

Перетворимо диференційні рівняння у нормальну форму Коші:

$$\begin{cases} \dfrac{dU_C}{dt} = (i_2 - i_1)/C, \\ \dfrac{di_3}{dt} = (E - i_1 r_1 - i_2 r_2 - i_3 r_3)/L. \end{cases} \tag{13.11}$$

Об'єднаємо змінні стану, вхідні та вихідні сигнали в оу відповідні вектори:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} U_c \\ i_3 \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} E \\ J_k \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} U_c \\ i_3 \\ i_1 \\ i_2 \end{bmatrix} \tag{13.12}$$

Як бачимо, два перших вихідних сигналів співпадають зі змінними стану, а два останніх можна виразити через змінні стану з рівнянь зв'язку:

$$\begin{cases} i_1 = (E + U_C)/r_1, \\ i_2 = i_3 - J_k. \end{cases} \tag{13.13}$$

Записуємо ДР (13.11) в операторній формі:

$$\begin{cases} sU_C(s) = (i_2(s) - i_1(s))/C, \\ si_3(s) = (E(s) - i_1(s)r_1 - i_2(s)r_2 - i_3(s)r_3)/L. \end{cases} \tag{13.14}$$

Рівняння виходу (13.13) у зображеннях залишаються такими ж, як і у просторі часу.

Розрахуємо початкові умови для заданої схеми, які збігаються з усталеними значеннями відповідних сигналів при розімкненому стані ключа. Така схема зображена на рис. 13.2.

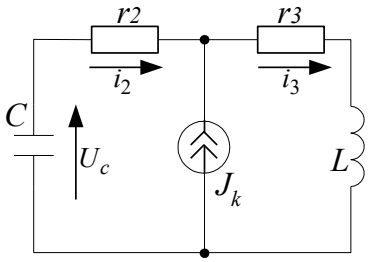


Рис. 13.2. Схема для визначення початкових умов

Для складання структурної моделі лінійного неперервного динамічного об‘єкта завжди можна обійтися тільки трьома типами блоків: інтеграторами, алгебраїчними суматорами та блоками множення на постійний коефіцієнт, які у *Simulink* називаються відповідно: *Integrator*, *Sum* та *Gain*. Ще декілька блоків знадобляться для запам‘ятовування або для візуалізації результатів моделювання.

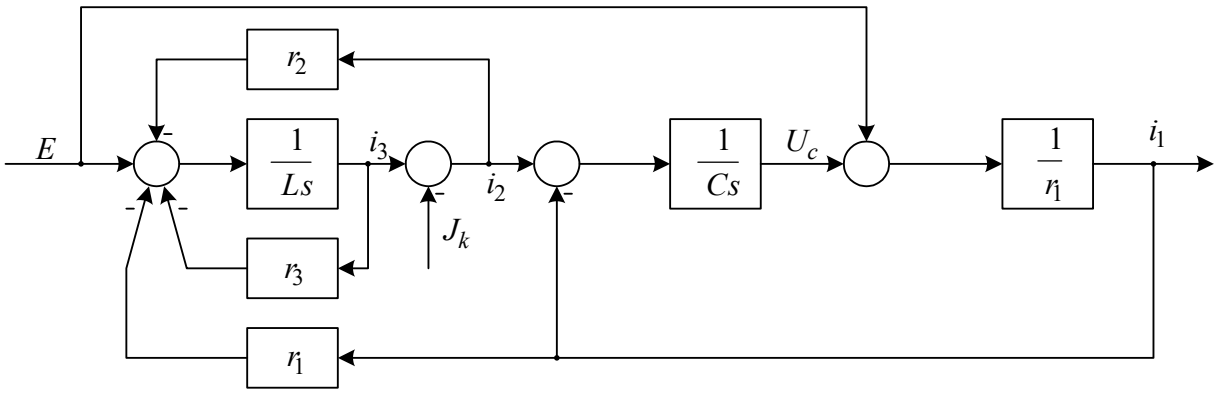


Рис. 13.3. Структурна схема досліджуваного електричного кола

Відповідна схема в блоках *Simulink* показана на рис. 13.4.

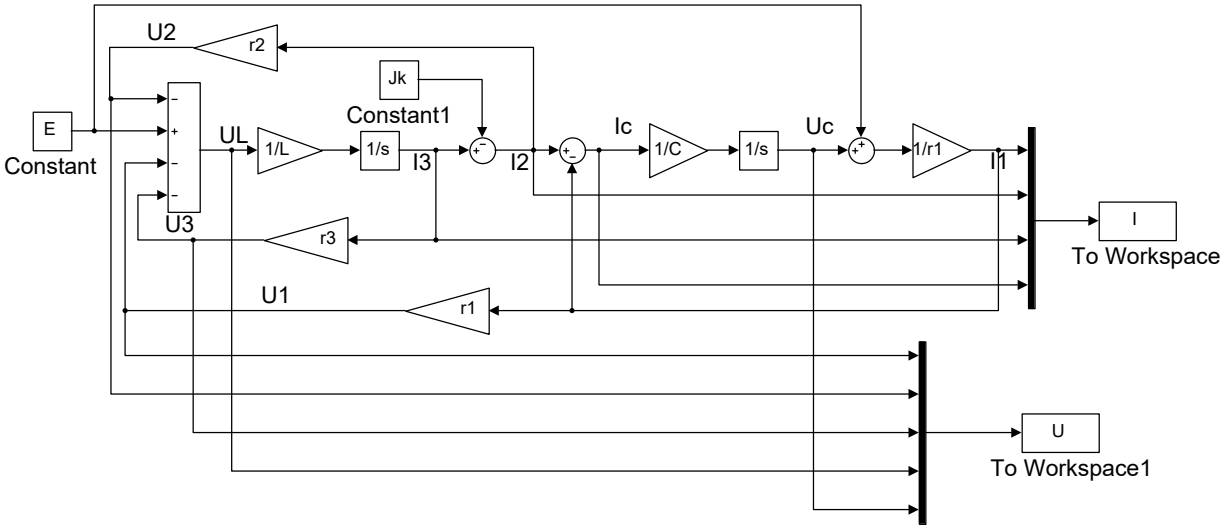


Рис. 13.4. Структурна *Simulink*-модель досліджуваного лінійного електричного кола

Результат роботи програми наведено на рис. 13.5.

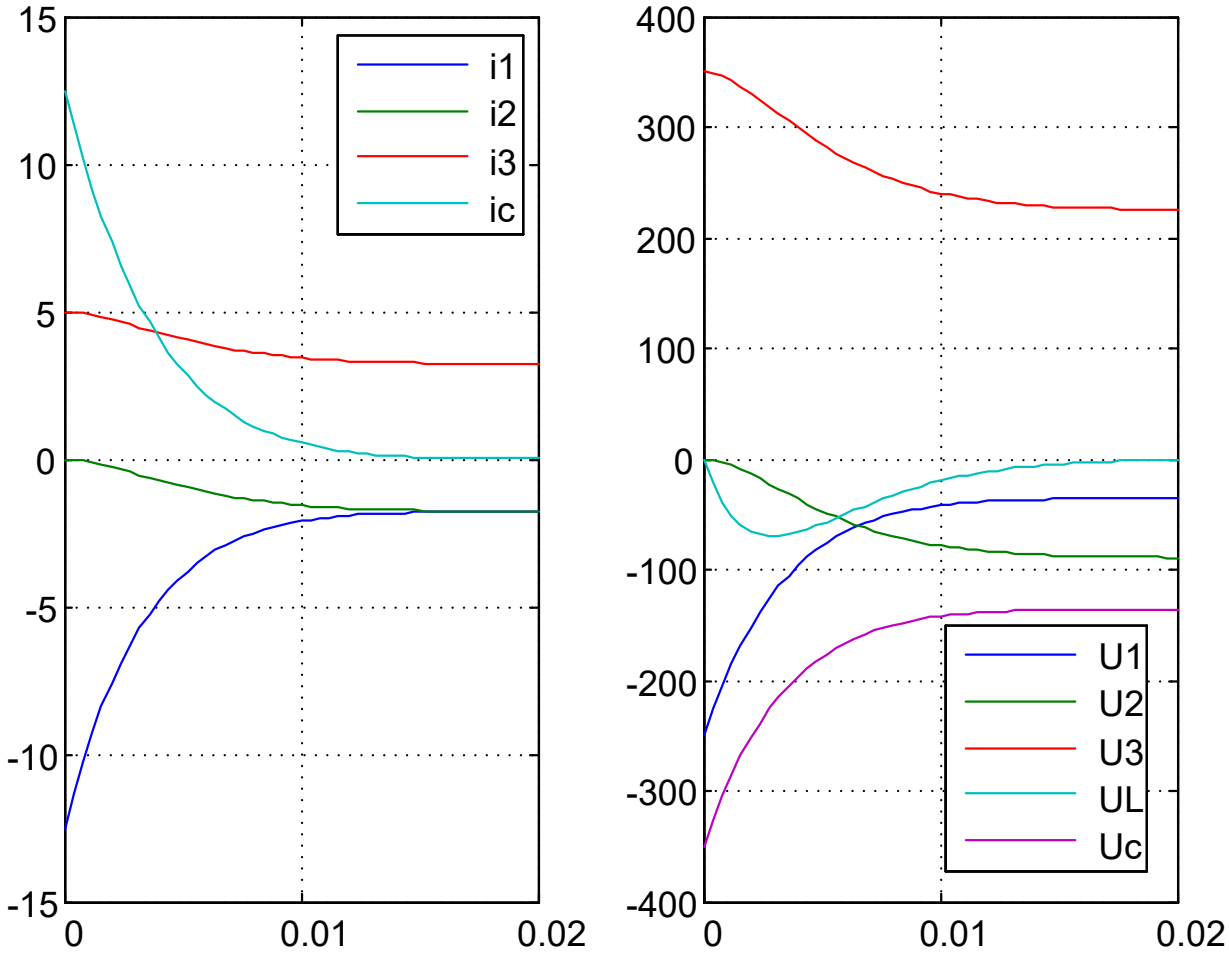


Рис. 13.5. Перехідні процеси у досліджуваному лінійному електричному колі

Якщо ЕРС та/або струми джерел змінюються у часі, то замість вхідних блоків *Const*, що генерують постійні сигнал E та Jk слід використати інші блоки бібліотеки *Sources*, наприклад, *Ramp, Sine Wive*, *Pulse Generator*, *Repeating Sequence* тощо.

13.3 Завдання

Розрахувати перехідні процеси напруги на конденсаторі та струмів у гілках однієї з електричних схем, зображених на рис. 13.6, методом структурного математичного моделювання для трьох випадків:

- $U=U_m$– джерело постійної напруги;
- $U=U_m\sin\omega t$ – джерело синусоїдальної напруги;

• періодична напруга (вигляд сигналу показано на рис. 13.7), T=0.1 с, $T1$=0.07 с.

За результатами розрахунків побудувати графіки перехідних процесів. Параметри схем надані у табл. 13.1.

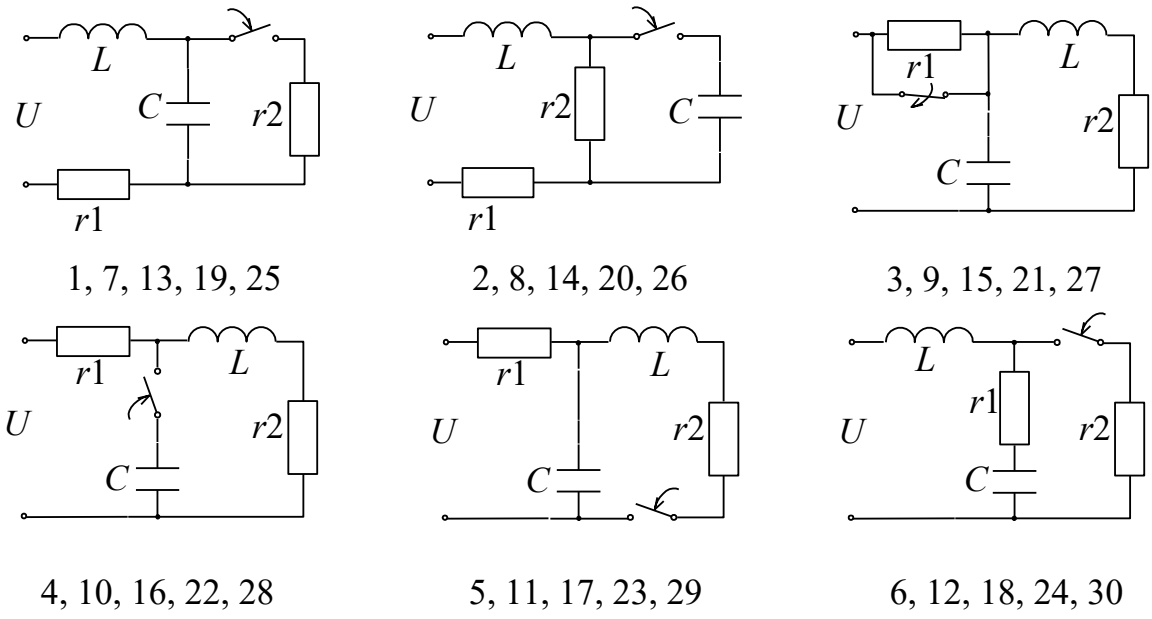


Рис. 13.6. Варіанти розгалужених лінійних електричних кіл

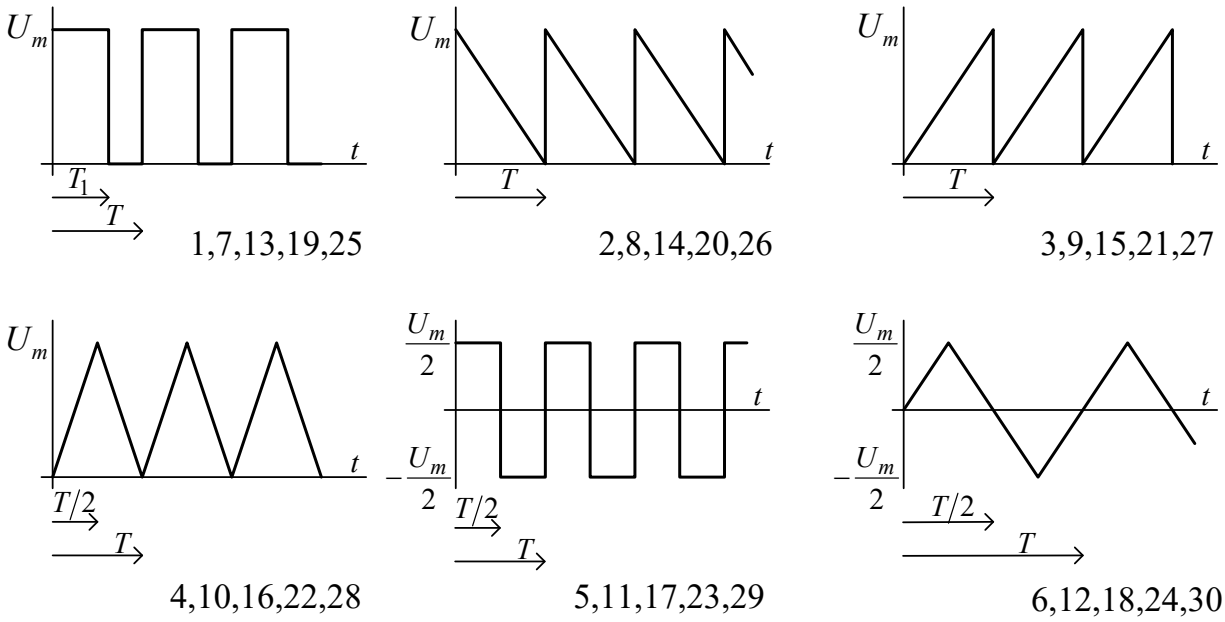


Рис. 13.7. Варіанти вхідних періодичних сигналів

Таблиця 13.1

№ вар.	U_m, В	$r1$, Ом	$r2$, Ом	ω, $с^{-1}$	L, Гн	C, мкФ
1-6	200	50	100	125	0,1	10
7-12	100	100	120	100	0,2	5
13-18	300	60	120	75	0,15	15
19-24	400	150	100	150	0,25	2
25-30	500	100	200	100	0,05	10

13.4 Методичні вказівки та рекомендації

1 Параметри моделей задавайте іменами змінних, а не константами.

2 Для попередньої візуалізації використовуйте блоки Scope, а для запам‘ятовування чисельних результатів з метою подальшої побудови графіків перехідних процесів – вихідні порти Out та блоки To Workspace. Усі струми виводьте на один графік, а напруги – на інший.

3 Ініціалізацію моделей та побудову графіків перехідних процесів виконайте шляхом занесення відповідної програмної інформації до вкладки *Callbacks* функції *Model Properties* меню *File*.

4 Час розрахунку перехідного процесу обирайте таким, щоб усталений режим був досягнутий, але не тривав дуже довго.

13.5 Контрольні питання та завдання

1. Які закони теоретичної електротехніки використовують для математичного опису електричних кіл?
2. Запишіть рівняння напруги на індуктивному елементі та струму, що протікає через конденсатор.
3. Які способи апроксимації нелінійних табличних залежностей Ви знаєте? Як їх реалізувати блоками *Simulink*?
4. Як записати диференційні рівняння у нормальній формі Коші?
5. Як створити структурну схему електричного кола з його математичного опису?

Лабораторна робота №14

СТРУКТУРНЕ МОДЕЛЮВАННЯ ДВИГУНА ПОСТІЙНОГО СТРУМУ З КЕРУВАННЯМ У КОЛІ ЯКОРЯ

Мета роботи: навчитися досліджувати статичні та динамічні властивості реальних об‘єктів методом математичного моделювання.

14.1 Математичний опис об‘єкту моделювання

Двигун постійного струму (ДПС) з незалежним збудженням широко застосовується у регульованих електроприводах, що працюють у напружених повторно-короткочасних режимах і потребують забезпечення високої якості перехідних процесів.

При складанні його математичного опису знехтуємо розмагнічувальною дією реакції якоря, падінням напруги на щітках та тертям, а індуктивність і активний опір якірного кола та сумарний момент інерції вважатимемо постійними величинами. Регулювання швидкості здійснюватимемо зміною напруги якоря при постійній напрузі збудження. При прийнятих припущеннях ДПС описується лінійними диференційними та алгебраїчними рівняннями [13]:

$$U_{\text{я}}(t) - E_{\text{д}}(t) = \Delta U_{\text{я}}(t) = I_{\text{я}}(t)R_{\text{я}} + L_{\text{я}}\frac{dI_{\text{я}}(t)}{dt}, \tag{14.1}$$

$$M(t) = cI_{\text{я}}(t), \tag{14.2}$$

$$M(t) - M_{\text{с}}(t) = M_j(t) = J\frac{d\omega(t)}{dt}, \tag{14.3}$$

$$E_{\text{д}}(t) = c\omega(t), \tag{14.4}$$

де

$U_{\text{я}}$, $I_{\text{я}}$ – напруга та струм якоря;

$\Delta U_{\text{я}}$ – падіння напруги у якірному колі;

$R_{\text{я}}$, $L_{\text{я}}$ – активний опір та індуктивність якоря;

$E_{\text{д}}$, ω – електрорушійна сила (ЕРС) та кутова швидкість двигуна;

$M, M_{\text{с}}, M_j,$ – електромагнітний момент двигуна, момент статичного опору та динамічний момент;

J – момент інерції;

$$c = k\Phi_{\text{зн}}; \tag{14.5}$$

$\Phi_{\text{зн}}$ – номінальний потік збудження двигуна;

$k = \dfrac{pN}{2\pi a}$ – конструктивний коефіцієнт двигуна;

p, a, N – кількість пар полюсів, паралельних гілок та активних проводників відповідно.

Рівняння (14.1) називають ***рівнянням електромагнітної рівноваги якірного кола***, що складається за другим законом Кірхгофа, рівняння (14.3) – ***рівнянням руху*** двигуна, що складається за другим законом Кірхгофа для тіл обертального руху. Рівняння моменту (14.2) та ЕРС (14.4) свідчать про те, що при постійному потоку збудження електромагнітний момент двигуна є пропорційним струму якоря, а його ЕРС є пропорційною кутовій швидкості двигуна.

Для складання структурної схеми в цьому разі застосуємо дещо інший підхід, ніж у попередній лабораторній роботі, а саме, запишемо ДР в операторному вигляді без попереднього перетворення їх у нормальну форму Коші:

$$U_{\text{я}}(s) - E_{\text{д}}(s) = \Delta U_{\text{я}}(s) = I_{\text{я}}(s)R_{\text{я}} + L_{\text{я}}sI_{\text{я}}(s), \tag{14.6}$$

$$M(s) - M_{\text{с}}(s) = M_j(s) = Js\omega(s). \tag{14.7}$$

З отриманих рівнянь визначаємо передавальні функції якірного кола і механічної частини двигуна як відношення зображень вихідного сигналу до вхідного:

$$W_{\text{я}}(s) = \frac{I_{\text{я}}(s)}{U_{\text{я}}(s) - E_{\text{д}}(s)} = \frac{I_{\text{я}}(s)}{\Delta U_{\text{я}}(s)} = \frac{1/R_{\text{я}}}{T_{\text{я}}s + 1}, \tag{14.8}$$

де $T_{\text{я}} = L_{\text{я}} / R_{\text{я}}$ (14.9)

– ***електромагнітна стала часу якірного кола***.

$$W_{\text{м}}(s) = \frac{\omega(s)}{M(s) - M_{\text{с}}(s)} = \frac{\omega(s)}{M_j(s)} = \frac{1}{Js}. \tag{14.10}$$

Структурна схема ДПС, складена за рівняннями (14.8), (14.2), (14.10) та (14.4) і (14.2), зображена на рис. 14.1.

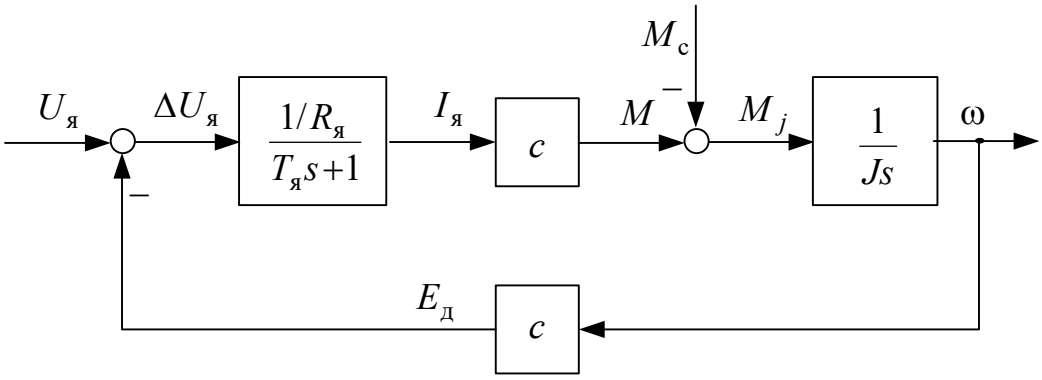


Рис. 14.1. Структурна схема двигуна постійного струму з нерегульованим незалежним збудженням

14.2 Завдання

Промоделюйте двигун постійного струму з незалежним збудженням при постійному потоку збудження з даними, які наведені в табл. 14.1, в таких режимах:

- при стрибкоподібній зміні напруги двигуна від 0 до номінального значення;
- при лінійній зміні напруги двигуна від 0 до номінального значення за час $t_{р0}$, що обирається з умов забезпечення бажаної величини динамічного моменту;
- при стрибкоподібному прикладанні активного номінального моменту статичного опору.
- при лінійній зміні напруги двигуна до номінального значення, стрибкоподібному накиданні і скиданні номінального навантаження на усталеній швидкості та лінійній зміні напруги якоря від номінального значення до нуля без навантаження.
  - У таблиці позначено:
  - $P_н$, $Ф_н$, $n_н$ – номінальні значення потужності, потоку збудження та кутової швидкості у обертах за хвилину;
  - $U_{ян}$, $U_{зн}$ – номінальні значення напруги якоря та напруги збудження;
  - $I_{ян}$, $I_{зн}$ – номінальні значення струму якоря двигуна та струму збудження.

Таблиця 14.1

Тип	$P_н$	$n_н$	$I_{ян}$	$R_я$	N	$2a$	$2p$	$w_п$	$R_з$	$I_{зн}$	$Ф_н$	J
	кВт	*об/хв*	*А*	*Ом*	*вит*	–	–	*вит*	*Ом*	*А*	*мВб*	*кг·м²*
$U_{ян} = 440\ В,\ U_{зн} = 440\ В$												
П81	32	1500	83	0,250	580	2	4	2000	208,4	1,62	14,4	0,68
П82	25	1000	66	0,373	630	2	4	1525	137,6	2,46	19,8	0,78
П91	55	1500	143	0,143	420	2	4	1615	129,2	2,62	19,6	1,48
П101	42	750	107	0,206	564	2	4	1660	130,0	2,60	29,0	2,58
П81	19	1000	52	0,630	870	2	4	2000	208,4	1,62	14,4	0,68
П101	100	1500	256	0,051	282	2	4	1660	130,0	2,60	29,9	2,58
П102	125	1500	316	0,034	222	4	4	1450	101,5	3,33	38,3	3,00
П111	95	1000	240	0,067	660	4	4	1460	87,0	3,89	19,0	5,10
П111	160	1500	400	0,025	420	4	4	1460	87,0	3,89	20,3	5,10
П112	85	750	220	0,075	660	2	4	1310	82,0	4,13	25,3	5,57
П112	180	1500	450	0,023	736	2	4	1310	82,0	4,13	16,2	5,75
$U_{ян} = 220\ В,\ U_{зн} = 110\ В$												
П112	200	1500	1000	0,050	184	4	4	750	24,0	6,55	16,2	5,75
П52	14	3000	74	0,027	248	2	4	1200	94,8	1,79	7,2	0,10
П61	19	3000	100	0,057	248	2	4	1600	104,0	1,63	8,2	0,14
П72	25	1500	132	0,083	324	2	4	1100	67,0	2,34	12,5	0,40
П111	160	1500	809	0,064	216	4	4	850	28,0	5,61	18,6	5,10
П102	125	1500	632	0,079	228	4	4	840	32,5	4,84	13,0	3,00
П101	100	1500	508	0,013	304	2	4	950	37,8	4,16	13,7	2,58
П92	75	1500	381	0,014	162	2	4	830	31,8	4,94	20,1	1,75
П22	1	1500	5,9	4,170	1728	2	2	4800	600,0	0,30	3,2	0,01
П42	7,4	3000	39,8	0,205	378	2	4	1350	136,0	1,24	5,1	0,05
П52	14	3000	74	0,027	248	2	4	1200	94,8	1,79	7,2	0,09

14.3 Методичні вказівки і рекомендації

1) При моделюванні фіксуйте напругу якоря, момент або струм якоря та кутову швидкість або ЕРС двигуна у відносних одиницях. У якості базових величин оберіть для напруги, моменту та струму якоря номінальні величини, а для швидкості – швидкість ідеального холостого ходу

$$\omega_0 = \frac{U_{\text{ян}}}{R_{\text{я}}}. \qquad (14.13)$$

Усі графіки зображуйте в одній системі координат.

2) Перехід від кутової швидкості в обертах за хвилину (технічна система одиниць) до кутової швидкості у радіанах за секунду (міжнародна система одиниць СІ) здійснюйте за формулою

$$\omega_{\text{н}} = \frac{\pi n_{\text{н}}}{30}. \qquad (14.14)$$

Після цього можна розрахувати номінальний момент

$$M_{\text{н}} = \frac{P_{\text{н}}}{\omega_{\text{н}}}, \qquad (14.15)$$

коефіцієнт моменту та ЕРС

$$c = \frac{M_{\text{н}}}{I_{\text{ян}}},$$

швидкість ідеального холостого ходу

$$\omega_0 = \frac{U_{\text{ян}}}{c}, \qquad (14.16)$$

статичну просадку швидкості при номінальному навантаженні

$$\Delta\omega_{\text{с}} = \frac{I_{\text{ян}} R_{\text{я}}}{c} \qquad (14.17)$$

та струм короткого замикання

$$I_{\text{кз}} = \frac{U_{\text{ян}}}{R_{\text{я}}}. \qquad (14.18)$$

2) Індуктивність якоря розрахуйте за емпіричною формулою Уманського-Лінвіля

$$L_{\text{я}} = k_L \frac{U_{\text{ян}}}{I_{\text{ян}} \omega_{\text{н}} p}, \qquad (14.19)$$

де $k_L = 0.6$ для двигунів з компенсаційною обмоткою, $k_L = 0.25$ для двигунів без компенсаційної обмотки.

3) Орієнтовний час перехідного процесу для першого і третього режимів складає

$$t_{k1}=t_{k3}\approx 4T_{\text{м}}+T_{\text{я}};$$

для другого режиму

$$t_{k2}=t_{\text{р0}}+t_{k1},$$

де

$$t_{\text{р0}}=\frac{J\omega_0}{M_j}; \qquad (14.20)$$

у четвертому режимі час накидання навантаження розрахуємо як

$$t_{\text{нн}}=t_{k2}+t_{k1},$$

час скидання навантаження – як

$$t_{\text{сн}}=t_{\text{нн}}+t_{k1},$$

час початку гальмування – як

$$t_{\text{пг}}=t_{\text{сн}}+t_{k1}$$

та загальний час перехідного процесу –

$$t_{k4}=t_{\text{пг}}+t_{k2}.$$

14.5 Контрольні питання і завдання

1) Як скласти структурну схему об’єкта моделювання за його математичним описом?

2) Поясніть, чому не можна здійснювати прямий пуск ДПС.

3) На яку фізичну величину впливає значення похідної від напруги якоря при розгоні та гальмуванні двигуна?

4) Як впливає електромагнітна стала часу якірного кола на максимальне значення струму при стрибкоподібній зміні $U_{\text{я}}$.

5) Як впливає електромеханічна стала часу привода на його швидкодію?

6) Як впливає величина активного опору якоря на просадку його швидкості при накиді навантаження?

Лабораторна робота №15

ВІРТУАЛЬНЕ ФІЗИЧНЕ МОДЕЛЮВАННЯ ЕЛЕКТРИЧНИХ КІЛ З ВИКОРИСТАННЯМ БЛОКІВ БІБЛІОТЕК *SimPowerSystem*

Мета роботи: ознайомитися з можливостями використання блоків бібліотек *SimPowerSystem* при моделюванні електротехнічних та електромеханічних об‘єктів.

15.1 Основні відомості про бібліотеку *SimPowerSystem* та правила її застосування

Програма *Simulink* поряд з власними блоками, розглянутими у роботах 1-8 може використовувати блоки бібліотеки *SimPowerSystems (SPS)*, призначеної для віртуального фізичного моделювання електротехнічних, електромагнітних, електронних та електромеханічних пристроїв та схем.

В останніх версіях *Simulink* розширення *SimPowerSystems* входить до складу додатку віртуального фізичного моделювання *SimScape* поряд з фундаментальною бібліотекою процесів *Foundation Library (Electrical*, *Hydraulic*, *Magnetic*, *Mechanical*, *Pneumatic*, *Thermal)* та бібліотеками *SimElectronics*, *SimMechanics*, *SimHydraulics* та *SimDriveline*.

На відміну від *Simulink*-блоків, *SPS*-блоки подано у вигляді позначень відповідних елементів на принципових електричних схемах. З‘єднуючись між собою, ці блоки утворюють електричні кола. Математичний опис окремих елементів приховано від користувача, завдяки чому створюється ілюзія віртуального фізичного моделювання. Насправді ж кожному з блоків *SimPowerSystems* поставлено у відповідність неперервні та дискретні *Simulink*-моделі, які можна побачити після завантаження файлів *powerlib_models*, *powerlib_extras*, *powerlib_meascontrol*. Шлях до цих файлів у версії *R2013a* має вигляд: *ProgramFiles\MATLAB\R2013a\toolbox\physmod \powersys\powersys*.

Отже, не зважаючи на ілюзію фізичного моделювання, користувач повинен чітко уявляти, що *SPS*-блоки створені на основі математичного опису,

складеного розробниками пакету із певними припущеннями, які необхідно знати для адекватного застосування *SPS*-моделей в процесі досліджень. Для цього треба користуватись довідковою інформацією за допомогою функції *help.* Якщо цієї інформації не вистачає, можна проаналізувати відповідні *Simulink*-моделі.

SPS-блоки мають такі особливості [12]:

- їх входи та виходи, на відміну від *Simulink*-блоків (*S*-блоків), не вказують напрямок передачі сигналу, бо вони фактично є еквівалентами електричних контактів;
- лінії зв‘язку між *SPS*-блоками є моделями ідеальних (без опору) електричних проводів, по яким струм може протікати в обох напрямках;
- *SPS*- та *S*-блоки не можуть з‘єднуватися один з іншим безпосередньо; сигнал від S-елементів можна передати до *SPS*-елементів тільки через керовані джерела енергії (Controlled Voltage/Current *Source*) *SPS*-бібліотеки *Electrical Sources*, а навпаки – через блоки бібліотек засобів вимірювання (*Measurements*);
- в моделі, яка отримує в собі *SPS*-блоки, має бути присутнім хоча б один з вимірювальних *SPS*-приборів, що пов‘язано з особливостями перетворення *SPS*-моделі в еквівалентну розрахункову S-модель;
- в *SPS*-модель необхідно встановлювати блок *powergui*.

Зміст бібліотеки *SimPowerSystems* показано на рис. 15.1.

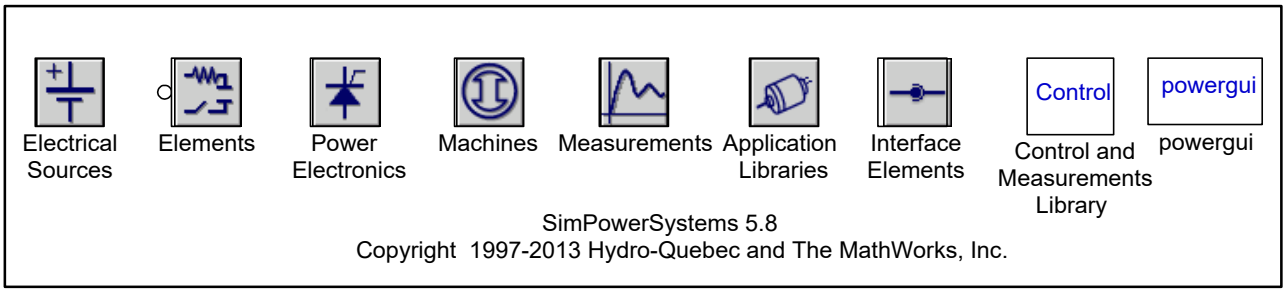


Рис. 15.1 Розділи бібліотеки *SimPowerSystems*

Вона отримує такі розділи:

- *Electrical Sources* – джерела електричної енергії;

- *Elements* – пасивні електротехнічні елементи;
- *Power Electronics* – пристрої силової електроніки, керовані напівпровідникові ключі;
- *Machines* – моделі електричних машин;
- *Measurements* – вимірювальні прибори;
- *Application Libraries* – прикладні бібліотеки;
- *Interface Elements* – інтерфейсні елементи;
- *Control and Measurements Library* – бібліотека керування та вимірювання;
- *PowerGUI* – графічний інтерфейс користувача.

Перелік блоків *SimPowerSystems*, необхідний для моделювання та аналізу властивостей розгалужених лінійних електричних кіл постійного і змінного струмів, надано на рис 15.2. Вони знаходяться у різних розділах бібліотеки *SPS*.

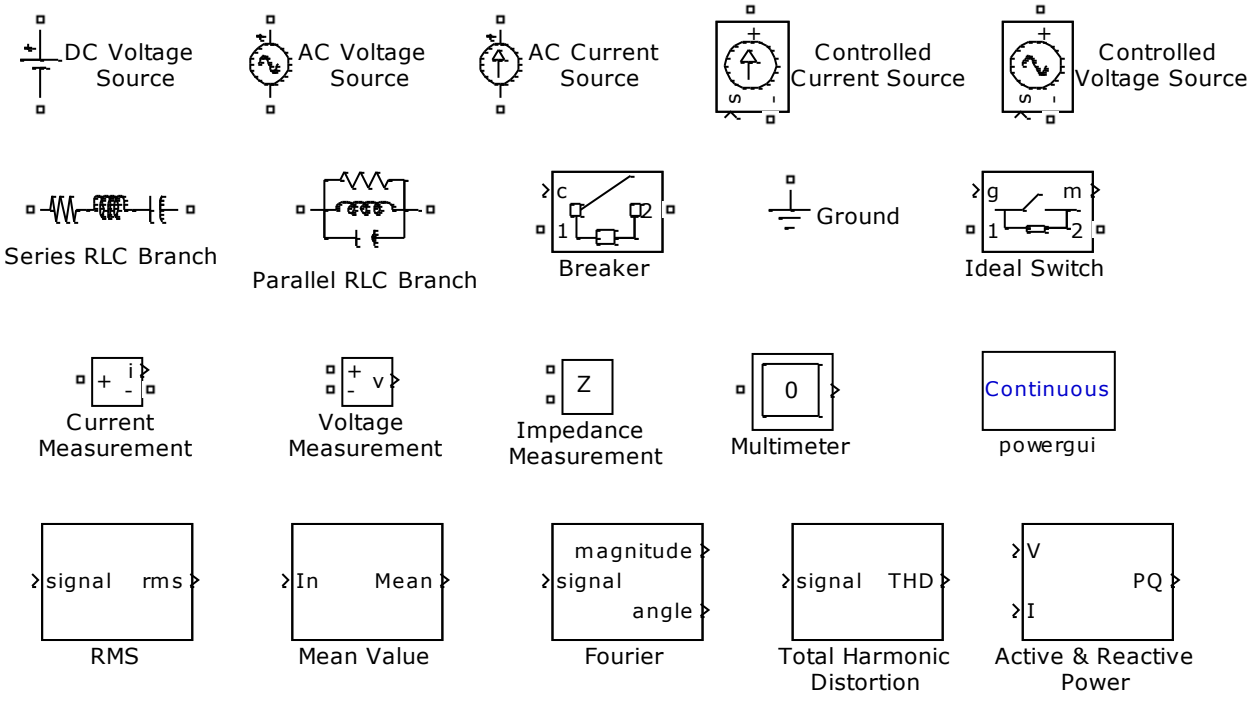


Рис. 15.2. *SPS*-блоки для віртуального фізичного моделювання електричних кіл

Подані блоки мають порти двох типів: *SPS*-порти, позначені дрібними квадратиками та *S*-порти, позначені кінцівками стрілок, що допомагає запобігти помилок при з‘єднанні блоків.

Параметри блоків можна розділити на ***основні***, які пов‘язані безпосередньо з фізичною сутністю блоків і без встановлення яких робота блоків принципово не можлива, і ***допоміжні***, які в багатьох випадках можна залишати такими, що встановлені за замовченням.

До допоміжних параметрів відносяться:

1) *Measurements* – здійснює вибір сигналів, які стають доступними для вимірювання блоком *Multimeter*. У меню, що випадає, можна обрати такі значення цього параметру:

- *None* – немає сигналів, доступних для вимірювання:
- *Voltage* – (для джерел напруги);
- *Current* – можна виміряти струм (для джерел струму);
- *Branch voltage* – можна виміряти напругу *RLC*-гілки або будь-якого сполучення резистора, котушки індуктивності та конденсатора (для блоків *RLC Branch* та *RLC Load*);
- *Branch current* – можна виміряти струм *RLC*-гілки;
- *Branch voltage and current* – можна виміряти струм і напругу *RLC*-гілки.

2) *Set the initial inductor current* у сполученні з параметром *Inductor initial current (A)*, дозволяють встановити початкове значення струму індуктивного елемента (для блоків *RLC Branch* та *RLC Load*).

3) *Set the initial capacitor voltage* у сполученні з параметром *Capacitor initial voltage (V)*, дозволяють встановити початкове значення напруги конденсатора (для блоків *RLC Branch* та *RLC Load*).

4) *Sample Time* – період дискретизації; встановлюється тільки при моделюванні дискретних процесів; при моделюванні неперервних процесів його залишають нульовим (0).

При створенні віртуальних фізичних моделей електричних кіл дотримуйтесь таких правил:

1) Не забувайте встановити блок *powergui.*

2) Враховуйте, що в *SPS*-моделях, на відміну від *S*-моделей, спочатку розраховується усталений режим.

3) Для вимірювання напруг та струмів треба, як мінімум, або підключити у схему амперметри та вольтметри, або скористатися мультиметром. До вихідних *S*-портів вимірювальних *SPS*-блоків приєднайте якісь із блоків *Simulink*-бібліотеки *Sinks*. Що здійснюють реєстрацію та/або візуалізацію результатів.

4) Розгалуження електричного кола не можна виконати з точок, що позначають *SPS*-порти та з точок між *SPS*-портами одного блоку. Тому, якщо Ви хочете, наприклад, виміряти напруги на кожному з елементів послідовної *RLC*-гілки, то треба включити послідовно 3 блоки *RLC Branch*, один з яких зробити чисто активним, другий – чисто індуктивним, а третій – чисто ємнісним.

5) Для формування напруг та струмів складної форми використовуйте будь-які *S*-блоки, сформований сигнал подавайте на вхідні *S*-порти контрольованих *SPS*-джерел електричної енергії *Controlled Voltage*/*Current Source* з *SPS*-бібліотеки *Electrical Sources*.

6) При виборі методу та параметрів чисельного інтегрування диференційних рівнянь, що здійснюється при виконанні команди *Mogel Configurations Parameters* меню *Simulink*, ретельно аналізуйте можливий характер та тривалість перехідних процесів. При недостатній кількості розрахованих точок для якісної візуалізації переходьте від методу *ode45* (вкладка *Solver*), що пропонується за замовченням до методів *ode23s*, *ode15s*, *ode23tb* або при використанні методу *ode45* встановлюйте значення параметра *Refine Factor* (вкладка *Data Import/Export*, *Refine* – підвищувати якість) більшим за 1 (максимальне значення дорівнює 10, рекомендоване – 5), при використанні якого щільність точок збільшується в установлене число разів методом інтерполювання. У колах змінного струму для якісної візуалізації іноді достатньо встановити максимальний шаг інтегрування (*Max Step Size*) таким,

щоб на періоді синусоїди розраховувалась достатня кількість точок, наприклад, $h_{\max} = 1/(50 \div 100) f$.

7) Для більшої наочності доцільно більшість блоків перейменувати у відповідності з досліджуваною електричною схемою, а імена вимірювальних блоків приховати (*Diagram→Format→Hide Block Name*), якщо їх призначення зрозуміло із піктограми блоку.

15.2 Характеристика основних блоків для моделювання кіл постійного струму та однофазних кіл змінного струму

Розглянемо призначення та основні параметри блоків, наведених на рис. 15.2.

15.2.1 Джерела електричної енергії

1) ***DC Voltage Source*** – ідеальне джерело постійної напруги з основним параметром *Amplitude (V)* – величина постійної напруги U_c (В).

2) ***AC Voltage Source*** – ідеальне джерело синусоїдальної напруги (з нульовим активним опором) з параметрами

- *Peak Amplitude (A)* – амплітуда напруги U_m (А),
- *Phase (deg)* – початкова фаза φ° (град.),
- *Frequency (Hz)* – частота f (Гц), що формує напругу

$$U(t) = U_m \sin\left(2\pi f t + \frac{\varphi^\circ}{180^\circ}\pi\right). \qquad (15.1)$$

3) ***AC Current Source*** – – ідеальне джерело синусоїдального струму (з нульовою провідністю) з параметрами

- *Peak Amplitude (A)* – амплітуда струму I_m (А),
- *Phase (deg)* – початкова фаза φ° (град.),
- *Frequency (Hz)* – частота f (Гц), що формує струм

$$I(t) = I_m \sin\left(2\pi f t + \frac{\varphi^\circ}{180^\circ}\pi\right). \qquad (15.2)$$

Оскільки джерело постійного струму у бібліотеках *SPS* відсутнє, то замість його можна використати блок *AC Voltage Source* з параметрами:

$I_m = I_c$, $f = 0$, $\varphi^\circ = 90^\circ$.

4) ***Controlled Voltage Source***, ***Controlled Current Source*** – керовані джерела напруги і струму. Використовуються для створення джерел, які виробляють струм або напругу, що відрізняються від постійної величини або від синусоїдальних сигналів (15.1), (15.2). Для цього треба сформувати бажані сигнали в *Simulink* і подати їх на *S*-входи керованих джерел електроенергії, які перетворюють *S*-сигнали у *SPS*-напруги або *SPS*-струми, що живлять віртуальні електричні кола через *SPS*-контакти, позначені як «+» та «-».

При використанні керованих джерел за прямим призначенням немає сенсу виконувати ініціалізацію контрольованих джерел.

Окрім *S*-блоків, включених до складу базових *Simulink*-бібліотек, до *SPS*-бібліотеки також залучено деякі *S*-блоки, які можуть стати у пригоді при створенні та при аналізі віртуальних електротехнічних схем. Зокрема, у розділі *Control and Measurements Library* знаходиться група блоків *Pulse & Signal Generators*, які можна застосовувати при формуванні нестандартних джерел напруги. Частина з них показана на рис. 15.3.

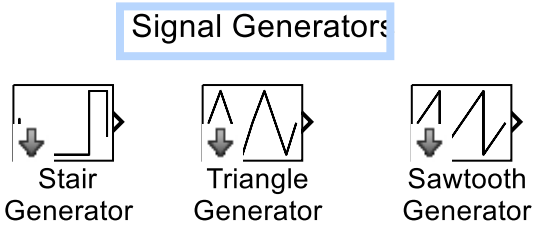


Рис. 15.3. Блоки із групи *Pulse & Signal Generators*

Блок *Stair Generator* (*Ступінчатий генератор*) формує послідовність ступінчатих сигналів на основі блоку *Look-Up Table*, як це показано на рис. 15.4. Його параметрами є координати точок, в яких відбувається ступінчата зміна вихідного сигналу, а саме: вектор часу – *Time (s)* та вектор амплітуд –

Amplitude. Його зручно використовувати для керування станом блоку *Ideal Switch*.

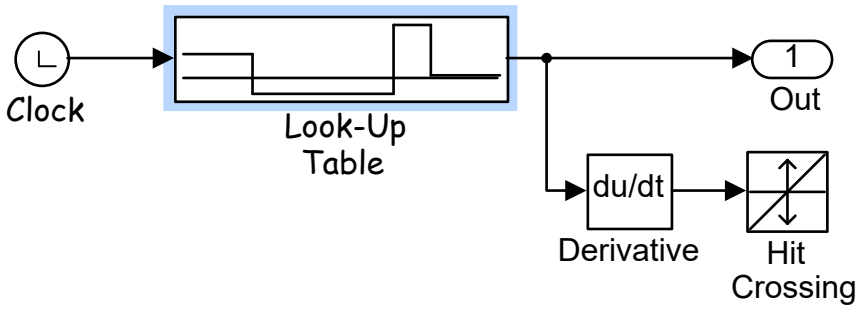


Рис. 15.4. Структура блоку *Stair Generator*

Блок *Triangle Generator* (*Трикутний Генератор*) формує трикутний, а блок *Sawtooth Generator* пилоподібний періодичні сигнали одиничної амплітуди з заданою частотою *Frequency (Hz)* та фазою *Phase (degree)*. Структура блоку *Sawtooth Generator* показано на рис. 15.5.

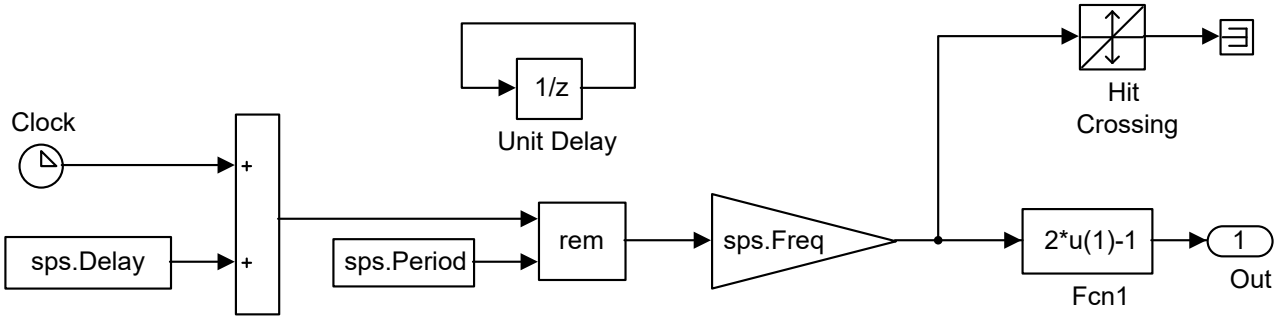


Рис. 15.5. Структура блоку *Sawtooth Generator*

15.2.2 Електротехнічні елементи

1) ***Series RLC Branch*** – послідовне *RLC*-з‘єднання з такими основними параметрами:

- *Branch type* – визначає комплектність з‘єднання, обирається із випадаючого меню і може приймати одне із значень: *RLC, RL, RC, LC, R, L, C*;
- *Resistance R (Ohms)* – активний опір, Ом;
- *Inductance L (H)* – індуктивність, Гн;
- *Capacitance C (F)* – ємність, Ф.

Вікно введення параметрів цього блоку відображене на рис. 15.6.

2) ***Parallel RLC Branch*** – паралельне *RLC*-з‘єднання з такими ж основними параметрами, як і блок *Series RLC Branch.*

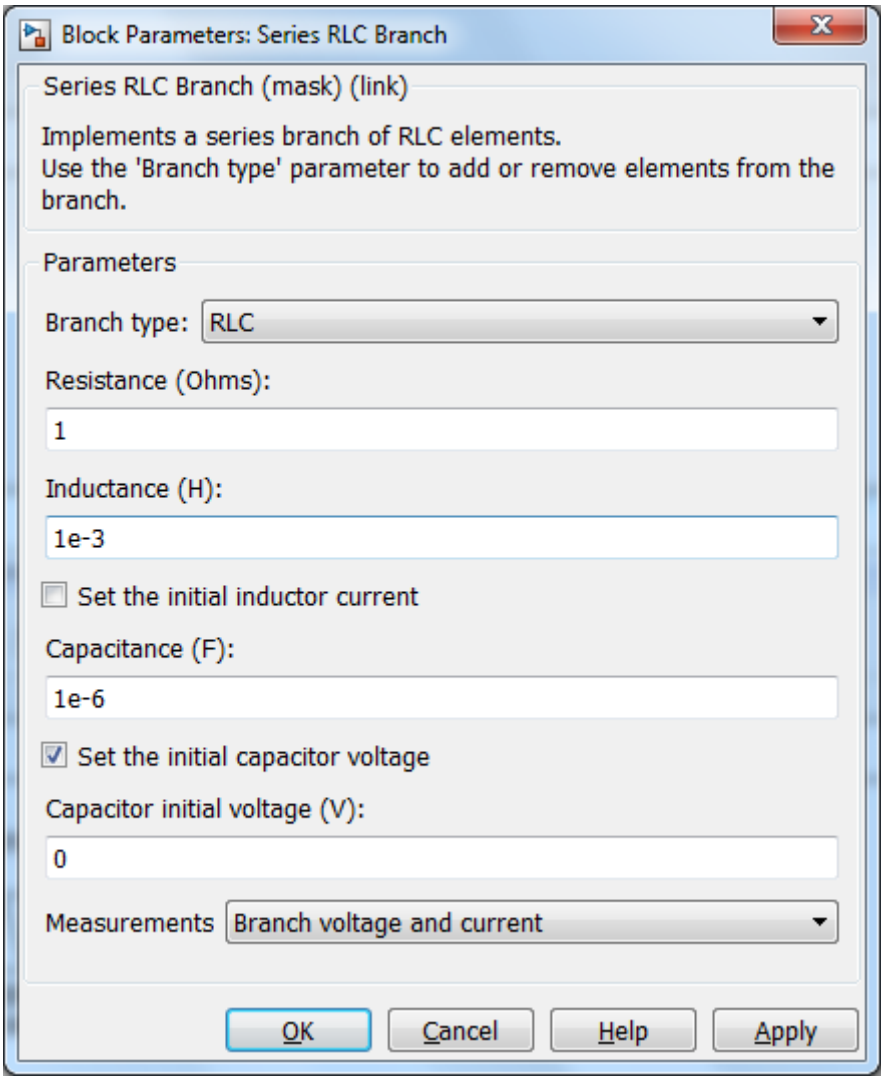


Рис. 15.6. Вікно параметрів блоку *Series RLC Branch*

15.2.3 Основні вимірювальні прибори

1) ***Voltage Measurement*** – ідеальний вольтметр.

Перетворює різницю потенціалів точок, приєднаних до входів «+» і «–», в еквівалентний *S*-сигнал на виході «v», який можна перетворювати, фіксувати або візуалізувати будь-яким блоком *Simulink*-бібліотек.

2) ***Current Measurement*** – ідеальний амперметр.

Перетворює *SPS*-сигнал електричного струму, що протікає у гілці, в яку включено прибор через контакти «+» і «–», в еквівалентний *Simulink*-сигнал на

S-виході «i», який можна перетворювати, фіксувати або візуалізувати будь-яким блоком *Simulink*-бібліотек.

3) ***Multimeter*** – багатоканальний вимірювач струмів та напруг.

Сигнали, доступні для вимірювання цим прибором повинні бути обрані у блоках елементів схеми за допомогою допоміжного параметра *Measurement*, як це описано у пункті 15.2.

Якщо такі сигнали існують, то після подвійного щиглика по піктограмі (іконці) блока відчиняється вікно, зображене на рис. 15.7.

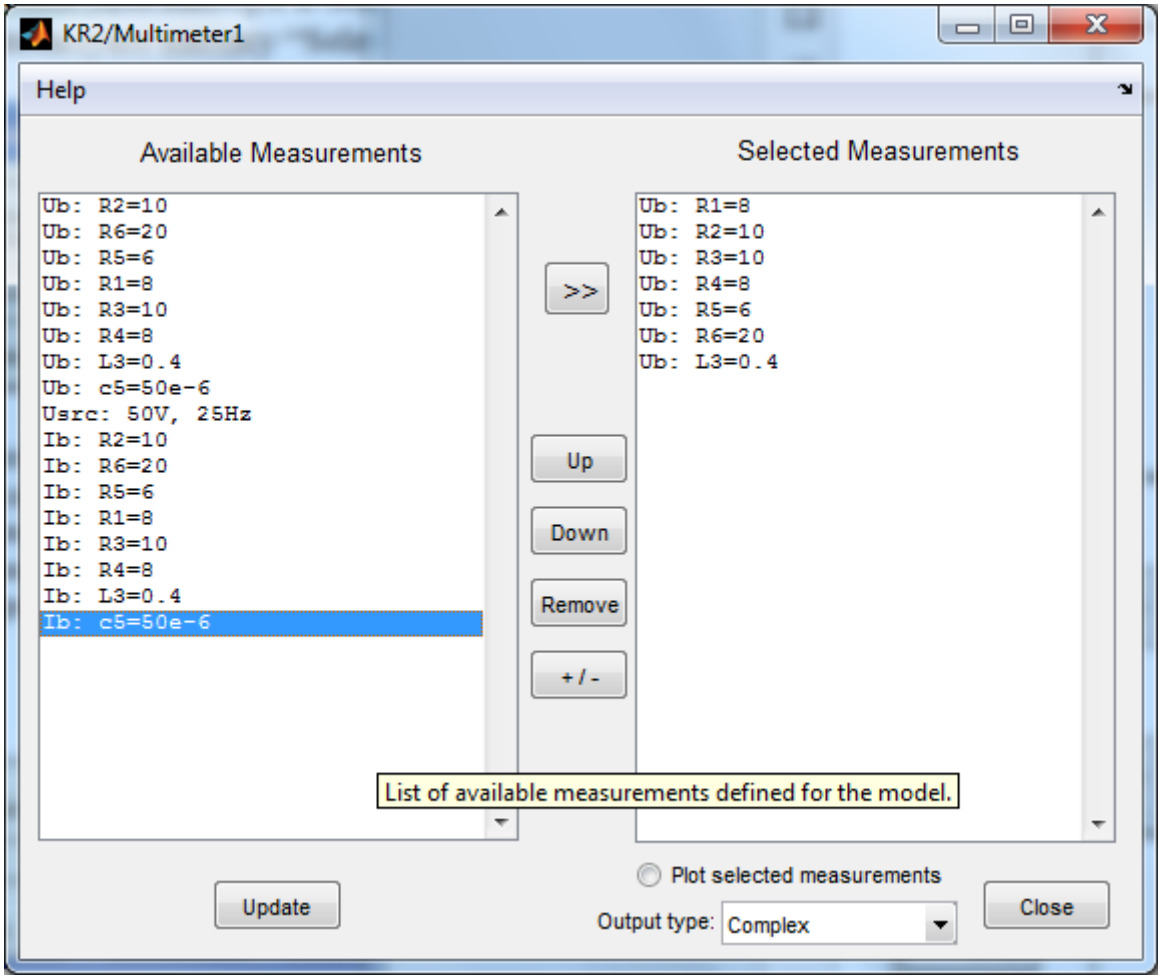


Рис. 15.7. Вікно настроювання блоку *Multimetr*

У колонці *Available Measurements* відображається список величин, доступних для вимірювання, а у колонці *Selected Measurements* – список величин, обраних для вимірювання.

Для вибору колонці *Available* виділяють мишкою потрібний сигнал і кнопкою >> дублюють його у колонку *Selected Measurements*.

Видалити сигнал з колонки *Selected Measurements* можна кнопкою *Remove*. Змінити порядок сигналів у колонці *Selected Measurements* можна кнопками *Up* і *Down*. Кнопкою +/- можна міняти полярність обраних сигналів. На піктограмі блока автоматично відображується кількість обраних сигналів.

Обновити список доступних сигналів після внесення змін у віртуальну модель можна кнопкою *Update*.

Якщо встановити прапорець у полі параметру *Plot selected measurements*, то після розрахунку перехідних процесів відкриється вікно з графіками обраних сигналів. Отже, у самому простому випадку у блоці можна не використовувати а ні вхідні, а ні вихідні порти.

4) ***Impedance Measurement*** – вимірювач повного (комплексного) опору (імпедансу) ділянки електричного кола у заданому діапазоні частот.

Відображення амплітудної та фазової частотних характеристик імпедансу відбувається за допомогою блоку *Powergui* у вікні, що відчиняється кнопкою *Impedance vs Frequency Measurements* (див. рис. 15.8). У цьому вікні можна скоригувати діапазон частот, нанести координатні сітки, обрати логарифмічний або лінійний масштаби зображення частотних характеристик, запам‘ятати результати розрахунків.

При використанні вимірювача повного опору слід мати на увазі, що цей блок виконано на основі джерела струму. Тому його не можна включати послідовно з індуктивними елементами. Для усунення цього обмеження ***блок Impedance Measurement при наявності у вимірюваній ділянці котушок індуктивності слід шунтовати резистором із достатньо великим опором.*** Величину опору треба обирати такою, щоб властивості схеми суттєво не змінювались.

Для вимірювання комплексних опорів пасивних елементів необхідно вилучити зі схем джерела енергії.

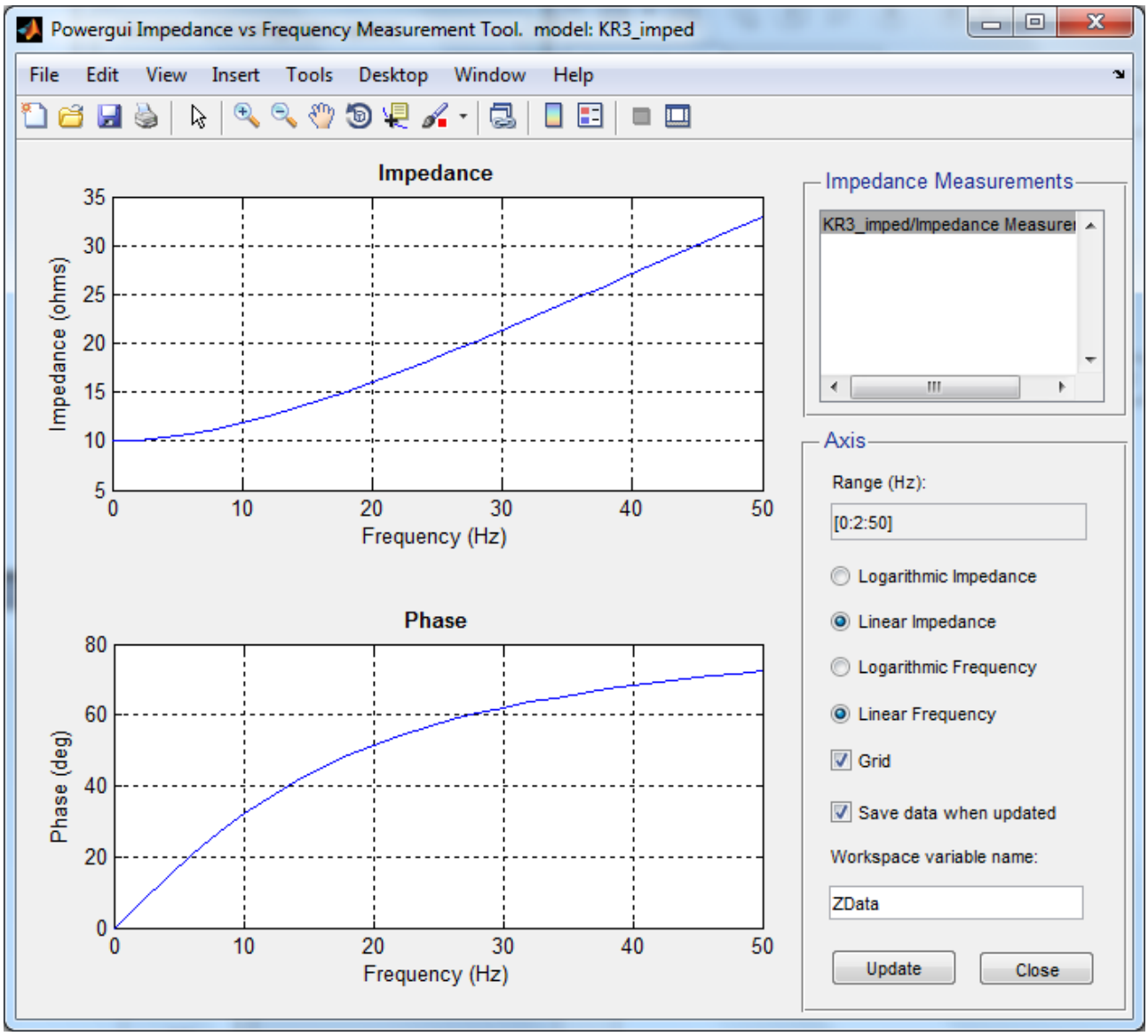


Рис.15.8. Вікно відображення частотних характеристик комплексного опору ділянки електричного кола

15.2.4 Ключові елементи

1) ***Breaker*** – перемикач, що здійснює комутацію між контактами 1 і 2, який може керуватися як у функції зовнішнього сигналу, що подається на керований вхід "*c*" (*External control mode*), так і за допомогою внутрішнього інтервального таймера (*Internal control mode*) (див. рис. 15.9).

Початковий стан ключа задається параметром *Initial State (0 for 'open', 1 for 'close')*. ***Параметр Breaker resistance Ron (Ohm)*** визначає внутрішній опір ключа у замкненому стані, який ***не можна встановлювати нульовим***. У розімкненому стані ключ має опір *Roff* = ∞.

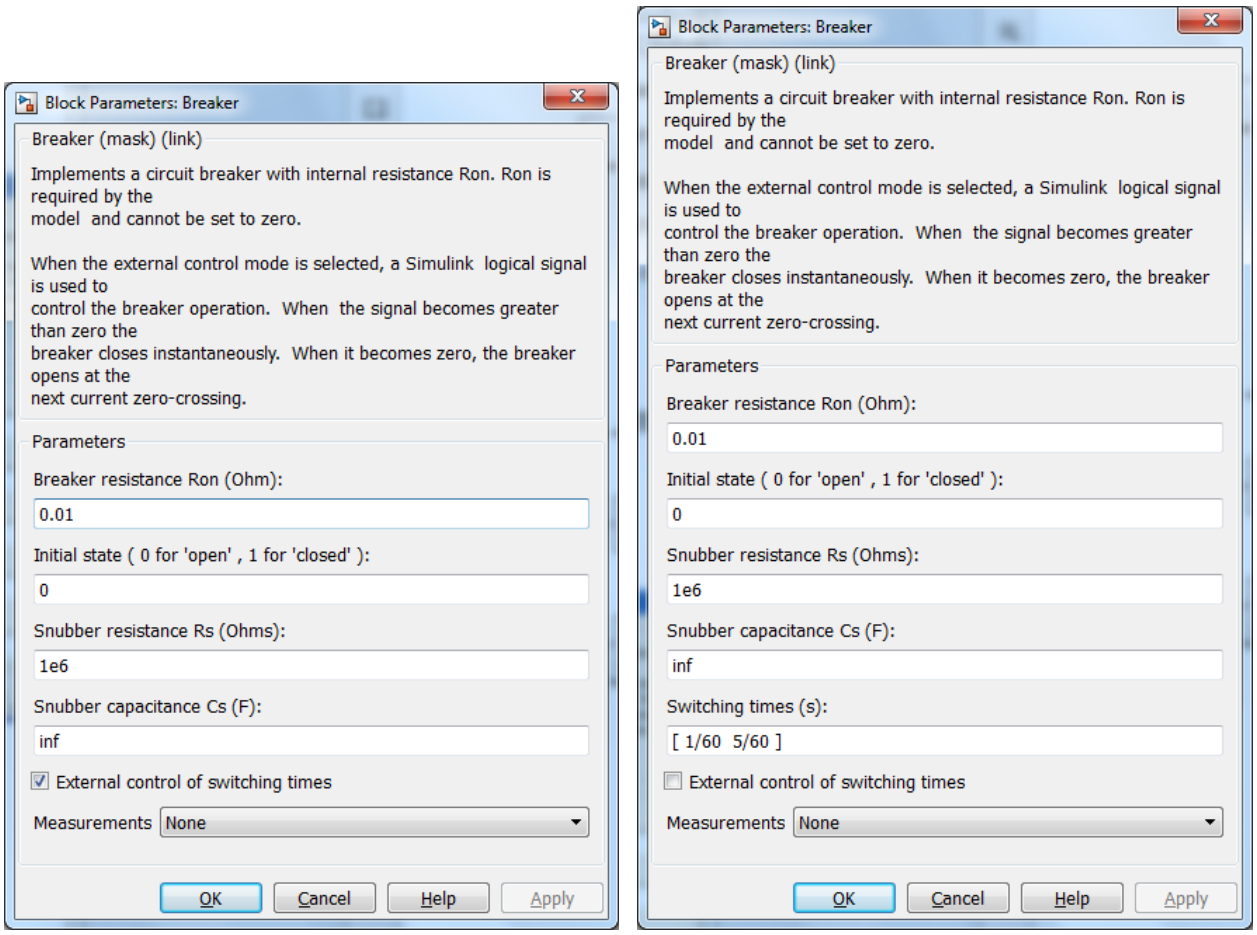
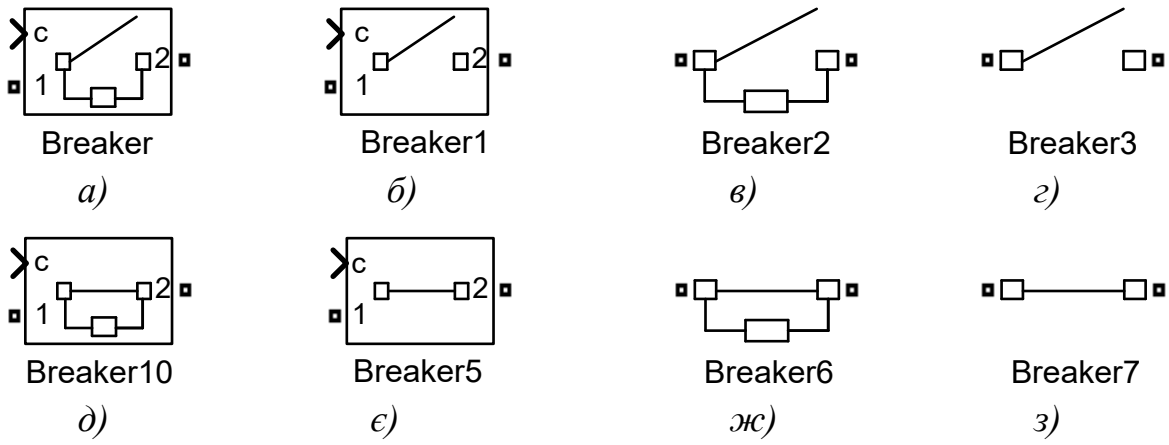


Рис. 15.9. Вікна параметрів блоку *Breaker*

У модель «брейкера» включено послідовну *Rs*-*Cs*-гілку (*Snubber*), що здійснює функцію дугогасіння при розмиканні ключа, з параметрами *Snubber resistance Rs (Ohm)* та *Snubber capacitance Cs (F)*. ***За замовчанням*** параметр *Cs* має значення *inf* (∞), тобто ***«снабер» є чисто резистивним***. ***Встановлення Cs = 0 рівнозначно розмиканню «снабера».*** При цьому змінюється «іконка» блоку (див. рис. 15.10). «Снабер» необхідно застосовувати, якщо послідовно до «брейкера» приєднані котушка індуктивності або джерело струму.

У режимі *External control mode* на іконці ключа з‘являється *S*-вхід "*c*", що дозволяє здійснити керування станом ключа зовнішнім логічним сигналом: 1 подає команду на замикання ключа, а 0 – на розмикання. При відключенні режиму *External control mode* вхідний порт зникає, а у вікні параметрів з‘являється поле *Switching times (s)*, в якому треба задати вектор моментів часу, в які стан ключа змінюється на протилежний.



а) *б)* *в)* *г)*

д) *є)* *ж)* *з)*

Рис. 15.10. Піктограми блоку *Breaker*:
Initial State = 0 – *а, б, в, г*; *Initial State* = 1 – *д, є, ж, з*;
External control mode – а, б, д, є; Internal control mode – в, г, ж, з;
Cs = inf – а, в, д, ж; Cs = 0 – б, г, є, з

Особливістю блоку ***Breaker*** є те, що він ***розмикає електричне коло після подачі сигналу на розмикання тільки тоді, коли струм, що протікає через ключ, стає нульовим***. Отже, ***Breaker може замкнути, але не може розімкнути коло постійного струму***. ***Його призначення – комутація у колах змінного струму.***

При наявності у віртуальній моделі блоку *Breaker* ***краще за все використовувати для розрахунку перехідних процесів метод ode23tb***.

2) ***Ideal Switch*** – ідеальний ключ, керований *Simulink*-сигналом, що подається на вхідний *S*-порт *gate (g)*: ключ замикається при додатному керуючому сигналі, що перевищує 1, а розмикається при нульовому. Вихідний вимірювальний *S*-порт *m* у включеному стані виводить струм та напругу ключа.

Вікно параметрів цього блоку показано на рис. 15.11.

Як бачимо, він має багато спільного із блоком *Breaker.* Це і початковий стан ключа, який задається параметром *Initial State (0 for ‘open’, 1 for ‘close’)*, і наявність резистивного *«снабера»*, який можна вимкнути, встановленням *Cs* = 0, що відображається на вигляді піктограми блок (див. рис. 15.9), і наявність внутрішнього опору ключа *Internal resistance Ron (Ohm)*.

Але, на відміну від блоку *Breaker*, блок *Ideal Switch* розмикає електричне коло одразу після того, як на порт приходить нульовий сигнал. Тому ***цей блок може повноцінно застосовуватися для комутації у колах постійного струму***.

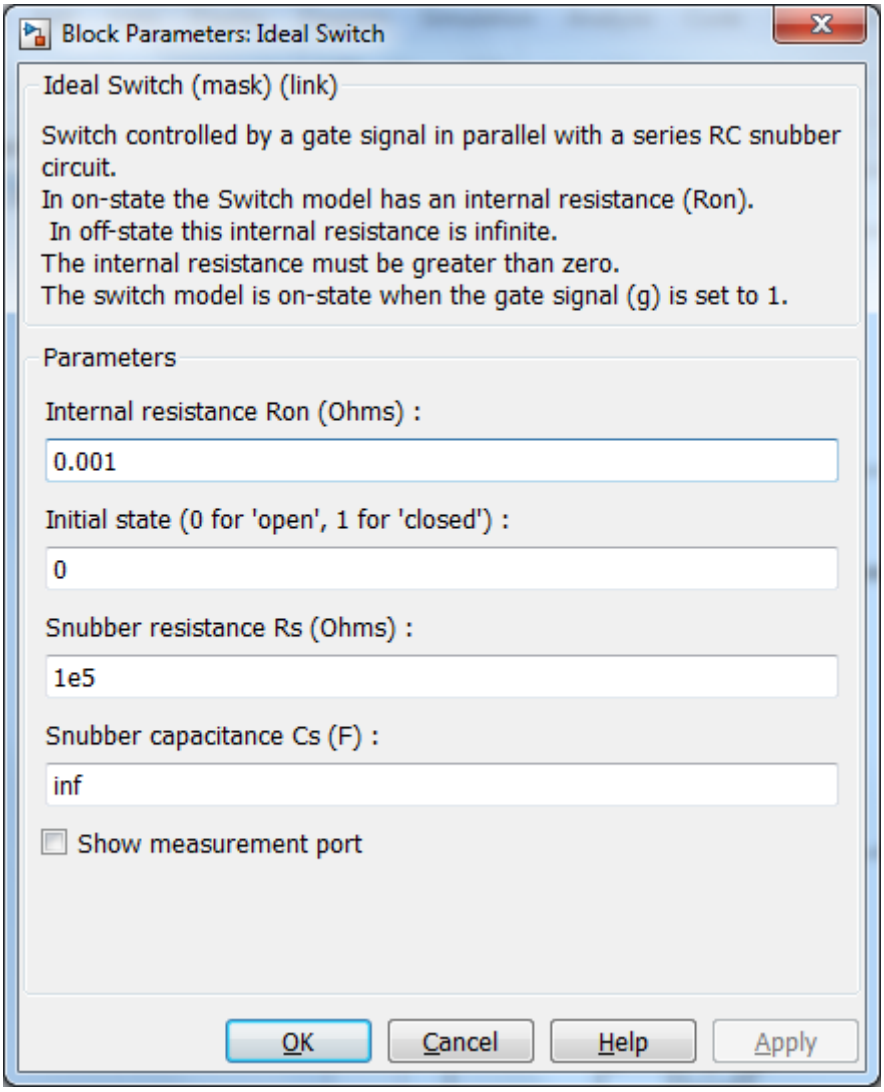


Рис. 15.11. Вікно параметрів блоку *Ideal Switch*

Менш суттєвими відмінностями є

- відсутність внутрішнього таймера для керування стану ключа
- відсутність внутрішнього таймера для керування стану ключа
- наявність вимірювального порту, який можна робити видимим і невидимим за допомогою параметра *Show measurement port*;

- на порядок менші опори *Ron* (0.001 Ом за замовченням замість 0.01 Ом у «брейкера») і *Rs* (10 кОм у порівнянні зі 100 кОм у «брейкера»);
- не змінює піктограму при замкненому початковому стані (див. рис. 15.9).

15.2.5 Графічний інтерфейс користувача *Powergui*

Powergui (Powerlib Graphical User Interface) призначений для аналізу віртуальних фізичних моделей електричних кіл і систем.

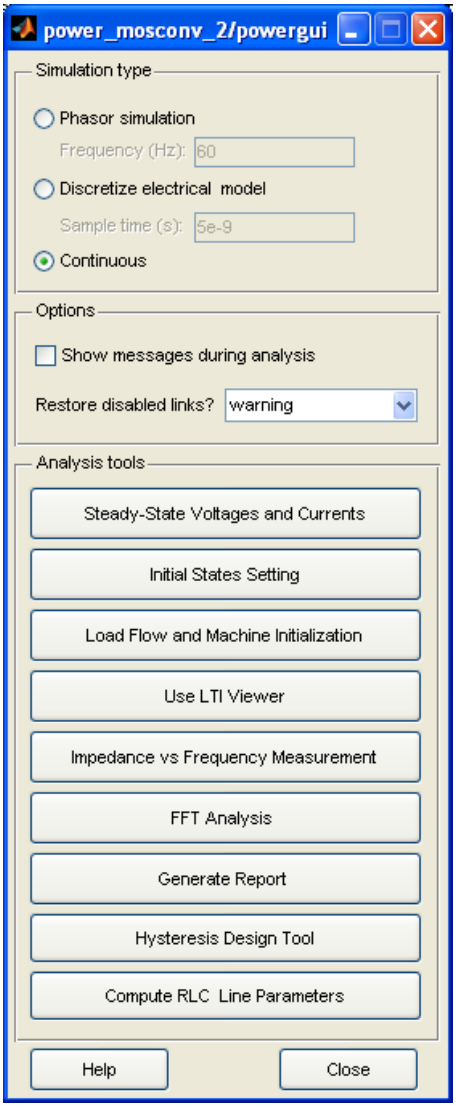


Рис. 15.12. Вікно параметрів блоку *Powergui*

Блок обов‘язково треба розміщати у вікні моделі, у якій присутній хоча б один з блоків бібліотек *SimPowerSystem* (у версіях MATLAB 7 і вище поміщається у вікно моделі автоматично). Блок повинен бути тільки один, і його не можна перейменовувати. Вікно завдання параметрів цього блоку подано на рис. 15.12.

Він дозволяє вирішувати такі задачі:

1) ***Simulation type*** – вибір методу розрахунку динамічних та статичних режимі, зазвичай у вигляді вікна вибору однієї з опцій: *Continuous* (неперервні), *Discretize electrical model* (дискретні, треба вказати період дискретності *Sample Time*), *Phasor simulation* (метод узагальнених векторів, треба вказати основну частоту).

2) ***Steady-State Voltages and Currents*** – відображення усталених значень перемінних (вікно *steady state values*).

Вікно визначення параметрів цієї операції зображено на рис. 15.13.

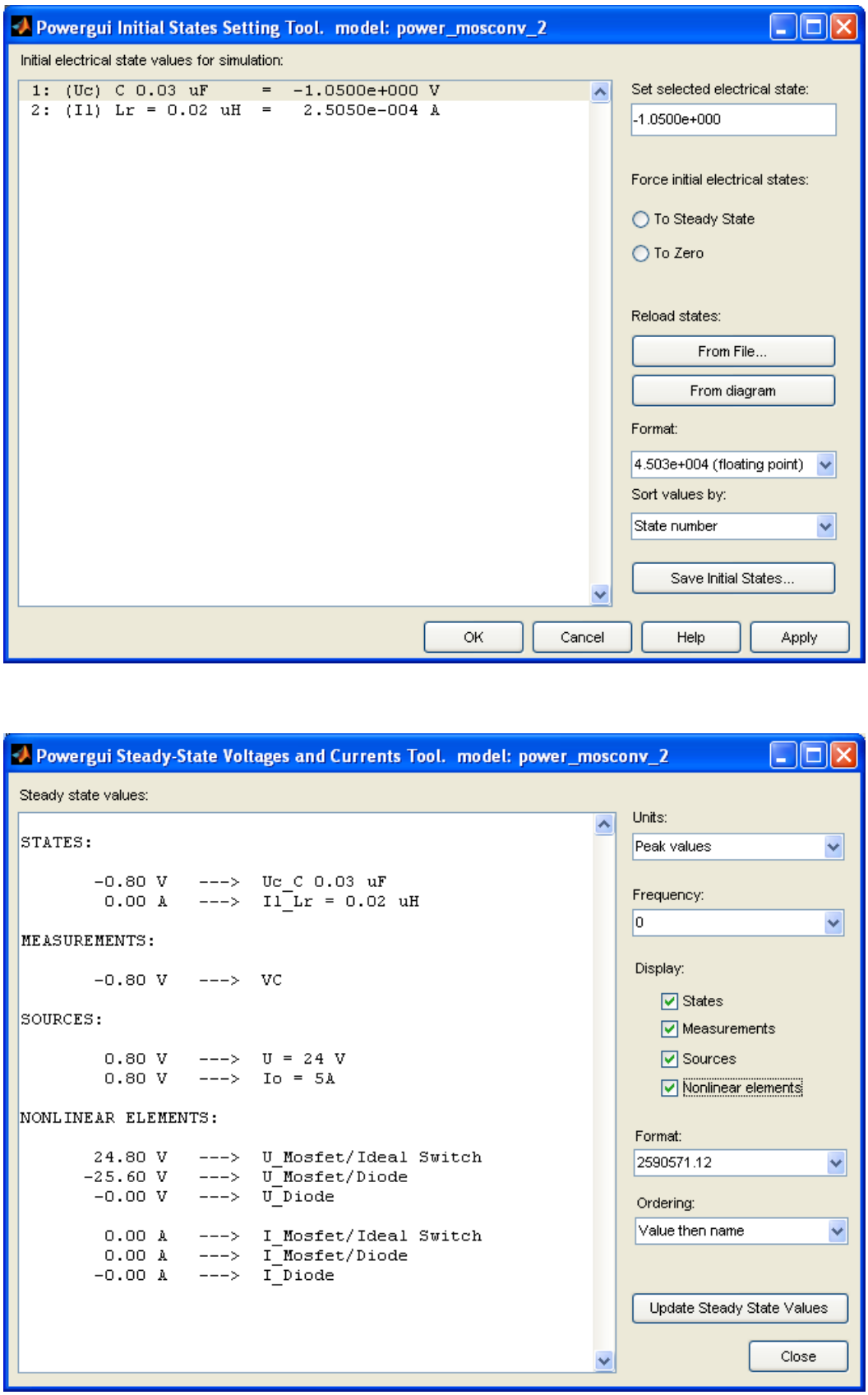


Рис. 15.13. Вікно відображення усталених значень

До них належать:

- *Units* (одиниці) – *peak values* (амплітудні значення) / *RMS values* (діючі значення);
- *Frequency* (частота в Гц);
- *Display* – прапорцями слід задати одну/один або декілька змінних стану (*States*), блоків вимірювання (*Measurements*) та джерел (*Sources*);
- *Format* (формат виведення чисел) – з фіксованою точкою або плаваючою;
- *Ordering* – порядок виводу (*Value then name*), *Name then value*

Оновлення інформації досягається натисканням на кнопку *Update steady state values*.

3) ***Initial States setting*** – редагування або установка початкового стану моделі (завдання початкових умов змінним стану), дозволяє виконати ініціалізацію моделі, визначаючи початок моделювання з усталеного режиму або з нульових початкових умов. Вікно визначення початкових умов подано на рис. 15.14.

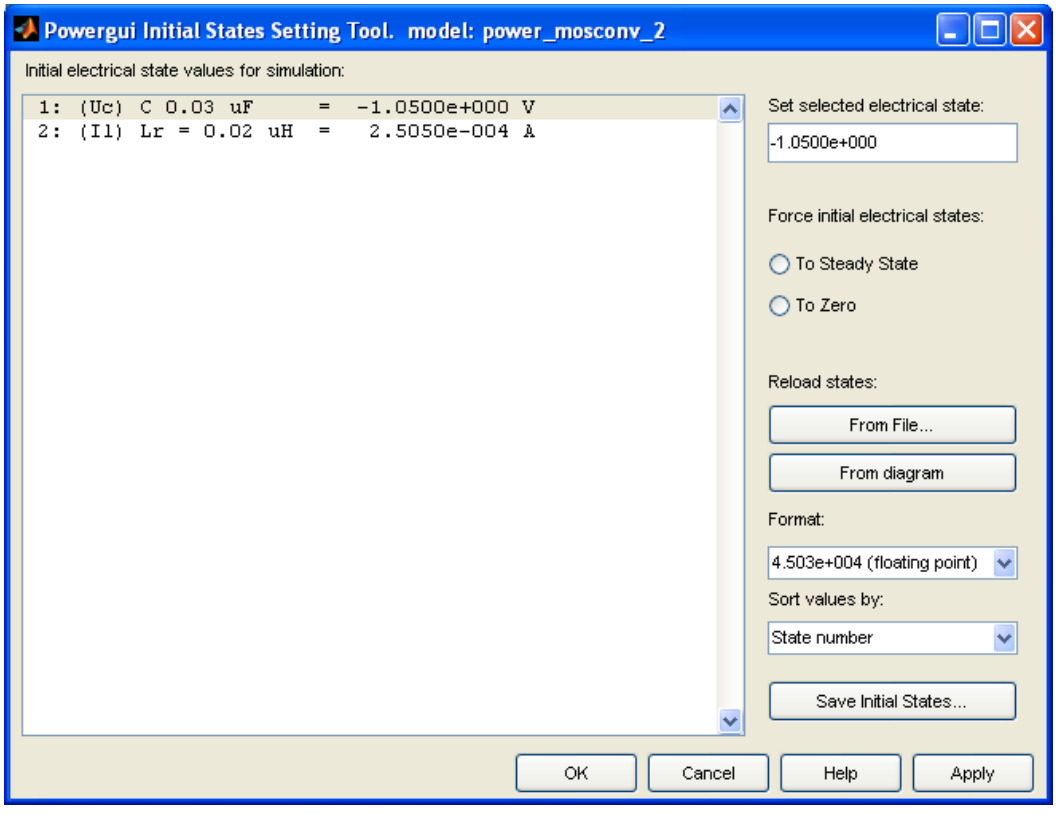


Рис. 15.14. Вікно відображення усталених значень

4) ***Load Flow and Machine Initialization*** – ініціалізація трифазних кіл (*Three-Phase Dynamic Load*) та електричних машин (*Machines*). Вікно для виконання цієї операції зображено на рис. 15.15. Більш детально ознайомитися з можливостями цієї операції можна у [Черних].

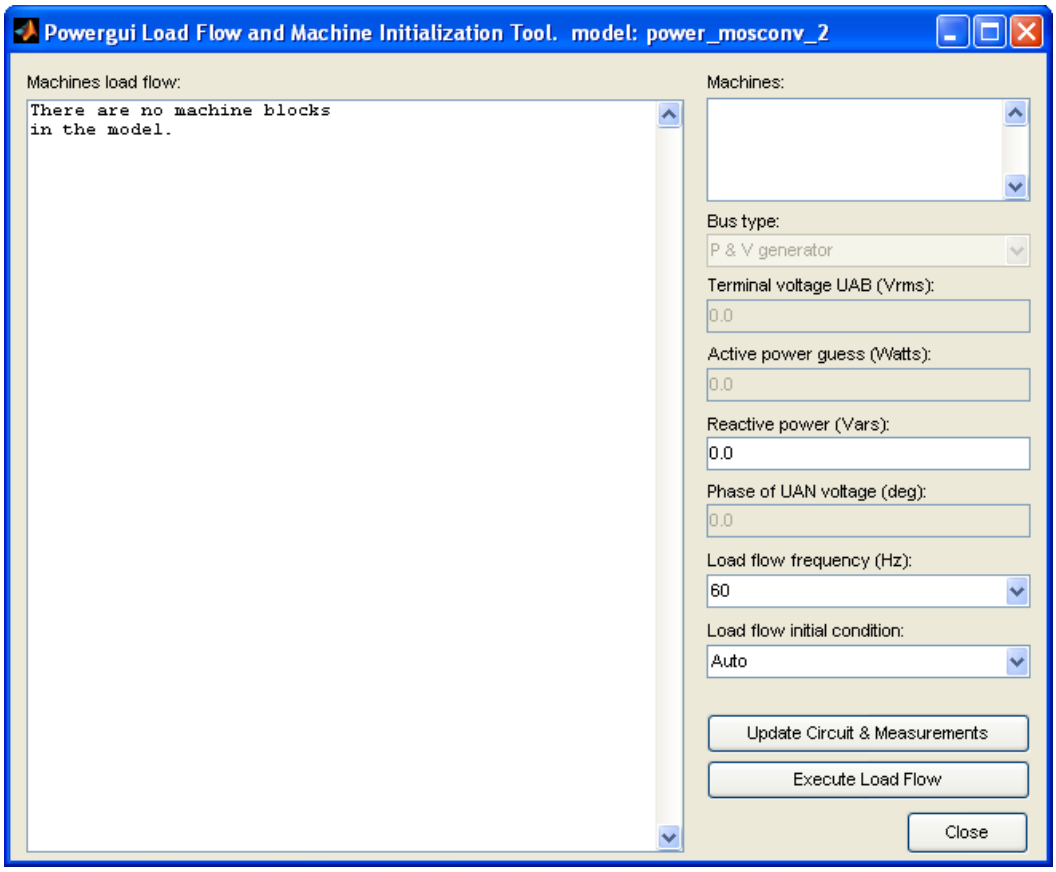


Рис. 15.15. Вікно ініціалізації динамічних трифазних електричних кіл та електричних машин

Інші функції інтерфейсу користувача будуть розглянуті пізніше у інших дисциплінах або при необхідності самостійно.

15.3 Приклад віртуального фізичного моделювання лінійного розгалуженого електричного кола

Перевагою віртуального фізичного моделювання є те, що для його застосування не потрібно складати математичний опис досліджуваної системи. Ця функція покладається на програмне забезпечення. Задачею користувача в даному разі є складання принципової електричної схеми у блоках віртуальної

бібліотеки *SimPowerSystem*, що потребує деякого досвіду. Для прикладу на рис. 15.16 подана *SPS*-модель електричного кола, зображеного на рис. 13.1.

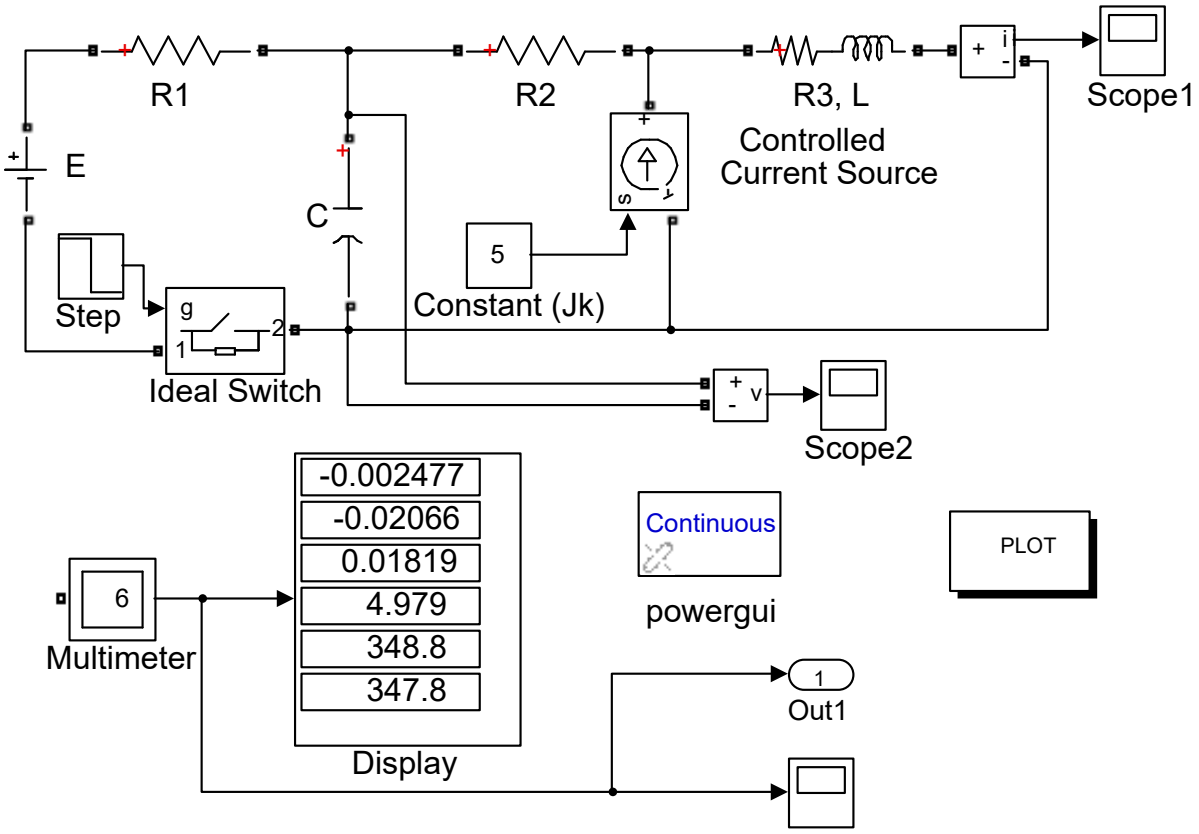
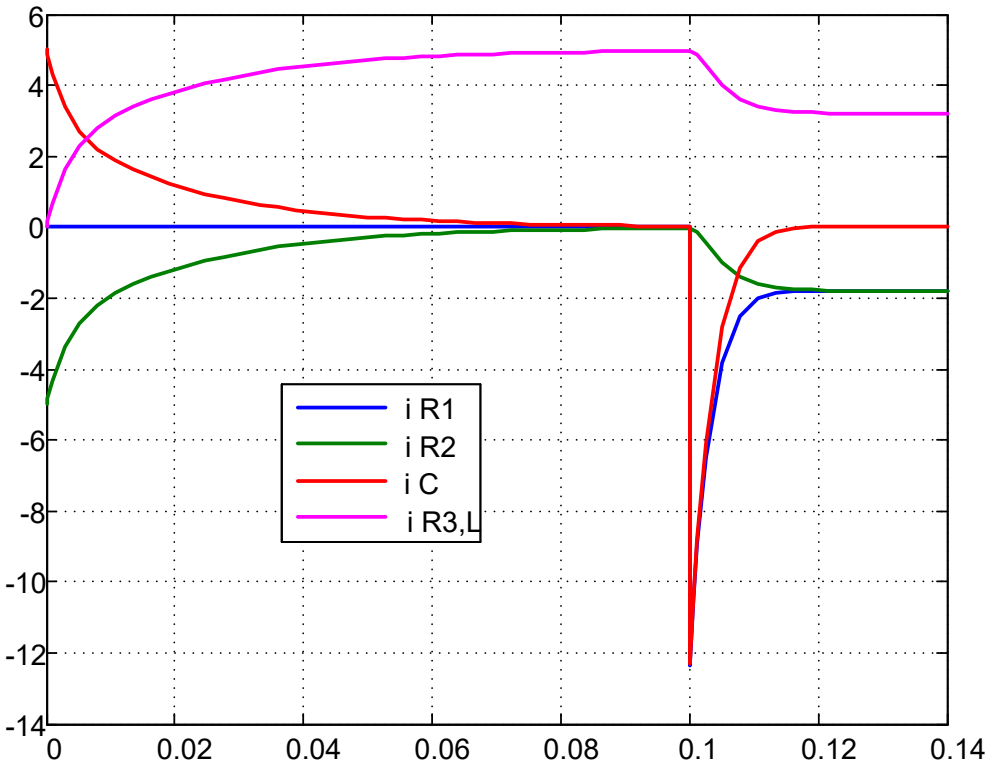


Рис. 15.16. *SPS*-модель лінійного електричного кола, зображеного на рис. 13.1

Simulink-блоки у моделі рис. 15.16 використані тільки для фіксації результатів, формування вихідного сигналу джерела струму та керування ключовим елементом *Ideal Switch*. На рисунку продемонстровано можливість вимірювання сигналів різними інструментами.

Графіки перехідних процесів, отриманих за результатами симуляції досліджуваної моделі показані на рис. 15.17.

Графіки на рис. 15.16 відрізняються від графіків на рис. 13.5 тим, що перші з них отримані при розмиканні ключа та його замиканні, а другі – тільки при замиканні.



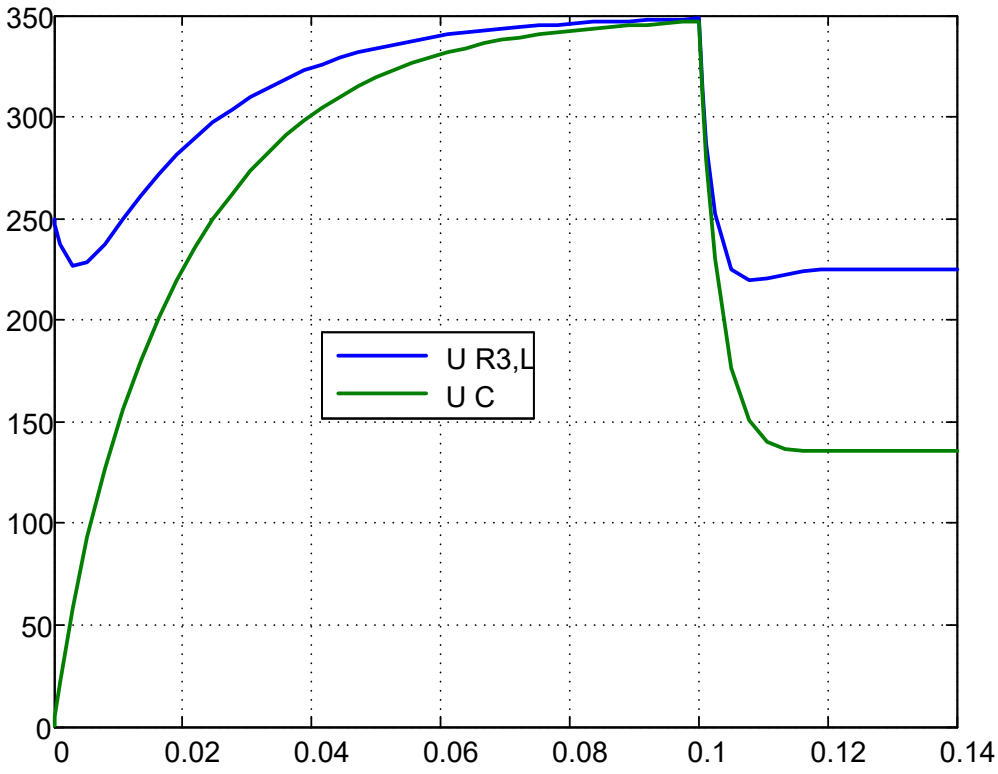


Рис. 15.17. Результати симуляції моделі рис. 15.16

15.4 Завдання

Виконати завдання 1 з лабораторної роботи №13 методом віртуального фізичного моделювання.

15.5 Методичні вказівки та рекомендації

1. Не забудьте витягнути у вікно моделі блок *powergui*.
2. Для того, щоб почати моделювання з нульовими початковими умовами, відкрийте вкладку *Initial States Setting* блоку *powergui* і установіть параметр *Force Initial Electrical States* у стан *To Zero*.
3. При застосуванні джерела змінного струму розмикайте електричне коло не блоком *Breaker*, а блоком *Ideal Switch*.

15.6 Контрольні питання та завдання

1. Чим відрізняються блоки бібліотек *SimPowerSystem* від *Simulink*-блоків?
2. Які порти мають *SPS*-блоки?
3. Як поєднати між собою *SPS*-блоки і блоки *Simulink*?
4. Для чого застосовують блок *powergui*?
5. Як працюватиме *SPS*-модель електричного кола без ключових елементів, якщо їй не задати нульові початкові умови?
6. Порівняйте між собою переваги та недоліки віртуального фізичного та структурного математичного моделювання.

Лабораторна робота №16

ВІРТУАЛЬНЕ ФІЗИЧНЕ МОДЕЛЮВАННЯ КОРОТКОЗАМКНЕНОГО АСИНХРОННОГО ДВИГУНА

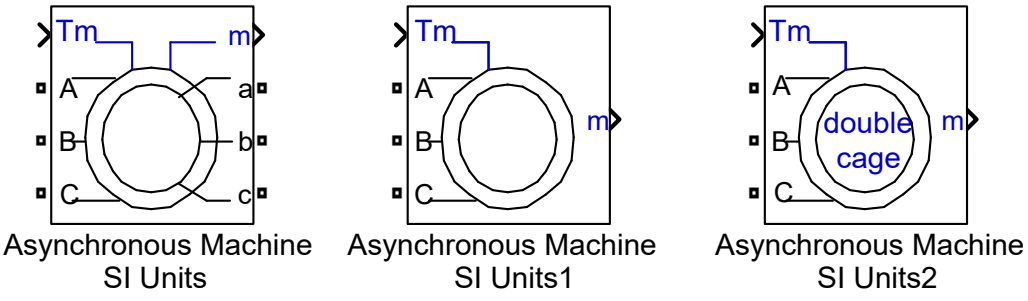
Мета роботи: придбати навички віртуального фізичного моделювання асинхронного двигуна в середовищі *MATLAB-Simulink-SimPowerSystem*

16.1 *SPS-блоки* моделей асинхронної машини

В розділі *Machines* бібліотеки *SimPowerSystems* представлені два блоки трифазної асинхронної машини: *Asynchronous Machine SI Units* (асинхронна машина в абсолютних одиницях системи СІ) та *Asynchronous Machine pu Units* (асинхронна машина у відносних одиницях).

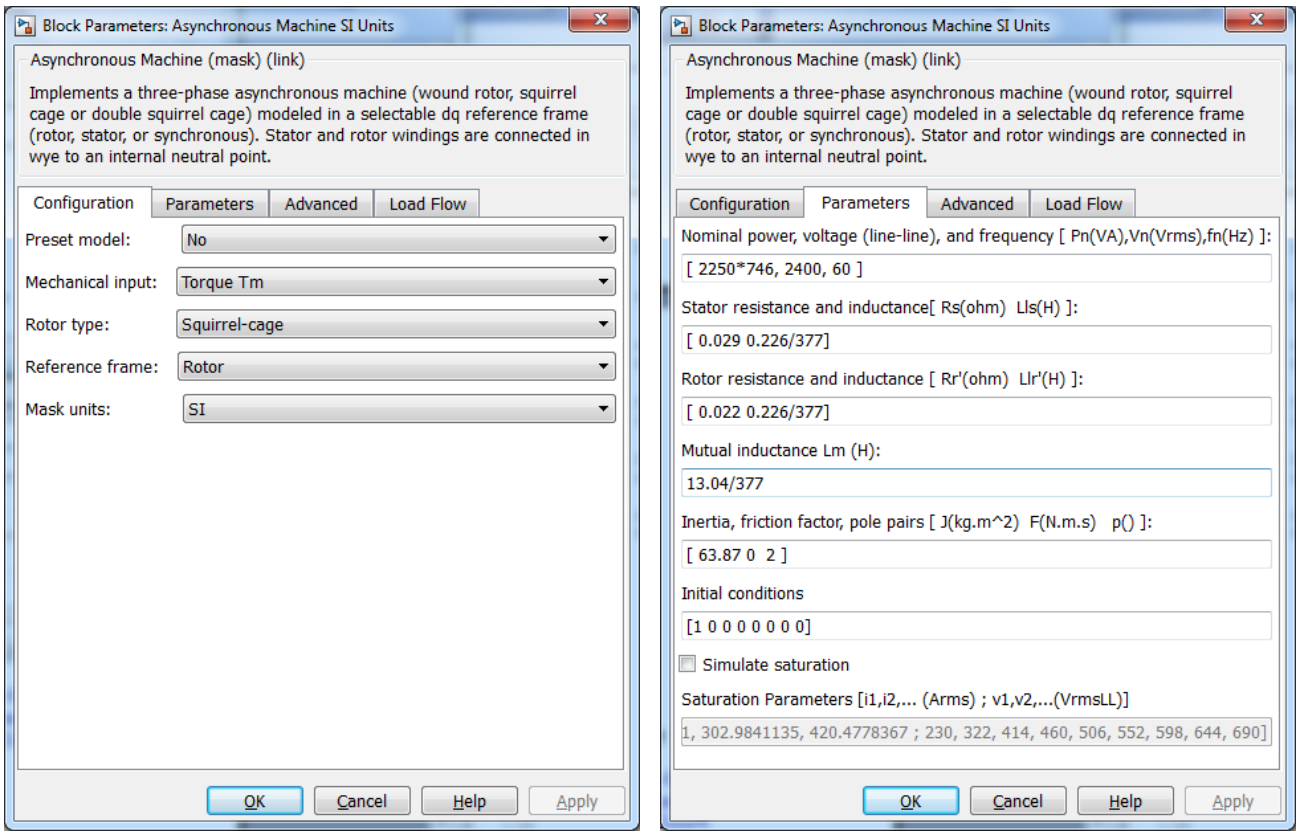
Блоки мають порти A, B, C та a, b, c, що відповідають «електричним» затискачам статора та ротора відповідно. Крім «електричних» портів, моделі мають один механічний вхід та векторний «інформаційний» вихід m (від *measurements*), який передбачає подальшу обробку або фіксацію сигналів засобами базових блоків *Simulink*.

Зовнішній вигляд блоків *Asynchronous Machine* (рис. 16.1) визначається типом ротора та типом механічного входу, які встановлюються за допомогою меню параметрів *Rotor type* і *Mechanical input* вкладки *Configuration* (рис. 16.2а).



а) *б)* *в)*

Рис. 16.1. Блоки *Asynchronous Machine* з різними типами ротора (*Rotor Type*): а – з фазним ротором (*Wound*), б – з білячою кліткою на роторі (*Squirrel-cage*), в –з двома білячими клітками на роторі (*Double squirrel-cage*)



а) *б)*

Рис. 16.2. Вкладки діалогового вікна встановлення параметрів блоку *Asynchronous Machine SI Units*

Вкладка *Parameters* (рис. 16.2*б*) діалогового вікна установки параметрів блока дозволяє встановити значення таких параметрів асинхронного двигуна(АД):

- *Nominal power* Pn (VA) – номінальна потужність P_n (ВА);
- *Voltage (line-line)* Vn (Vrms) – діюче, тобто ефективне, або середньоквадратичне (*rms – root mean square*) значення номінальної лінійної напруги статора U_{len} (В);
- *Frequency* fn (Hz) – номінальна частота напруги живлення f_n (Гц);
- *Stator resistance* Rs (Ohm) – активний опір фази статора, R_s (Ом);
- *Stator inductance* Lls – індуктивність розсіяння фази статора, $L_{s\sigma}$ (Гн);
- *Rotor resistance* Rr' (Ohm) – активний опір фази ротора, приведений до статора R_r (Ом);

- *Rotor inductance* Llr' – індуктивність розсіяння фази ротора, приведена до статора $L_{r\sigma}$ (Гн);
- *Mutual inductance* Lm (H) – головна взаємна індуктивність, L_m (Гн);
- *Inertia* J (kg.m^2) – момент інерції ротора, J (кг $\cdot$ м^2);
- *Friction factor* F (N.m.s) – коефіцієнт в'язкого тертя k_{fv} (Н $\cdot$ м/(рад/с));
- *Pole pairs* p () – кількість пар полюсів Z_p;
- *Initial conditions* – початкові умови за координатами: ковзання, кутове положення ротора, амплітуди та фазові кути струмів фаз статора та (якщо треба) ротора;
- *Simulate saturation* – при наявності в полі цього параметру прапорця буде враховуватись ефект насичення магнітного кола за табличними даними кривої намагнічування $U_s = f(I_s)$, що вводяться у вигляді дворядкової матриці параметру *Saturation Parameters;* перший стовпець цієї матриці отримує координати тієї точки кривої намагнічування, з якої починає проявлятися ефект насичення, тобто він повинен бути ненульовим.

Параметри АД та можливість їх встановлення або коригування залежать від стану опції *Preset Model* вкладки *Configuration* (рис. 16.2*а*).

У початковому стані ця функція має значення *No*, параметри вкладки *Parameters* мають певні значення, які після переміщення блоку у вікно моделі можна змінювати.

В меню функції *Preset Model* наводиться перелік, з якого можна обрати конкретний двигун за його потужністю у кінських силах HP (1HP = 746 Вт), діючим значенням лінійної напруги статора у Vrms, номінальною частотою в Hz та номінальною швидкістю в RPM (*Revolutions Per Minute* = об/хв). Вибір певного двигуна, підтверджений натисканням кнопки *Apply*, призводить до автоматичного встановлення відповідних параметрів у вкладці *Parameters*, які тепер не можуть бути скориговані користувачем.

Слід зазначити, що до параметрів попередньо обраних двигунів не

входять координати кривої намагнічування, тобто, ці координати задані тільки для одного двигуна, параметри якого встановлюються за замовченням.

Для того, щоб зробити дані попередньо обраного двигуна доступними до коригування, треба після вибору цього двигуна віртуальною кнопкою *Apply* знову встановити функцію *Preset Model* у значення *No*. Підкреслимо, що можливість ініціалізації моделі через встановлення певного набору параметрів АД передбачена тільки для двигунів з однією білячою кліткою на роторі.

Параметри вкладок *Reference frame* будуть пояснені у подальших дисциплінах, а параметри вкладок *Advanced* та *Load Flow* (рис. 5.3а) зазвичай можна не змінювати.

На рис. 16.3 зображена *SPS*-модель, призначена для дослідження прямого пуску АД [19].

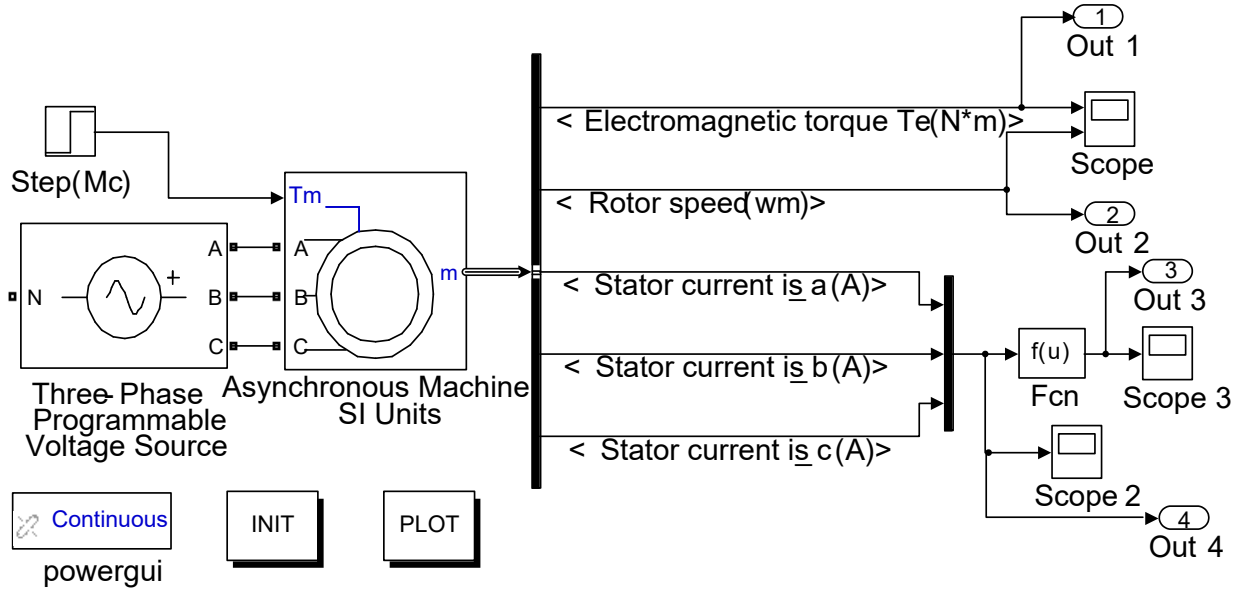


Рис. 16.3. *SPS*-модель прямого пуску асинхронного двигуна

У наведеній моделі вибір сигналів для реєстрації здійснюється з загального вектору вихідних *Simulink*-сигналів (порт m) *SPS*-блока *Asynchronous Machine SI Units S*-блоком *Bus Selector* бібліотеки *Signal Routing*. Вікно блоку *Bus Selector* показане на рис. 16.4.

Обчислення діючого (ефективного) значення струму статора виконує блок *Fcn* за формулою:

$$I_{se} = \sqrt{\left(I_A^2 + I_B^2 + I_C^2\right)/3}\,. \qquad (16.1)$$

Фазні напруги формуються джерелом *Tree-Phase Programmable Voltage Source*, вікно параметрів якого відображено на рис. 16.5.

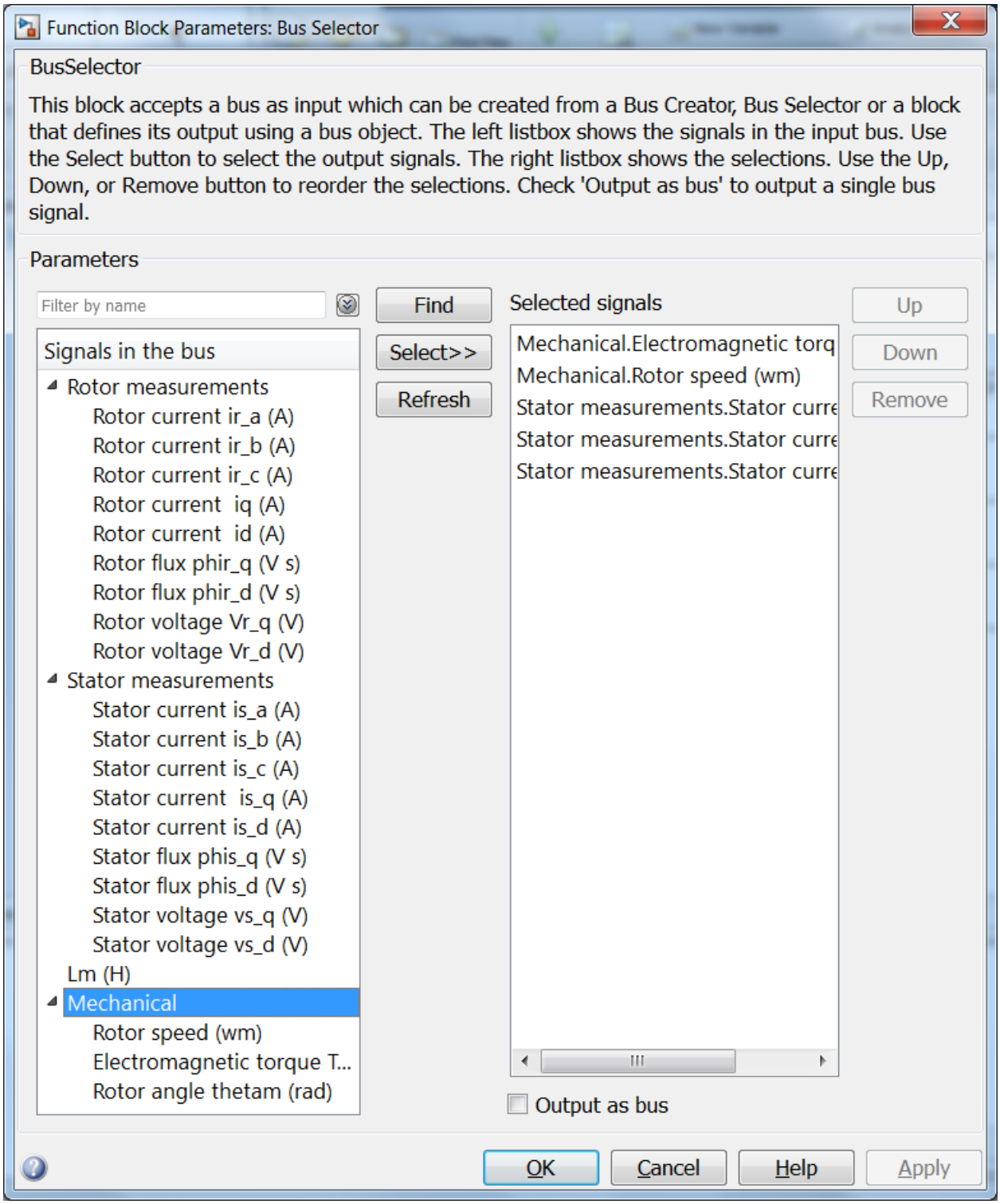


Рис. 16.4. Діалогове вікно блоку *Bus Selector,*
підключеного до вихідного інформаційного порту двигуна

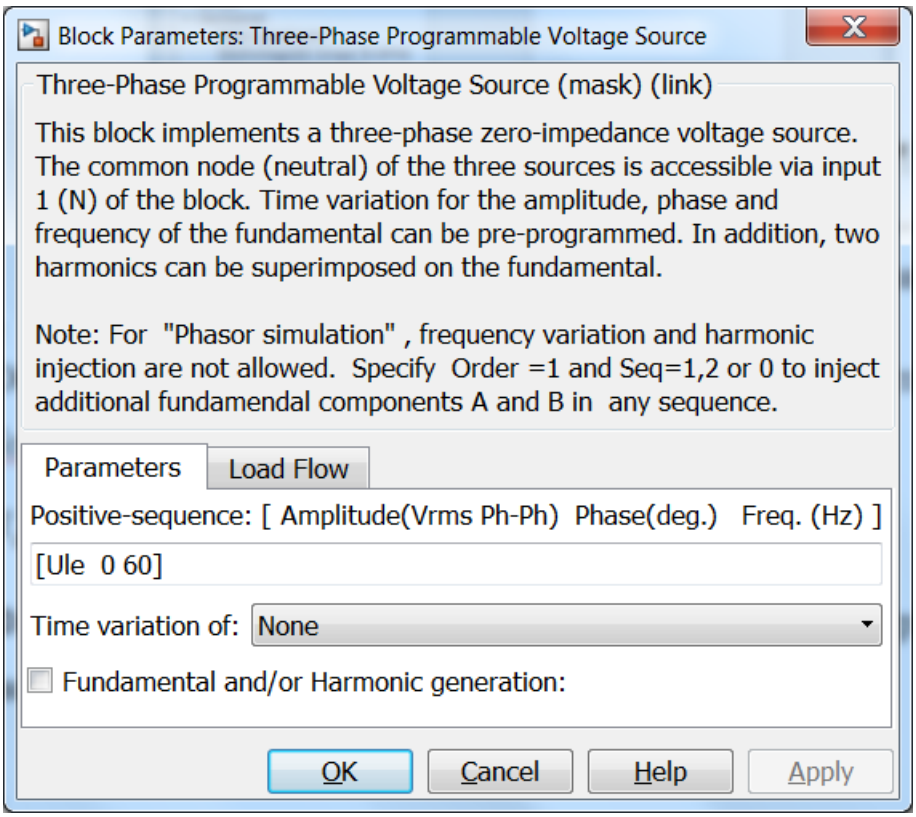


Рис. 16.5. Діалогове вікно блоку *Tree-Phase Programmable Voltage Source*

Замість одного джерела *Tree-Phase Programmable Voltage Source*, можна використати три блока *AC Voltage Source* (див. лабораторну роботу №11, пункт 1.2.1), з‘єднані зіркою.

При визначенні параметрів джерел необхідно звернути увагу на те, що в параметрах двигуна задано діюче значення лінійної напруги [Voltage (line-line) (Vrms)], а блоки *AC Voltage Source* потребують амплітудних значень фазних напруг [Peek amplitude (V)], а також, що SPS-джерела *AC Voltage Source* використовують частоту не в рад/с, а в герцах, а фазовий зсув синусоїдальних сигналів – не в радіанах, а в градусах [Phase (deg)].

Перехідні процеси, отримані за результатами симуляції моделі рис. 16.3 подані на рис. 16.6. За допомогою цієї ж моделі можна побудувати і статичні характеристики двигуна, якщо змінювати момент статичного опору за лінійним законом настільки повільно, щоб практично не призводити до збудження електромагнітного перехідного процесу. Завершення сеансу моделювання у цьому експерименті можна здійснити блоком *Stop Simulation*, який здійснює

зупинку моделювання тоді, коли швидкість АД після перекидання явно перейде в область від'ємних значень, що досягається застосуванням блоку *Relational Operator*.

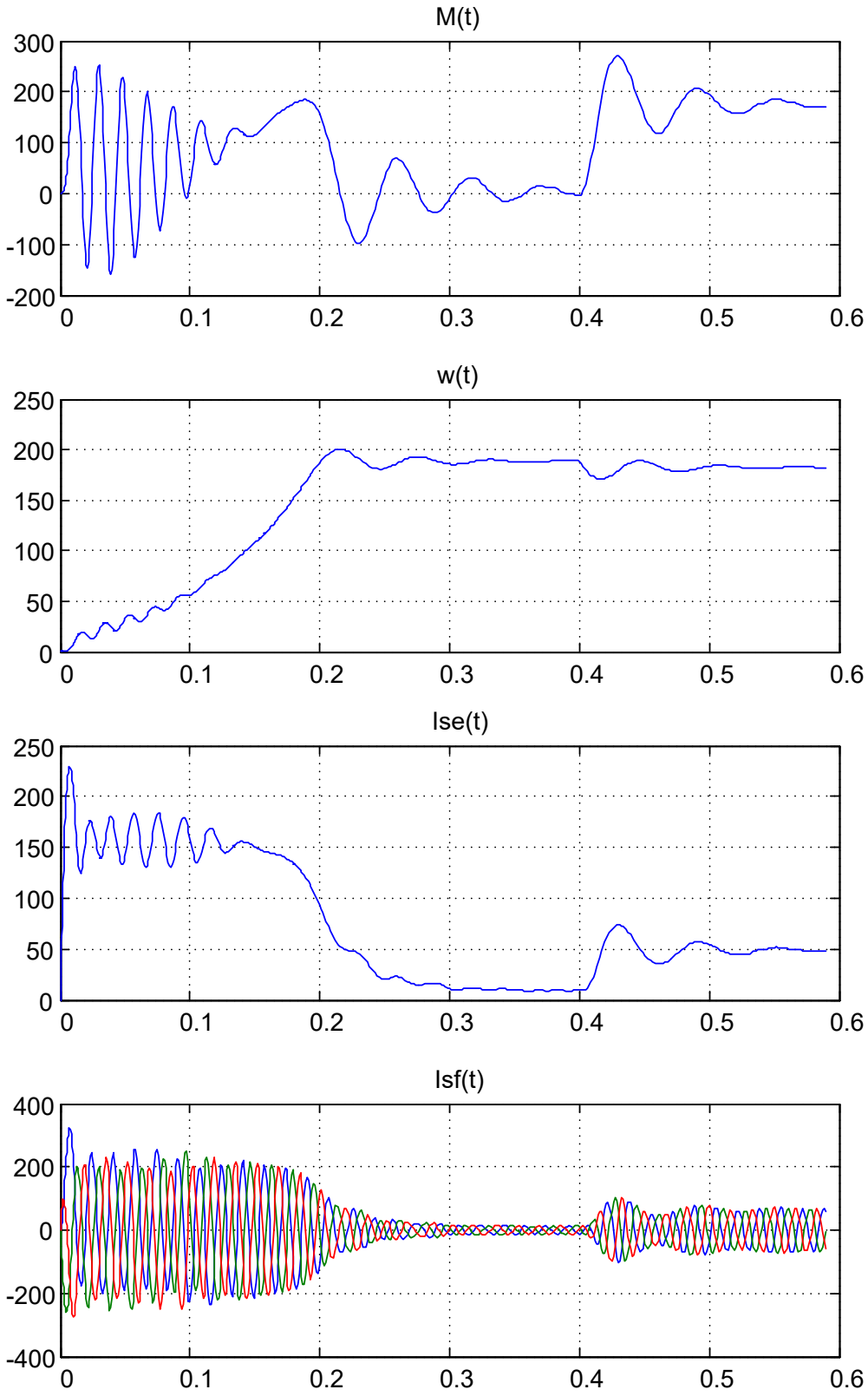


Рис. 16.6. Перехідні процеси при прямому пуску асинхронного двигуна

16.2 Завдання

Виконати дослідження процесу прямого пуску асинхронного двигуна методом віртуального фізичного моделювання за даними табл. 16.1, наведеними для АД з короткозамкненим ротором серії МТК, 380 В, 50 Гц.

Таблиця 16.1

№ вар.	Тип	$P_н$ кВт	$n_н$ об/хв	X_m Гн	Статор		Ротор		
					R_1 Ом	$X_{1\sigma}$ Ом	R_2 Ом	$X_{2\sigma}$ Ом	J кг*м^2
1	МТК011-6	1,4	870	54,0	5,98	3,93	8,41	3,80	0,02
2	МТК012-6	2,2	875	36,4	3,6	2,58	5,7	2,63	0,028
3	МТК-11-6	2,2	883	48,4	3,67	2,54	5,02	2,88	0,04
4	МТК-12-6	3,5	875	35,2	2,09	1,61	3,36	1,92	0,063
5	МТК111-6	3,5	870	29,8	2,16	2,03	3,33	1,46	0,046
6	МТК-21-6	5,0	910	24,5	1,11	1,07	1,86	1,5	0,098
7	МТК112-6	5,0	890	22,9	1,32	1,39	2,34	1,02	0,065
8	МТК-22-6	7,5	905	17,6	0,69	0,74	1,33	1,07	0,138
9	МТК-31-8	7,5	682	17,3	0,79	0,90	1,30	0,79	0,25
10	МТК-31-6	11,0	920	13,5	0,42	0,47	0,82	0,71	0,25
11	МТК-41-8	11,0	685	11,0	0,43	0,53	0,84	0,66	0,445
12	МТК-42-8	16,0	685	8,8	0,27	0,36	0,60	0,47	0,65
13	МТК-51-8	22,0	692	6,9	0,18	0,30	0,38	0,39	1,05
14	МТК-52-8	28,0	695	6,0	0,14	0,23	0,30	0,30	1,38

Виконайте такі віртуальні експерименти:

- прямий пуск двигуна від джерела, що виробляє трифазну синусоїдальну напругу, параметри якої відповідають номінальній напрузі та номінальній частоті статора; після досягнення двигуном синхронної швидкості здійсніть стрибкоподібний накид навантаження;

- поступово підвищуючи момент статичного опору на валу двигуна, знайдіть його критичний момент та критичне ковзання;
- виконайте попередні 2 експерименти зі зменшеною на 50% частотою джерела при номінальній амплітуді та зі зменшеною на 20% амплітудою напруги джерела при номінальній частоті;
- здійсніть пуск двигуна у протилежному напрямку.

Побудуйте статичну механічну та статичну швидкісну характеристики двигуна.

16.3 Методичні вказівки та рекомендації

1 Виведіть на екран додатково до графіків, зображених на рис. 16.6 струми ротора та потокозчеплення ротора і статора.

2 Для зміни напряму обертання двигуна поміняйте місцями 2 будь-які фази джерела. На ходу це можна зробити блоком *Selector*.

3 При побудові статичних характеристик відкиньте з векторів часу та вихідних величин значення, що передують зміні моменту статичного опору за допомогою логічної функції `find` та використання пустої матриці (`[]`).

16.4 Контрольні питання та завдання

1. Які параметри АД треба знати, щоб скористатися блоком *Asynchronous Machine SI Units*?
2. Як впливає на усталену швидкість зміна частоти напруги статора?
3. Як впливає на час розгону двигуна, просадку швидкості при накиді навантаження та критичний момент зміна амплітуди напруги статора?
4. Який із струмів підтримує АД у намагніченому стані: струм статора чи струм ротора?

ЛІТЕРАТУРА

1. Дьяконов В. П. Справочник по применению системы PC MatLAB. – М.: Наука Физматлит, 1993.
2. Курбатова Е.А. MATLAB 7. Самоучитель. – Издательство Вильямс, 2005. – 256 с.
3. Иглин С.П. Математические расчеты на базе MATLAB. – СПб.: БХВ-Петербург, 2005. – 640 с.
4. Лазарев Ю.Ф. МАТЛАБ 5.х. Серия “Библиотека студента” – К.: Издательская группа BHV, 2000. – 384с.
5. Кетков Ю.Л., Кетков А.Ю., Шульц М.М. MATLAB 7: программирование, численные методы. – СПб.: БХВ-Петербург, 2005. – 752 с.
6. Новгородцев А.Б. Расчет электрических цепей в MATLAB: Учебный курс. – СПб.: Питер, 2004. – 250 с.
7. Островерхов М.Я., Пижов В.М. Моделювання електромеханічних систем в Simulink: Навч. посібник для студентів вищих навчальних закладів. – К.: ВД «Стилос», 2008. – 528 с.
8. MATLAB 6/6.1/6.5 + Simulink 4/5. Основы применения / Дьяконов В.П. – М.: СОЛОН-Пресс, 2004. –768 с.
9. Дьяконов В.П. Simulink 4. Специальный справочник. – СПб.: Питер, 2001. – 528с.
10. Герман-Галкин С.Г. Компьютерное моделирование полупроводниковых систем в МАТЛАБ 6.0: Учебное пособие. – СПб.: КОРОНА принт, 2001. – 320 с.
11. Герман-Галкин С.Г. Линейные электрические цепи. Лабораторные работы на ПК. – СПб.: КОРОНА принт, 2002. – 192 с.
12. Черных И.В. Simulink: среда создания инженерных приложений. – Диалог-МИФИ, 2003. – 496 с.

ЗМІСТ