МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ
АВТОМОБІЛЬНО-ДОРОЖНІЙ УНІВЕРСИТЕТ

О.Г. Гурко, І.Ф. Єрьоменко

АНАЛІЗ І СИНТЕЗ СИСТЕМ АВТОМАТИЧНОГО КЕРУВАННЯ В MATLAB

Навчальний посібник

Затверджено методичною
радою університету,
протокол №4 від 16.02.2011 р.

Харків
ХНАДУ
2011

УДК 681.51.011
ББК 32.973
Г 95

Рецензенти:
О.Г. Руденко, д-р. техн. наук, професор Харківського національного університету радіоелектроніки;

В.Г. Ягуп, д-р. техн. наук, професор Харківської національної академії міського господарства;

В.Д. Сахацький, д-р. техн. наук, професор Української інженерно-педагогічної академії.

Гурко О.Г., Єрьоменко І.Ф.
Г 95 Аналіз та синтез систем автоматичного керування в MATLAB. Навчальний посібник/ **О.Г. Гурко, І.Ф.Єрьоменко.** – Харків: ХНАДУ, 2011. - 286 с.

ISBN

Посібник містить відомості по використанню однієї з найефективніших систем комп'ютерної математики MATLAB для аналізу і синтезу систем автоматичного керування. Матеріал ґрунтується на MATLAB версії 6.5. Посібник розділений на 3 частині. У першій частині розглянуті основні прийоми роботи в MATLAB; друга та третя присвячені можливостям додатків Control System Toolbox та Simulink для моделювання систем керування.

Книга призначена для студентів, аспірантів і слухачів факультету підвищення кваліфікації, що вивчають курси «Теорія автоматичного керування», «Моделювання систем керування», а також для викладачів, інженерів і науковців, що цікавляться питаннями автоматичного керування.

Іл. 246. Табл. 5. Бібліогр. 21 назв.

Пособие содержит сведения по использованию одной из самых эффективных систем компьютерной математики MATLAB для анализа и синтеза систем автоматического управления. Материал основывается на MATLAB версии 6.5. Пособие разделено на 3 части. В первой части рассмотрены основные приемы работы в MATLAB; вторая и третья посвящены возможностям приложений Control System Toolbox и Simulink для моделирования систем управления.

Книга предназначена для студентов, аспирантов и слушателей факультета повышения квалификации, изучающих курсы «Теория автоматического управления», «Моделирование систем управления», а также для преподавателей, инженеров и научных работников, интересующихся вопросами анализа и синтеза систем управления.

Ил. 246. Табл. 5. Библиогр. 21 наим.

УДК 681.51.011
ББК 32.973

ISBN

ЗМІСТ

ВСТУП

Проектування сучасних систем керування неможливо без використання математичне моделювання, що пояснюється, насамперед, складністю промислових об'єктів. При цьому широке поширення придбали математичні пакети, серед яких одним з найвідоміших і пристосованих для інженерних потреб є середовище MATLAB. До складу MATLAB входять кілька пакетів розширення (Toolboxes), що призначені для рішення завдань керування та імітаційного моделювання, зокрема Control System Toolbox і Simulink.

Control System Toolbox працює з безперервними та дискретними моделями систем у вигляді передаточних функцій, а також з моделями у просторі станів і дозволяє вирішувати як класичні завдання, так і завдання сучасної теорії керування: розраховувати динамічні та частотні характеристики; проектувати регулятори частотними та кореневими методами; синтезувати оптимальні системи керування; вирішувати стаціонарні рівняння Ляпунова, Риккаті та ін.

Потужним засобом дослідження та синтезу динамічних систем є середовище візуального моделювання Simulink.

Для найбільш ефективного використання MATLAB при аналізі та синтезі систем автоматичного керування необхідно вміти працювати з усіма спеціалізованими пакетами розширення, які вигідно доповнюють один одного. Зрозуміло, що використання цих пакетів неможливо без володіння навичками роботи в MATLAB.

На жаль, при всьому великому обсязі видань, присвячених MATLAB, література, яка написана зрозумілою інженерною мовою і присвячена вживанню MATLAB для дослідження та проектування систем автоматичного керування, в нашій країні практично відсутня.

Даний посібник призваний допомогти придбати навички використання MATLAB 6.5 для аналізу і синтезу систем автоматичного керування.

ЧАСТИНА І
ОСНОВИ MATLAB

1. ПОЧАТОК РОБОТИ З MATLAB

1.1 Загальні положення

У наші дні комп'ютерна математика одержала належне поширення та інтенсивно розвивається як науковий напрямок на стику математики та інформатики. Серед ряду сучасних засобів комп'ютерної математики (таких, як Mathcad, Maple, Mathematica) вигідно виділяється MATLAB - одна з найстарших (перша версія вийшла в 1984 р.), ретельно пророблених і перевірених часом систем автоматизації математичних розрахунків, побудована на розширеному застосуванні матричних операцій. Це знайшло відбиття в назві системи: MATrix LABoratory - матрична лабораторія.

MATLAB - це система програмування, об’єктно орієнтована на математичні розрахунки та переважає по зручності в цій області кожну з універсальних мов Fortran, Pascal або C(C++) і т.п.

Система MATLAB пропонується розроблювачами (MathWorks, Inc.) як мова програмування високого рівня для технічних обчислень (ядро), яка розширюється великим пакетом прикладних програм (toolboxes), що складаються з M- файлів, які вирішують спеціалізовані задачі.

Познайомимося з основними об'єктами мови програмування MATLAB.

1.2 Інструкції та змінні

Після запуску MATLAB на екрані з'являється вікно, основна область якого призначена для введення команд і виводу результатів. Ця область так і називається *командним вікном* – **Command Window**.

У загальному випадку програма рішення деякої задачі являє собою послідовність інструкцій виду:

```
>> змінна=вираження
```

Таким чином, MATLAB дозволяє працювати в режимі звичайного командного рядка. Командний рядок в MATLAB

позначається двома спрямованими вправо стрілками « >> ». Після заповнення командного рядка і натискання клавіші <Enter> відкривається наступний командний рядок.

Центральним поняттям всіх математичних систем є «математичне вираження», яке задає те, що повинно бути обчислено. Математичні вираження будуються на основі чисел, змінних, констант, операторів і функцій.

При роботі з MATLAB у вираженнях використовуються звичайні символи математичних операцій, наведені в табл. 1.1, а у якості оператору присвоювання використовується звичний знак рівності «=». При обчисленні виражень операції зведення в ступінь, множення та ділення мають пріоритет перед операціями додавання та віднімання. Порядок виконання арифметичних операцій можна змінити за допомогою круглих дужок.

Таблиця 1.1

Символи математичних операцій

+	Додавання
-	Віднімання
*	Множення
/	Ділення
^	Зведення в ступінь

Розглянемо приклади.

Приклад 1.

```
>> x=1+2  ←————————  Натиснути < Enter >

x =

      3

>>   (MATLAB готовий до наступної інструкції)
```

Обчислення виконується після натискання клавіші <Enter>. Після цього значення `x` автоматично відображається на екрані.

Приклад 2.

```
>> x=1+2*3  ←————————  Натиснути < Enter >
```

```
x =

      7

>>
```

Приклад 3.

```
>> x=(1+2)*3

x =

      9

>>
```

Якщо після інструкції стоїть крапка з комою (**;**), то вивід значення змінної `x` на екран подавлюється. Це зручно, коли виконуються якісь проміжні обчислення, вивід результатів яких на екран не представляє інтересу. Довідатися про значення змінної можна, ввівши її ім'я:

```
>> x          <----------  Натиснути < Enter >

      9
>>
```

Своє ім'я повинна мати кожна змінна, що задається або повинна обчислюватися. Якщо користувач із якої-небудь причини не задав імені змінної, то їй автоматично привласнюється ім'я `ans`:

```
>> 2+2

ans =

     4
>>
```

Ім'я змінної може містити скільки завгодно символів, але запам'ятовується тільки 31 початковий символ. Ім'я повинне починатися з букви, може містити букви латинського алфавіту, цифри і символ підкреслення, реалізований клавішами <shift> + <->. Неприпустимо включати в імена змінних, пробіли та спеціальні

знаки, наприклад «+», оскільки в цьому випадку інтерпретація виражень стає неможливою.

Бажано використовувати змістовні імена для позначення змінних, наприклад, змінної, що позначає силу струму, можна дати ім'я `silastrumy`.

MATLAB розрізняє прописні і малі літери, тому імена `silastrumy` та `Silastrumy` є різними.

В MATLAB є кілька констант із заздалегідь закріпленими за ними іменами. Це `pi`, `inf`, `Na`, `i` та `j`.

`Na` (скорочення від *Not-a-Number*) використовується для позначення невизначеного (нечислового) результату операції:

```
>> z=0/0
Warning: Divide by zero.
(Попередження: Ділення на нуль)
(Type "warning off MATLAB:divideByZero" to suppress this warning.)

z =

  Na

>>
```

`Inf` відповідає $+\infty$, а `pi` - числу π. Змінні `i` та `j` позначають мниму одиницю і використовуються при арифметичних операціях з комплексними числами. У принципі, цим іменам можна привласнювати й інші значення, однак, щоб уникнути непорозумінь, настійно рекомендується не використовувати їх без особливої потреби.

Функції - це об'єкти, що мають унікальні імена, виконують певні перетворення своїх аргументів і при цьому повертають результати цих перетворень. Результат обчислення функції з одним вихідним параметром підставляється на місце її виклику, що дозволяє безпосередньо використовувати функції в математичних вираженнях. Наприклад:

```
>>X=2*sin(pi/3)
```

Функції в загальному випадку мають список аргументів (параметрів), обмежений круглими дужками. Якщо функція повертає кілька значень, то вона записується у вигляді:

```
[Y1,Y2,…]=func(x1, x2,…),
```

де `Y1, Y2,…` - список вихідних параметрів;

`x1, x2,…` - список вхідних аргументів (параметрів).

Із списком елементарних функцій можна ознайомитися в довідковій системі MATLAB, вибравши меню **Help** на панелі інструментів або набравши в командному рядку команду `help elfun`, а із списком спеціальних функцій – за допомогою команди `help specfun`. Функції можуть бути вбудованими (внутрішніми) і зовнішніми (M- функціями). MATLAB містить багато зовнішніх (спеціалізованих) функцій - m-файлів. Крім того, користувач має можливість розробляти свої зовнішні функції за допомогою вбудованого редактора m-файлів. Список найпоширеніших вбудованих функцій наведений у табл. 1.2.

Таблиця 1.2

Деякі функції MATLAB

sin(x)	Синус	log2(x)	Логарифм по підставі 2
asin(x)	Арксинус	sqrt(x)	Квадратний корінь
cos(x)	Косинус	abs(x)	Абсолютне значення числа
acos(x)	Арккосинус	conj(x)	Комплексно - сполучене число
tan(x)	Тангенс	real(x)	Дійсна частина комплексного числа
atan(x)	Арктангенс	imag(x)	Мнима частина комплексного числа
cot(x)	Котангенс	fin(x)	Округлення убік нуля
acot(x)	Арккотангенс	floor(x)	Округлення убік -∞
exp(x)	Експонентна функція	ceil(x)	Округлення убік +∞
log(x)	Натуральний логарифм	factorial(x)	Факторіал
log10(x)	Десятковий логарифм	sgn(x)	Сигнум (виділення знака)

1.3 Формати виводу чисел

За замовчуванням MATLAB видає числові результати в наступних двох форматах:

1) якщо результат дробове число, то воно виводиться в нормалізованій формі із чотирма цифрами після десятинної крапки;

2) якщо результат ціле число, то воно виводиться на екран без десятинної крапки та наступних нулів.

Багатьох користувачів така форма подання результатів не завжди влаштовує. Тому при роботі із числовими даними MATLAB дозволяє задавати інші формати подання чисел. Для цього використовується команда:

```
>> format тип формату; ім'я змінної;
```

Нижче зазначені основні типи форматів.

- `format` або `format short` – коротке подання із чотирма знаками після десятинної крапки, наприклад

```
>> x=2.1;
>> y=x;
>> format short;y

y =

  2.1000
```

- `short e` – коротке подання в експонентному форматі (1 цифра до десятинної крапки, 4 цифри після десятинної цифри і 3 знаки порядку):

```
>> x=20.2;
>> y=x;
>> format short e; y

y =

2.0200e+001
```

- `long` – довге подання у фіксованому форматі з 14 знаками після десятинної крапки:

```
>> x=20.2;
>> y=x;
>> format long; y

y =

20.20000000000000
```

- `long e` – довге подання в експонентному форматі (1 цифра до десятинної крапки, 14 цифр після десятинної крапки та 3 знаки порядку):

```
>> x=20.2;
>> y=x;
>> format long e;y

y =

2.020000000000000e+001
```

Завдання формату позначається тільки на формі виводу чисел. Обчислення однаково відбуваються у форматі подвійної точності, а введення чисел можливе в будь-якому зручному для користувача виді.

1.4 Редагування програми

У деяких випадках математичне вираження, що вводиться, може бути настільки довгим, що для нього не вистачить видимої частини вікна. Це незручно. У цьому випадку частину вираження можна перенести на новий рядок за допомогою знака крапок <....> (4 або більше крапок). Наприклад,

```
>> s=1-1/2 + 1/3 - 1/4 + 1/5 - 1/6 +....<Enter>
1/7-1/8 + 1/9 - 1/10 + 1/11 - 1/12

s =

0.6532
```

При наборі програми використовуються команди рядкового редактора, наведені в табл. 1.3.

Для тих, хто працював з універсальними мовами, у табл. 1.3 на перший погляд нічого нового немає. Однак використання клавіш <↑> та <↓> має свою специфіку. Якщо в універсальних мовах використання цих клавіш приводить до переміщення курсору по тексту програми, то в MATLAB навпаки - до переміщення раніше введених рядків у командний рядок для виправлення або дублювання.

Таблиця 1.3

Команди рядкового редактора MATLAB

Комбінація клавіш	Призначення
→ або Ctrl + b	Переміщення курсору вправо на один символ
← або Ctrl + f	Переміщення курсору вліво на один символ
Ctrl + → або Ctrl + r	Переміщення курсору вправо на одне слово
Ctrl + ← або Ctrl + *l*	Переміщення курсору вліво на одне слово
Home або Ctrl + a	Переміщення курсору в початок рядка
End або Ctrl + e	Переміщення курсору в кінець рядка
↑ та ↓ або Ctrl + p та Ctrl + n	Перегортання попередніх команд угору або вниз для підстановки в рядок введення
PgUp	Перегортання сторінок сесії вверх
PgUn	Перегортання сторінок сесії вниз
Del або Ctrl + d	Стирання символу праворуч від курсору
← або Ctrl + h	Стирання символу ліворуч від курсору
Ctrl + k	Стирання до кінця рядка
Esc	Очищення рядка введення
Ins	Включення/виключення режиму вставки

Для цих же цілей можна використовувати вікно **Command History**, що за замовчуванням розташовано в лівому нижньому куті вікна MATLAB. У цьому випадку необхідну команду досить

перетягнути мишкою в командний рядок (цей спосіб більш докладно буде розглянутий у главі 2). Проте, для редагування невеликих програм застосування клавіш <↑> і <↓> дуже зручно.

Розглянемо приклад. Дослідник розробляє програму обчислення функції

$$W(x,y)=\frac{x^2-3y+\sin(x+y)}{x+y+\cos(x)}$$

при довільно заданих значеннях змінних *x* та *y*. Набрано програму:

```
>> x=2;
>> y=3;
>> w1=x^2-3*y+sin(x*y);
>> w2=x+y+cos(x);
>> w=w1/w2
```

Перед тим, як натиснути клавішу <Enter> дослідник помітив, що в 3-ому рядку припустився помилки, а саме: замість `sin(x+y)` набрана функція `sin(x*y)`.

Порядок виправлення помилки полягає в наступному.

1. Необхідно закрити останній рядок крапкою з комою (`;`), щоб не виводити на екран невірний результат.

2. Натиснути клавішу <Enter>, при цьому відкриється новий командний рядок:

```
>> w=w1/w2;
>>
```

3. Натискати клавішу <↑> доти, поки в командний рядок не потрапить рядок, у якому допущена помилка:

```
>> w1=x^2-3*y+sin(x*y);
```

4. Виправити помилку. Натиснути клавішу <Enter>. На екрані буде спостерігатися програма:

```
>> x=2;
>> y=3;
>> w1=x^2-3*y+sin(x*y);   ←——  невірне вираження
>> w2=x+y+cos(x);
```

```
>> w=w1/w2;
>> w1=x^2-3*y+sin(x+y); ←——  вірне вираження
>>
```

5. Перерахувати значення змінної w при новому значенні w1. Для цього можна двічі натиснути клавішу <↑>:

```
>> w1=x^2-3*y+sin(x+y);
>> w=w1/w2;
```

6. Видалити крапку з комою і натиснути клавішу <Enter>. Оскільки рядок не закритий знаком (;), на екрані з'явиться результат:

```
w =

   -1.3000
```

Розглянутий приклад показує, що працювати з великими програмами в режимі командного рядка досить незручно. Проте, MATLAB - це наймогутніший пакет для рішення складних інженерних і наукових задач. Отже, існують більш зручні способи написання і редагування програм. Це питання, а також деякі інші можливості MATLAB будуть розглянуті в наступних главах.

Завдання для самостійної роботи

1. За допомогою MATLAB знайдіть значення наступних виражень

а) $e^{2,45}$; *в*) $\sqrt[3]{5^3+17^4}$; *д*) $\log_2 4+\sqrt{-4}$; *ж*) $\lg_{10}(-150)$;

б) $e^{-1,18}$; *г*) $\left|\lg 0,1\right|$; *е*) $\ln 2,72$; *з*) $3+2\cos\pi$.

2. Представте результати обчислень виражень *а, д* і *ж* у фіксованому форматі з 14 знаками після десятинної крапки.

3. Представте результати обчислень виражень *б, е* і *з* у довгому експонентному форматі.

4. Виділіть окремо дійсну та мниму частини з результатів обчислень виражень *д* і *ж* першого завдання.

2. ОПЕРАЦІЇ З ТЕКСТОМ СЕСІЇ І РОБОЧОЮ ОБЛАСТЮ

2.1 Ще раз про роботу з командним рядком

Сеанс роботи з MATLAB прийнято називати *сесією* (session). Сесія відображує всю роботу користувача по складанню програми. Наприклад. Користувач хоче скласти програму обчислення суми ($\sin^2 x+\cos^2 y$) і суми ($\sin^3 x+y$) при довільних x и y.

```
>>% Відкрити щоденник сесії; zses - це ім'я
>>% файлу, у якому буде зберігатися
>>% текст сесії.
>> diary('zses')
>>% Користувач визначає
>>% конкретні значення x и y.
>> x=3;
>> y=4;
>>% Користувач вводить функцію z
>>% двох аргументів x и y, яка
>>% при довільних значеннях x и y
>>% обчислить суму sin^2x+cos^2y
>> z=inline('sin(x)^2+cos(x)^2');
>>% Користувач помилився і ввів
>>% замість cos(y) cos(x).
>>% Вивід результату на екран.
>> R=z(x,y)
??? Error using ==> inline/subsref
Too many inputs to inline function
```

MATLAB виводить повідомлення про помилку: «Занадто багато входів у функцію `inline`». Дійсно, при визначенні функції *z* не використаний аргумент *y*, хоча функція *z* була оголошена як функція двох аргументів.

Дослідник виправляє програму, як описано в п. 1.4:

- двічі натиснувши клавішу <↑> викликає рядок з помилкою

```
>>z= inline ('sin(x)^2+cos(x)^2');
```

- виправляє помилку

```
>> z=inline('sin(x)^2+cos(y)^2');
```

- перераховує значення функції *R* і виводить результат на екран

```
>>R=z(x,y)          <Enter>
R=
 0.4472
```

Вся проведена робота над складанням програми – це і є сесія. Програма ще не закінчена, оскільки не визначена функція, що обчислює ($\sin^3 x+y$). Але вже містить так зване «сміття» - неправильне визначення функції *z* і коментарі MATLAB про виявлену помилку.

Не маючи у своєму розпорядженні час для продовження розробки програми, дослідник хоче зберегти розроблену частину. Прогнозуючи саме таку ситуацію, він першою інструкцією використав команду `diary('zses')` – ведення щоденника. Завдяки цьому в пам'яті MATLAB буде зберігатися файл `zses` з повним текстом сесії (тобто, з усім «сміттям»).

Починаючи роботу із завершення програми, користувач у новій сесії першої використає наступну інструкцію:

```
>>type zses
```

У командному вікні з'явиться текст попередньої сесії. Але текст - неробочий, без позначень «>>» командних рядків. Тому користувачеві прийдеться, користуючись клавішами <↑> і <↓> створити робочий варіант програми, попутно позбуваючись від «сміття». Перед формуванням робочого варіанта має сенс знову використати запис у щоденник, але дати файлу інше ім'я:

```
>>Попередня неробоча сесія із «сміттям»
……..……
>>diary ('przses')
>>Робочий текст програми.
```

Тепер файл `zses` можна видалити з пам'яті MATLAB.

Існує й інша, більш зручна можливість відновлення тексту попередньої сесії.

2.2 Вікно Command History

В MATLAB є область **Command History** (Історія команд), у якій зберігаються в неробочій формі тексти попередніх сесій. Якщо між сесіями ніхто цю область не очищав то, користуючись нею, можна відновити робочий варіант програми. За замовчуванням вікно **Command History** розташоване в лівому нижньому куті вікна MATLAB. Якщо його немає, то необхідно щигликом лівої клавіші миші відкрити меню **View** і потім поставити прапорець напроти позиції **Command History**. Тепер на екрані будуть розташовуватися два вікна: **Command Window** і **Command History** (рис. 2.1).

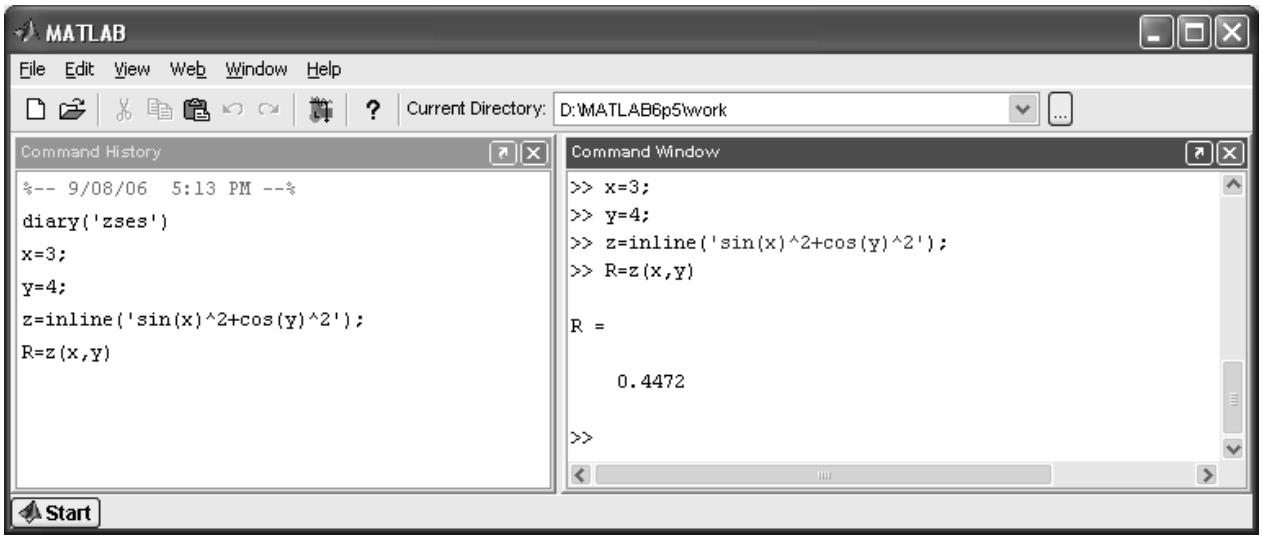


Рис. 2.1. Вікна **Command History** (ліворуч) і **Command Window** (праворуч)

Після відкриття нового командного рядка щигликом лівої клавіші миші треба виділити необхідний рядок в **Command History** і перетягнути його мишею в командний рядок, розташовуючи праворуч від «>>» і сполучаючи ліву границю рядка, що перетаскується, з курсором у вигляді вертикальної лінії. Потім треба натиснути клавішу <Enter>. Відкриється новий командний рядок; і т.д. до кінця програми.

Новий файл не буде містити «сміття», буде перебувати в робочому стані й буде «вічно» зберігатися в неробочому стані в пам'яті MATLAB під ім'ям, визначеним на початку сесії.

2.3 Вікна Workspace і Array Editor

Перейдемо тепер до розгляду операції з **Workspace** – робочою областю. У робочій області зберігаються імена змінних і функцій, заданих у командному вікні, а також їхні характеристики (рис. 2.2).

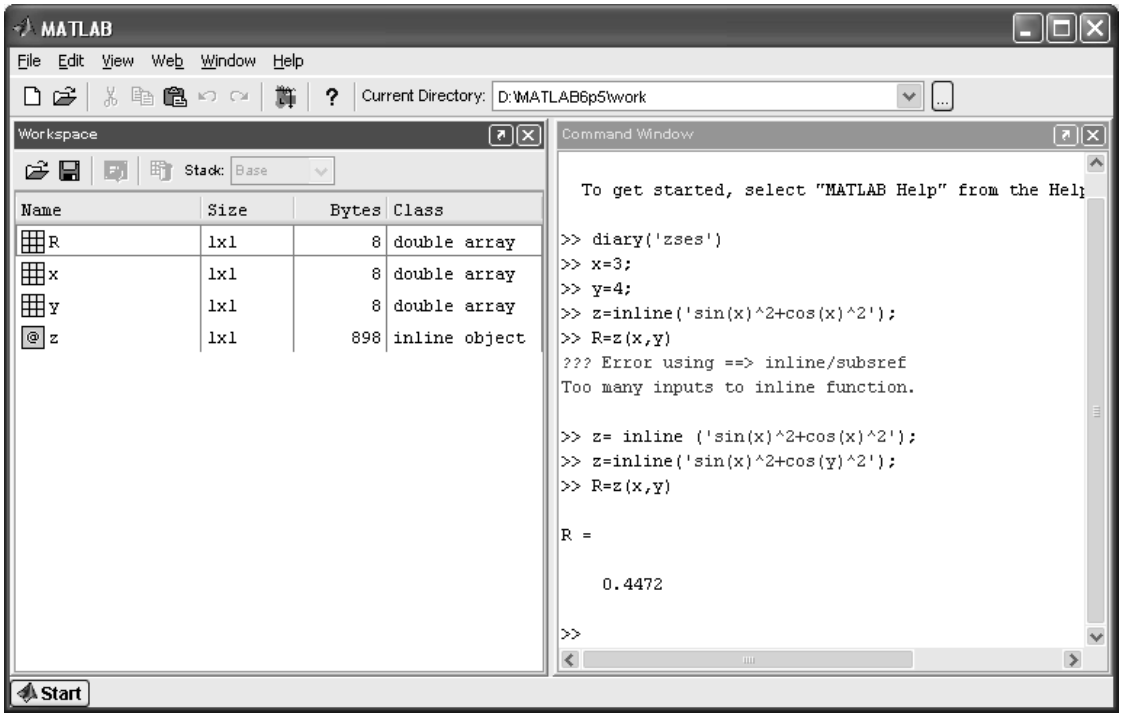


Рис. 2.2. Вікно **Workspace** (ліворуч) і **Command Window** (праворуч)

За замовчуванням вікно **Workspace** розташовано в лівому верхньому куті вікна MATLAB. У разі відсутності, викликати його можна, поставивши прапорець напроти відповідної позиції в меню **View**. До речі, вибравши **View→Desktop Layout→Default** можна задати вид вікна MATLAB, прийнятий за замовчуванням (рис. 2.3), з вікнами **Command Window**, **Command History**, **Workspace** і закладкою **Current Directory** (Поточна директорія).

Переглянути значення змінної або вид функції можна у вікні редактора **Array Editor**. Для цього у вікні **Workspace** необхідно двічі клацнути лівою кнопкою миші на необхідній змінній або функції.

У вікні **Array Editor** можна привласнити змінній будь-яке значення в будь-якому форматі. Якщо число в комірці не міститься, комірку можна розсунути за допомогою миші, помістивши стрілку над її правою границею (рис. 2.4).

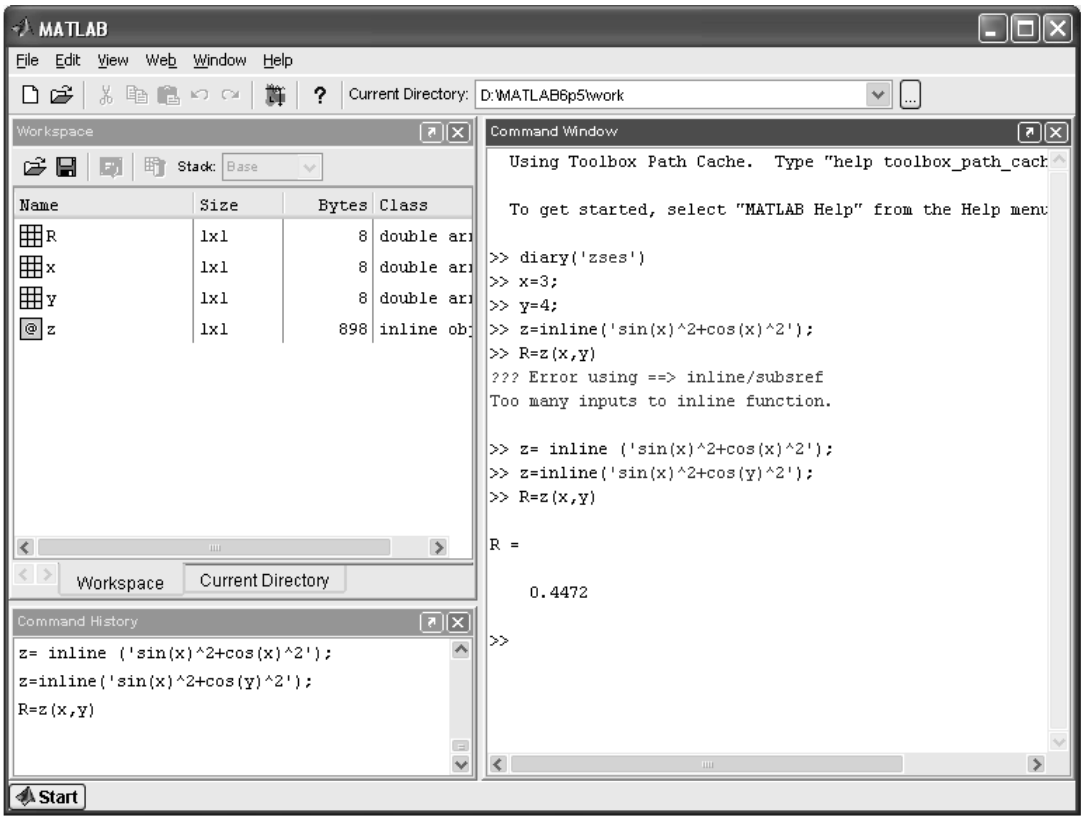


Рис. 2.3. Вікно MATLAB, прийняте за замовчуванням

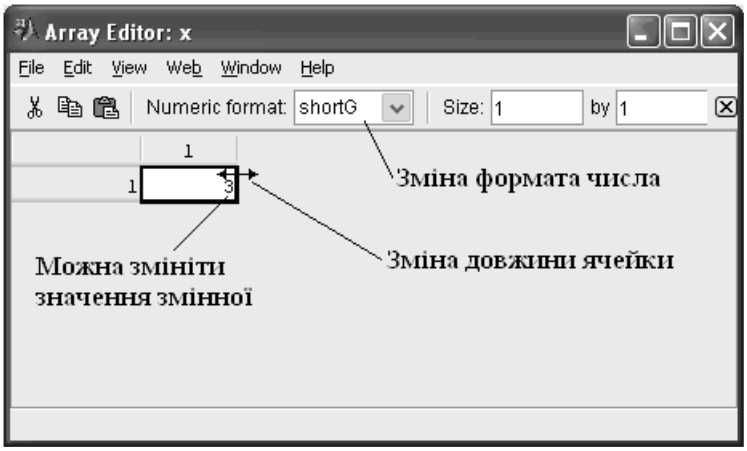


Рис. 2.4. Вікно **Array Editor**

При подвійному щиглику мишею на піктограмі функції *z* в **Workspace** у вікні **Array Editor** з'явиться визначення функції *z*(*x*,*y*):

```
inline function:
z(x,y)= sin(x)^2+cos(y)^2
```

MATLAB дозволяє зберігати зміст **Workspace** у вигляді файлів з розширенням .mat. Для цього в меню **File** необхідно

вибрати команду **Save Workspace As**. При цьому відкриється вікно **Save to MAT-File**, у якому можна вказати ім'я файлу і місце його розташування (за замовчуванням це каталог work).

Для завантаження раніше проведеної сесії (яка вже має ім'я) у робочу область використовується команда:

```
>>load (ім'я файлу.mat)
```

Під завантаженням (англ. *load*) розуміється залучення файлу в поточну сесію, відкриття можливості використання його даних.

Змінні програми, що розробляється, автоматично запам'ятовуються в робочій області та можуть бути використані протягом усього сеансу роботи в MATLAB. За допомогою функції `who` (всі букви рядкові!) можна вивести на екран список всіх змінних, що зберігаються в робочій області. Функція `whos` виводить на екран список змінних у робочій області разом з додатковою інформацією про їх тип, розмірність і пам'ять, що вони займають. Так, якщо в робочій області зберігаються тільки змінні з розглянутого раніше прикладу, то використання команд `who` і `whos` дасть наступні результати:

```
>> who

Your variables are:

R x y z

>> whos
  Name  Size            Bytes  Class

  R     1x1                 8  double array
  x     1x1                 8  double array
  y     1x1                 8  double array
  z     1x1               898  inline object

Grand total is 77 elements using 922 byte
```

Нагадаємо, що цю же інформацію можна побачити у вікні робочої області **Workspace**.

Змінні можна видалити з робочої області за допомогою функції `clear`. Сама функція `clear` видаляє з робочої області всі

дані (змінні та функції); `clear variables` видаляє всі змінні; `clear name1 name2…` видаляє змінні `name1 name2` і т.д. Наприклад:

```
>> clear x
>> who

Your variables are:

y  R
```

Змінна *x* вилучена з робочої області. Залишилися тільки *y* та *R*.

Завдання для самостійної роботи

1. Потренуйтеся в зміні розміщення вікон MATLAB.

а) Відокремте **Command Window** від інших вікон.

б) Розташуєте по горизонталі поруч вікна **Command History** та **Command Window.**

в) Розташуєте вікна в порядку, прийнятому в MATLAB за замовчанням.

г) Розташуєте вікна так, як зручно Вам.

2. *а*) Створіть в командному вікні програму, що обчислює значення функції

$$f = x + 3y - \sqrt{2x + 12z}$$

при довільних значеннях *x*, *y* та *z*.

б) Збережіть сесію під будь-яким іменем та очистить робоче вікно.

в) Завантажте збережену раніше сесію і визначите значення функції $f(x,y,z)$ при $x = 2$, $y = 3$ та $z = 1$.

г) Зручним для Вас способом визначте, які змінні зберігаються у робочій області MATLAB.

д) Видаліть ці змінні.

3. ІНТЕРФЕЙС КОРИСТУВАЧА СИСТЕМИ MATLAB 6.5

З деякими елементами інтерфейсу MATLAB ми познайомилися вище. У цій главі ми розглянемо інтерфейс MATLAB більш докладно, однак зупинимося лише на основних елементах інтерфейсу MATLAB 6.5.

Меню користувача системи MATLAB 6.5 має стандартний для Windows-додатків набір меню, що містить наступні розділи (рис. 3.1):

- **File** (Файл);
- **Edit** (Редагування);
- **View** (Вид);
- **Web** (Мережа);
- **Window** (Вікно);
- **Help** (Допомога).

Рис. 3.1. Меню користувача системи MATLAB 6.5

Тепер розглянемо елементи меню докладніше.

3.1 Меню File

У меню **File** (Файл) наведені команди для роботи з файлами системи MATLAB 6.5, розкритий список команд меню представлений на рис. 3.2.

Команда **New** (Новий) розкриває список файлів, які можна створити за допомогою системи MATLAB 6.5:

- **M-file** - файл для написання скріптів і функцій;
- **Figure** - файл, у якому зберігаються графіки;
- **Model** - файли імітаційних моделей додатку системи MATLAB 6.5 - Simulink 5;
- **GUI** - файли графічного інтерфейсу.

Команда **Open**... (Відкрити...) дозволяє відкрити вже існуючі файли MATLAB. При виборі цієї команди з'являється діалогове вікно, що представлене на рис. 3.3.

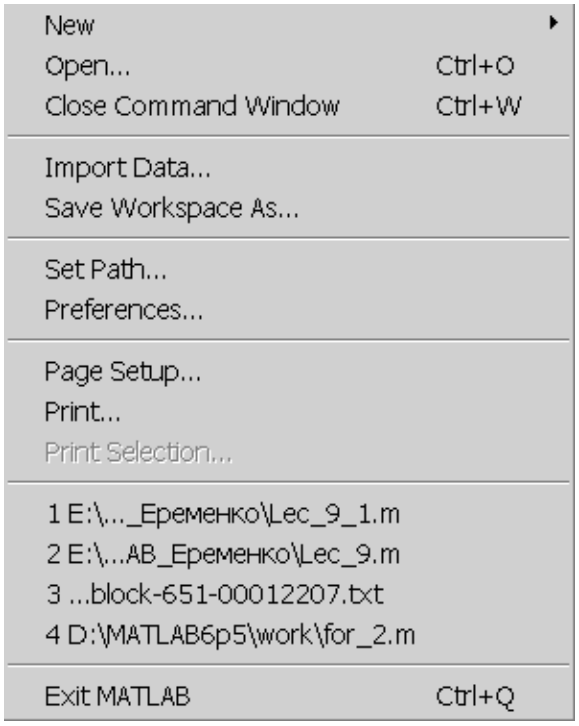


Рис. 3.2. Меню **File** (Файл)

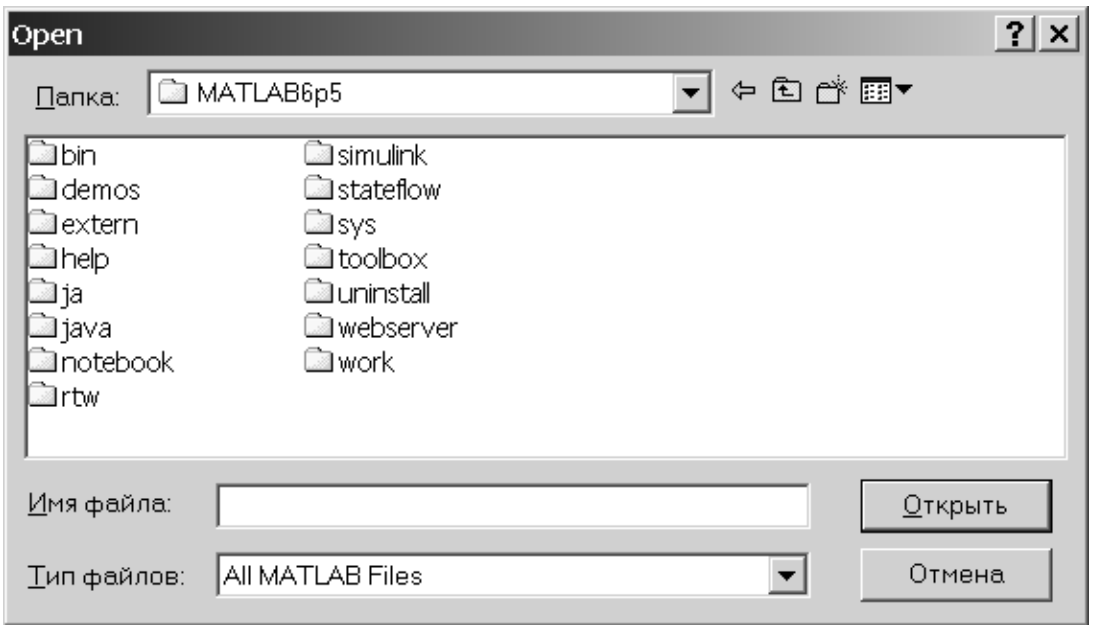
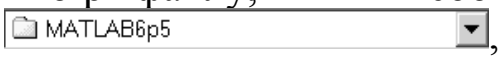


Рис. 3.3. Діалогове вікно команди **Open…** (Відкрити...)

У цьому вікні виконується вибір файлу, який необхідно відкрити. Для цього в списку 

, що розкривається і відображає дерево логічних дисків і каталогів, вказуємо каталог, у якому перебуває потрібний файл. Потім у

діалоговому вікні відзначаємо необхідний файл (ім'я файлу відображається в полі **Имя файла**) і відкриваємо його натисканням на кнопку Открыть. Кнопка Отмена дозволяє покинути діалогове вікно **Open…** (Відкрити...) без відкриття файлу.

Список **Тип файлов**, що розкривається**,** призначений для вибору типу файлів, що будуть відображатися у вікні **Open…**. Це полегшує пошук потрібного файлу на комп'ютері. Даний список дозволяє зробити вибір наступних типів файлів:

- **All MATLAB files** - всі файли, що підтримуються MATLAB;
- **M-files (*.m)** - m-файли із скріптами;
- **Figures (*.fig)** - графіки функцій;
- **MAT-files (*.mat)** – файли, що містять збережену робочу область MATLAB;
- **Models (*.mdl)** - файли імітаційних моделей Simulink;
- **All files (*.*)** - всі файли, що знаходяться у каталозі.

Команда **Close Command Window** закриває командне вікно MATLAB. Якщо активним є інше вікно, то замість **Command Window** буде вказана назва цього вікна і воно ж буде закриватися, тобто ця команда закриває активне вікно.

Команда **Import Data…**дозволяє робити вставку (імпортувати) даних, що не є файлами MATLAB. При виконанні цієї команди з'являється діалогове вікно **Import** (рис. 3.4), у якому відбувається вибір імпортованого файлу. Робота із цим діалоговим вікном аналогічна роботі з діалоговим вікном **Open…**. У списку, що розкривається, відображаються наступні типи файлів для імпортування:

- **Recognized data files** – всі файли даних;
- **MAT (*.mat)** – файли, що містять збережену робочу область MATLAB.
- **Text (*.txt, *.dat, *.dlm, *.tab)** – текстові файли;
- **All Files (*.*)** – всі файли, що знаходяться у каталозі.

Всі імпортовані дані привласнюються змінним MATLAB 6.5.

Команда **Save Workspace As…**дозволяє mat-файл MATLAB 6.5 (робочу область) зберегти під новим ім'ям. Діалогове вікно, що з'являється, надано на рис. 3.5. У цьому вікні вказується шлях до файлу та ім'я файлу в полі **Имя файла.**

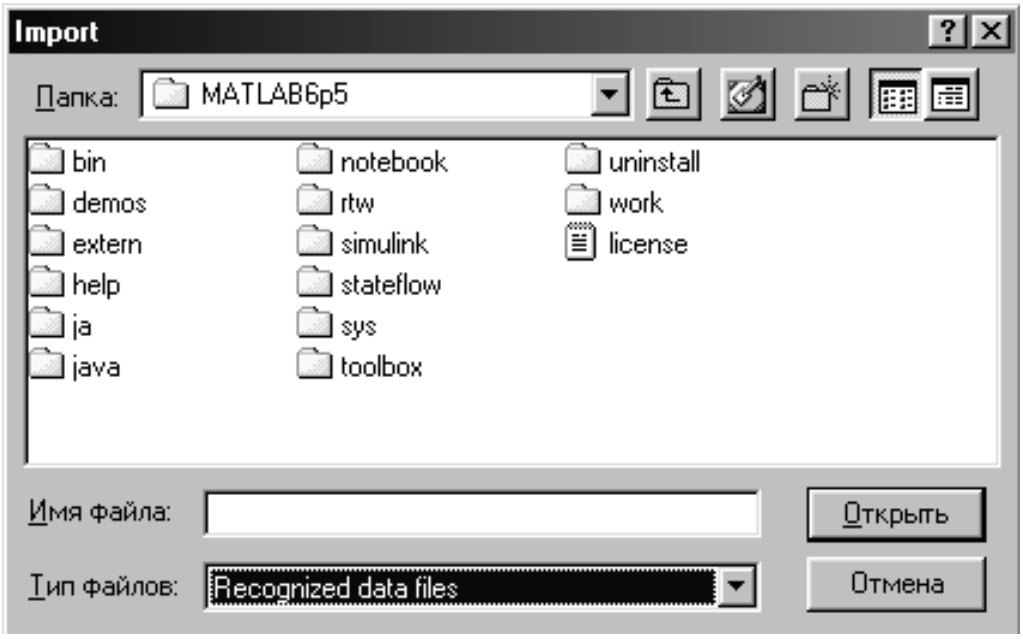


Рис. 3.4. Діалогове вікно команди **Import Data…**

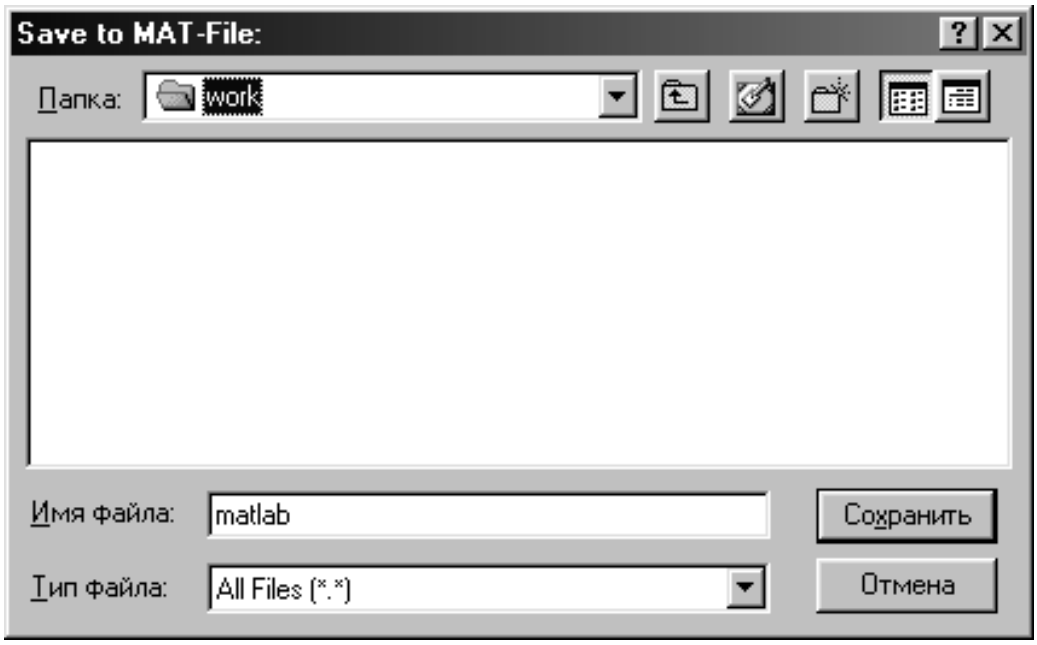


Рис. 3.5. Діалогове вікно команди **Save Workspace As**

Команда **Preferences** відкриває доступ до настроєк шрифтів, кольорів та інших опцій, які притаманні всім робочим вікнам MATLAB 6.5.

Команда **Page Setup** відкриває доступ до настроєк сторінки, тобто визначається який вигляд будуть мати дані, відображені у вікнах MATLAB, при перенесенні на тверду копію.

Команда **Print…**дозволяє зробити друк даних, відображених у зазначеному вікні MATLAB. На друк буде виводитись активне вікно.

Команда **Print Selection…**дозволяє вибрати принтер для друку.

Останні три команди виконуються тільки при наявності хоча б одного настроєного принтера на Вашому комп'ютері.

Команда **Exit MATLAB** закриває програмне середовище MATLAB.

3.2 Меню Edit

У меню **Edit** (Виправлення) наведені команди для роботи з даними, що відображені в робочих вікнах. Розкритий список команд меню представлений на рис. 3.6.

Рис. 3.6. Меню **Edit** (Виправлення)

Команда **Undo** здійснює скасування останньої дії.

Команда **Redo** здійснює повтор скасованої дії.

Команда **Cut** вирізає виділений фрагмент.

Команда **Copy** копіює виділений фрагмент.

Команда **Paste** вставляє раніше скопійований або вирізаний фрагмент.

Команда **Select All** виділяє всі дані в активному вікні.

Команда **Delete** видаляє виділений фрагмент.

Останньою групою команд у меню **Edit** (Виправлення) є команди очищення вікон середовища MATLAB:

- **Command Window** - вікно введення команд;
- **Command History** - історія введених команд у вікні введення команд;
- **Workspace** - робоча область, що містить список всіх змінних, які використовуються у даний момент.

3.3 Меню View

Меню **View** (Вид) містить команди для настроювання основних вікон MATLAB; розкритий список команд меню представлений на рис. 3.7.

Рис. 3.7. Меню **View** (Вид)

Desktop Layout розкриває список стандартних конфігурацій робочого стола MATLAB:

- **Default** - розміщення вікон за замовчуванням;
- **Command Window Only** - відкрите тільки командне вікно;
- **Simple** - залишає тільки вікно історії команд (**Command History**) і вікно для введення команд (**Command Window**);
- **Short history** - мінімізує вікно історії команд (**Command History**);

- **Tall History** - розвертає вікно історії команд (**Command History**) на всю доступну висоту робочого стола;
- **Five Panel** - відкриває всі п'ять основних вікон MATLAB.

Команда **Undock+ім'я вікна** дозволяє винести активне вікно за межі робочого стола MATLAB, для винесеного вікна ця команда перетворюється в **Dock+ім'я вікна**.

Наступний список вказує на те, які вікна MATLAB є відкритими (це вказує встановлена галочка (прапорець) напроти назви вікна).

Нагадаємо, що в MATLAB є п'ять основних вікон. Їх перелік відображений у меню **View** (рис. 3.7). Нижче зупинимося на кожнім вікні більш докладніше.

Command Window – вікно, всередині якого вводяться команди MATLAB. Ці команди засновані на мові програмування C++. Також у цьому вікні відображається поточна інформація про процеси моделювання та розрахунків.

Command History – вікно, що відображає історію введених команд за останні кілька днів роботи пакета.

Current Directory – вікно, у якому відображаються файли поточного каталогу, тут також можна змінити поточний робочий каталог.

Workspace – вікно, що відображає список всіх змінних, що в даний момент зберігаються у пам’яті, а також їхніх параметрів: ім'я змінної (**Name**), її розмірність (**Size**), об’єм пам'яті, що вона займає (**Bytes**), а також тип змінної (**Class**).

Launch Pad - вікно запуску програмних додатків (Toolboxes) MATLAB, допомоги і демонстраційних файлів. Для більш швидкої навігації по програмним додаткам у лівому нижньому куті MATLAB передбачена кнопка 

. По своєму призначенню вона аналогічна вікну **Launch Pad**.

Список, що розкривається, **Current Directory Filter** визначає фільтрацію файлів у вікні поточного каталогу (відображення файлів аналогічно діалоговому вікну **Open…**). Поточний каталог – це робочий каталог середовища MATLAB у даний момент. Всі посилання на скріпти та підсистеми повинні знаходитися в цьому каталозі. Поточний каталог можна змінювати під кожного конкретного користувача, щоб уникнути перемішування в одному

каталозі файлів різних користувачів.

Список **Workspace View Options** (рис. 3.8) представляє настройки перегляду робочої області, що містить список змінних, які використовуються.

Рис. 3.8. **Workspace View Options**

Перша група команд вказує параметри для відображення:

- **Show Size** - відображення розмірності;
- **Show Bytes** - відображення розміру в бітах;
- **Show Class** - відображення класу, що характеризує точність і вид змінних.

Друга група вказує тип сортування змінних.

Галочка напроти команди **Sort Ascending** вказує на сортування по зростанню в обраній категорії.

3.4 Меню Web, Windows і Help

У меню **Web** представлені Internet-посилання на різні сервісні служби фірми розроблювача MATLAB 6.5 - The Math Works, Inc.

Меню **Windows** відображає всі відкриті вікна в MATLAB, якщо вказати на будь-яке обране зі списку вікно, воно стає активним. Єдина команда меню - **Close All**, що дозволяє закрити всі вікна одночасно.

В останньому меню - **Help**, розміщені посилання на файли допомоги, а також демонстраційні файли, які були встановлені при інсталяції пакета. Всі файли допомоги в MATLAB представлені

англійською мовою. У меню **Help** особливий інтерес представляють команди **MATLAB Help** і **Demos**. При виконанні кожної з цих команд відкривається діалогове вікно, що представлене на рис. 3.9.

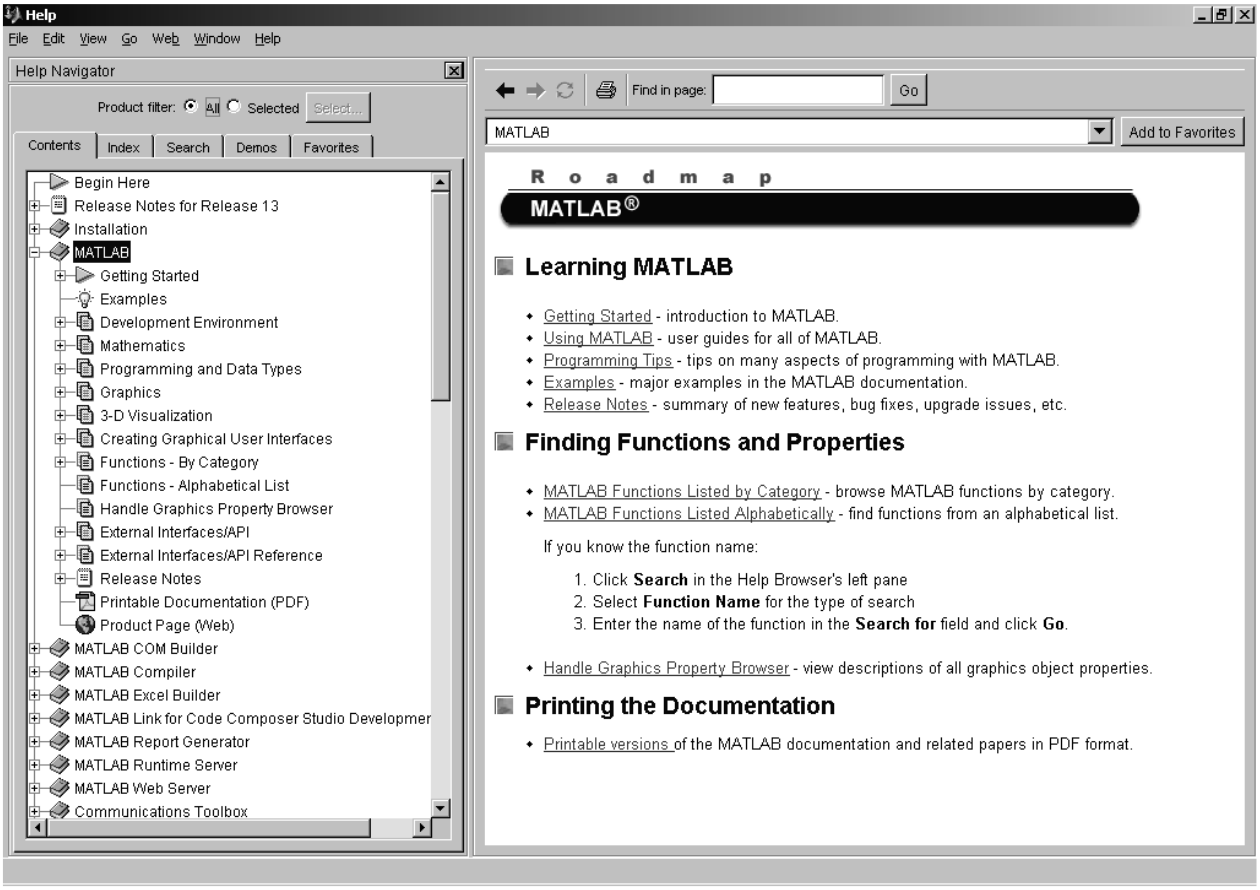


Рис. 3.9. Діалогове вікно **MATLAB Help**

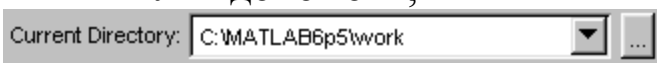
У залежності від обраної вкладки, в лівій частині діалогового вікна відображається дерево файлів допомоги (**Contents**), вікно пошуку за індексом (**Index**), вікно розширеного пошуку (**Search**), дерево демонстраційних файлів (**Demos**) або обрані посилання (**Favorites**), де за замовчуванням розміщено посілання на сайт підтримки. У правій частині діалогового вікна відображається інформація, що відповідає обраній позиції у її лівій частині.

3.5 Панель інструментів

MATLAB 6.5 має одну панель інструментів, що представлена на рис. 3.10.

Рис. 3.10. Панель інструментів MATLAB

Ця панель містить наступні елементи, які дублюють ті команди головного меню, що найбільш часто використовуються:

- - створення нового m-файлу;
- - відкриття файлу MATLAB 6.5;
- - вирізання виділеного фрагмента;
- - копіювання виділеного фрагмента;
- - вставка з буфера обміну;
- - скасування останньої дії;
- - повтор останньої скасованої дії;
- - виклик бібліотек Simulink;
- - виклик допомоги;
- 

- вибір поточного каталогу.

Завдання для самостійної роботи

1. Відкрийте демонстраційний М-файл rlcdemo.m з каталогу MATLAB6p5/toolbox/control/ctrldemos. Цей М-файл на прикладі RLC-ланцюга показує можливості MATLAB.

а) Запустіть М-файл натиснувши кнопку на панелі інструментів редактора М-файлів (детальніше про роботу з М-файлами дивиться у гл. 5).

б) У з'явившемуся вікні RLC Circuit Response Demo змінюючи тип RLC-ланцюга дослідіть вплив параметрів R, L та C на його характеристики.

2. Розгляньте можливості MATLAB за допомогою інших демонстраційних прикладів, що можна знайти у меню **Help → Demos**.

4. ПОБУДОВА ГРАФІКІВ В MATLAB. ІНТЕРФЕЙС КОРИСТУВАЧА ГРАФІЧНОГО ВІКНА

4.1 Побудова графіків функцій однієї змінної

Розглянемо, як побудувати в MATLAB найпростійший графік. Для цього задамо масив *x* у межах від -2π до +2π із шагом 0,01. Швидше і зручніше за все масив задавати в командному вікні (**Command Window**).

```
>> x=-2*pi:0.01:2*pi
```

Тут вказуємо ім'я змінної *x,* якій привласнюємо масив, а після знака рівності розділяємо двокрапкою параметри масиву. У такий спосіб одержуємо наступний формат команди:

```
x=перше значення масиву:шаг:останнє значення масиву
```

Нагадаємо, що введення кожної команди для виконання завершується натисканням клавіші <Enter> на клавіатурі.

В MATLAB не можна пропускати математичні знаки, інакше буде видане повідомлення про помилку. Для більшої точності розрахунків необхідно використовувати не число 3.14, а вбудовану змінну MATLAB *pi*, оскільки їй відповідає значення π, що враховує більш ніж два знака після коми.

Після введення вказаної команди у вікні **Workspace** буде відображена нова змінна з усіма властивими їй параметрами (ім'я, розмірність, розмір, тип) і у вікно **Command History** буде занесена ця команда. У командному вікні (**Command Window**) буде відображений весь введений масив. Якщо немає необхідності переглядати введений масив даних, а хотілося б бачити в командному вікні всі набирані команди, не використовуючи при цьому смугу прокручування, то наприкінці введеної команди потрібно вказати крапку з комою.

Далі розраховуємо значення функції синуса для кожного значення введеного масиву *x*. Для цього набираємо в командному рядку:

```
>> y=sin(x);
```

Не можна забувати, що рядкова і прописна букви в MATLAB – це різні змінні. Для побудови графіка синусоїди застосуємо команду друку на екран, що має наступний формат:

```
plot(ім'я аргументу, ім'я функції)
```

Отже, наберемо:

```
>> plot(x,y);
```

У результаті відкривається нове вікно - Figure No. 1, у якому розташований наш графік. На рис. 4.1 представлене вікно Figure No. 1, а на рис. 4.2 види вікон: **Workspace, Command History** і **Command Window**.

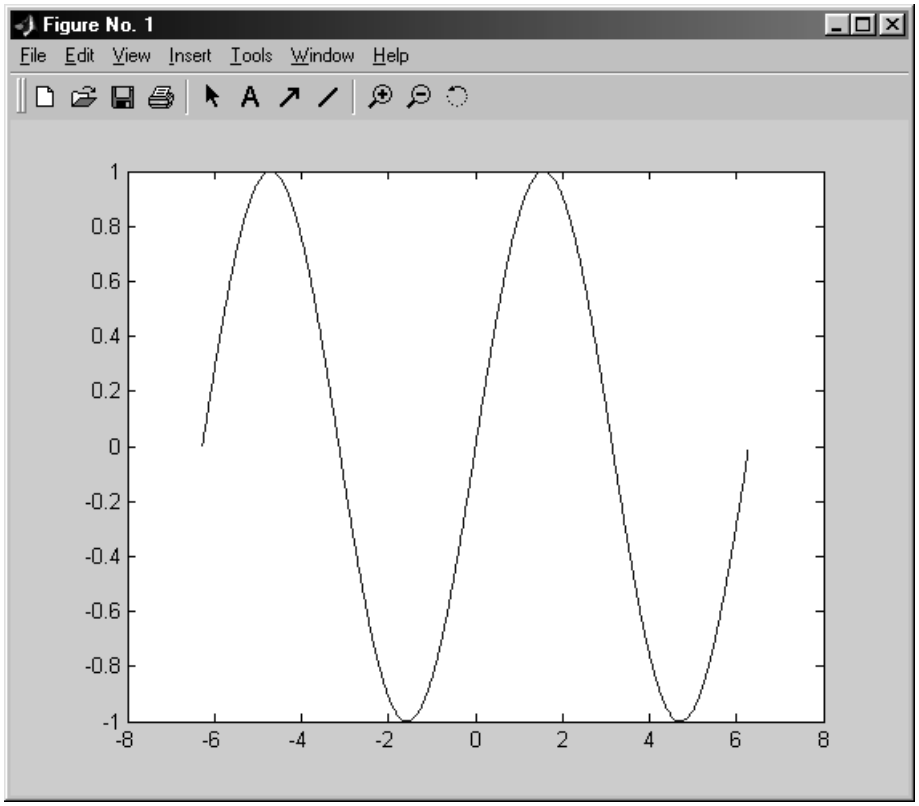


Рис. 4.1. Графічне вікно Figure No. 1

При повторному використанні команди `plot` попередній графік замінюється новим. Для того, щоб вивести графік у іншому графічному вікні використовується команда `figure(n)`, де *n* – номер нового графічного вікна.

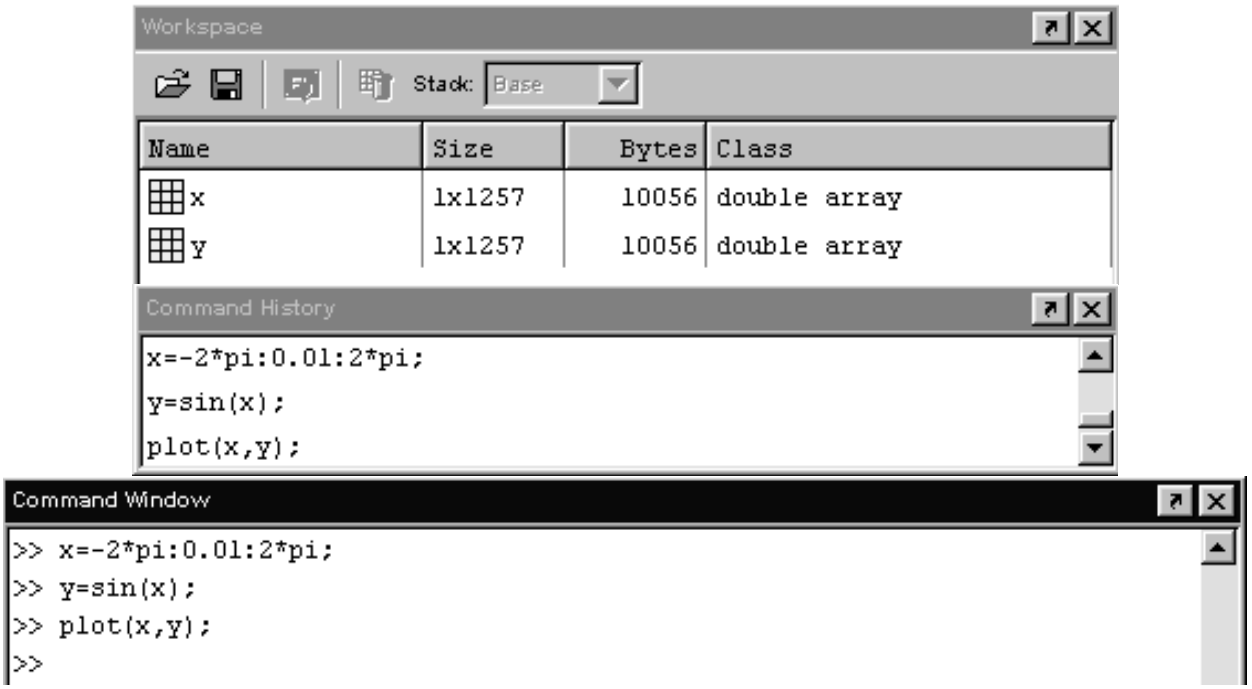


Рис. 4.2. Вікна **Workspace**, **Command History** і **Command Window** після виконання команд

MATLAB дозволяє побудову декількох графіків в одному вікні. Це можна зробити декількома способами.

По-перше, при використанні команди `plot` можна одночасно задавати декілька пар аргументів та функцій, що відокремлюються одна від одної комою, наприклад

```
>> x=0:pi/100:2*pi;
>> y1=sin(x);
>> y2=sin(x-0.25);
>> y3=sin(x-0.5);
>> plot(x,y,x,y2,x,y3)
```

По-друге, можна використовувати команду `hold on`, яка дозволяє виводити наступний графік у останнє графічне вікно не стираючи попередні графіки:

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

По- третє, графічне вікно можна розбити на декілька частин (у вигляді таблиці). Для цього використовується команда `subplot`,

що вказує вид розміщення графіків у вікні і команда `plot` для їхньої побудови. Ці команди зручно набирати в один рядок, розділяючи їх комою або крапкою з комою:

```
subplot(кількість рядків, кількість стовпців,
№ активної комірки);plot(ім'я аргументу,ім'я функції)
```

За допомогою зв'язки цих команд побудуємо в одному вікні два графіки - функцію синуса і функцію косинуса масиву *x*. Нижче наданий текст команд, який буде необхідно ввести за допомогою командного рядка.

```
>> x=-2*pi:0.01:2*pi; % Введення масиву x
>> y1=sin(x); %розрахунок значень синуса
>> y2=cos(x); %розрахунок значень косинуса
>>%побудова синусоїди:
>> subplot(1,2,1);plot(x,y1)
>>%побудова косинусоїди
>> subplot(1,2,2);plot(x,y2)
```

На рис. 4.3 представлене графічне вікно, що відкривається в результаті виконання наведеної вище програми, а на рис. 4.4 - вид вікон **Workspace** і **Command History**.

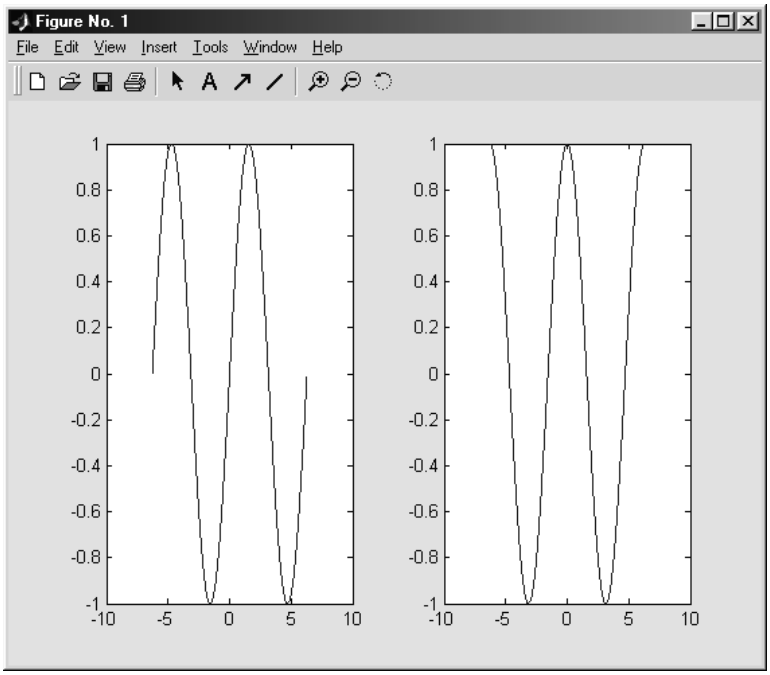


Рис. 4.3. Вид графічного вікна Figure No. 1
після застосуванням команди `subplot`

Далі розглянемо інтерфейс графічного вікна, що представлене на рис. 4.1 та 4.3.

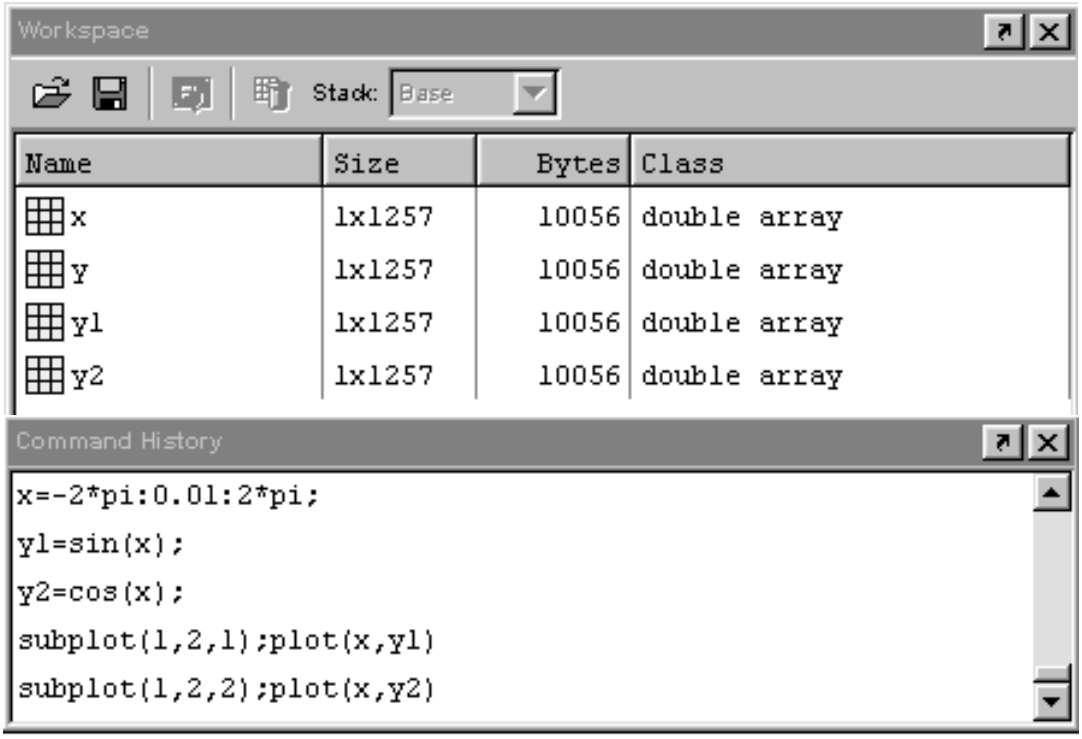


Рис. 4.4. Вікна **Workspace** і **Command History**
після виконання команд

4.2 Інтерфейс графічного вікна MATLAB

Графічний інтерфейс MATLAB також відповідає канонам сучасного інтерфейсу Windows – додатків. Графічне вікно (рис. 4.5) містить меню і панель інструментів, що забезпечують великі можливості при роботі із графікою.

У меню виділені групи однорідних операцій. Наприклад, команди підміню **File** забезпечують збереження побудованої фігури у файлі або у вигляді надрукованої на принтері твердої копії; команди підміню **Edit** відповідають за оформлення ‘ фігур і т.д.

На панелі інструментів виведені кнопки, що відповідають тим командам головного меню, до яких користувачі звертаються найчастіше.

Меню графічного вікна має наступні розділи:

- **File** (Файл);
- **Edit** (Виправлення);

- **View** (Вид);
- **Insert** (Вставка);
- **Tools** (Інструменти);
- **Window** (Вікно);
- **Help** (Допомога).

Меню графічного вікна MATLAB версії 6.5 представлене на рис. 4.5.

Рис. 4.5. Меню графічного вікна MATLAB 6.5

У меню **File** (Файл) наведені команди для роботи із графічними файлами; розкритий список команд цього меню представлений на рис. 4.6. Перелічимо коротко призначення команд меню **File** (Файл), що взагалі аналогічно однойменному меню в інтерфейсі робочого стола системи MATLAB 6.5.

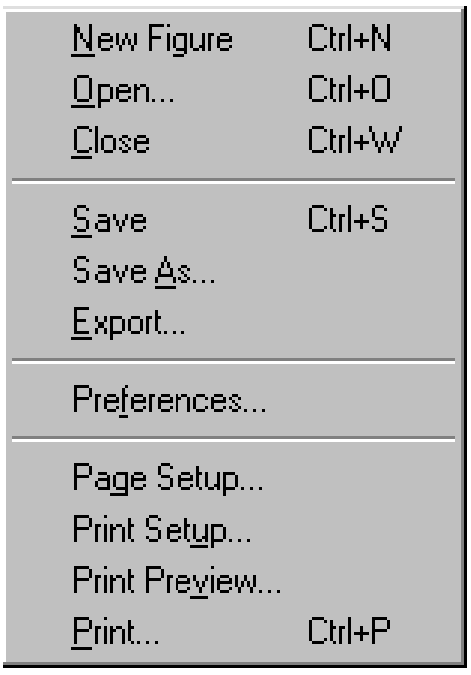


Рис. 4.6. Меню **File** (Файл)

New Figure – відкриття нового порожнього графічного вікна. При використанні команд `plot` і `subplot` ця команда виконується автоматично. На панелі інструментів даній команді відповідає іконка  .

Open… – відкриття раніше збережених графічних файлів MATLAB (рис. 3.3). За замовчуванням у вікні, що відкривається, значаться файли з розширенням <*.fig*>. На панелі інструментів цій команді відповідає іконка .

Close - закриття активного графічного вікна.

Save – збереження активного графічного вікна. Цю операцію дублює іконка на панелі інструментів.

Save As… – збереження активного графічного вікна під іншим ім'ям.

Export… – експортування вмісту графічного вікна у файл будь-якого графічного формату *(*.emf, *.bmp, *.jpg, *.tif* і т.д.).

Preferences… – відкриття вікна властивостей системи MATLAB.

Page Setup… – настроювання параметрів сторінки перед друком на тверді носії.

Print Setup… – настроювання поточного принтера.

Print Preview… – перегляд вмісту графічного вікна перед друком.

Print… – друк вмісту графічного вікна. Цій команді відповідає іконка на панелі інструментів.

Останні три команди виконуються тільки у випадку, якщо на комп'ютері встановлений хоча б один друкувальний пристрій.

У меню **Edit** (Виправлення) наведені команди для роботи з даними. Розкритий список команд меню представлений на рис. 4.7.

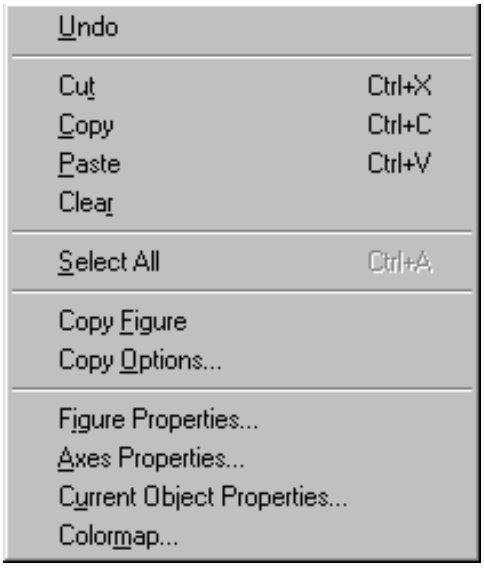


Рис. 4.7. Меню **Edit** (Виправлення)

Команда **Undo** скасовує останню дію.

Команда **Cut** вирізає виділений фрагмент.

Команда **Copy** копіює виділений фрагмент.

Команда **Paste** вставляє раніше скопійований або вирізаний фрагмент.

Команда **Clear** очищує виділену область.

Команда **Select All** робить виділення всіх елементів.

Як правило, отримані в MATLAB результати необхідно вставляти в інші додатки, наприклад в Word. Для цих цілей дуже зручно використовувати команду **Copy Figure**, що робить копіювання всієї робочої області (без меню і панелей) графічного вікна. А команда **Copy Options…**відкриває діалогове вікно для настроювання параметрів цього копіювання (рис. 4.8).

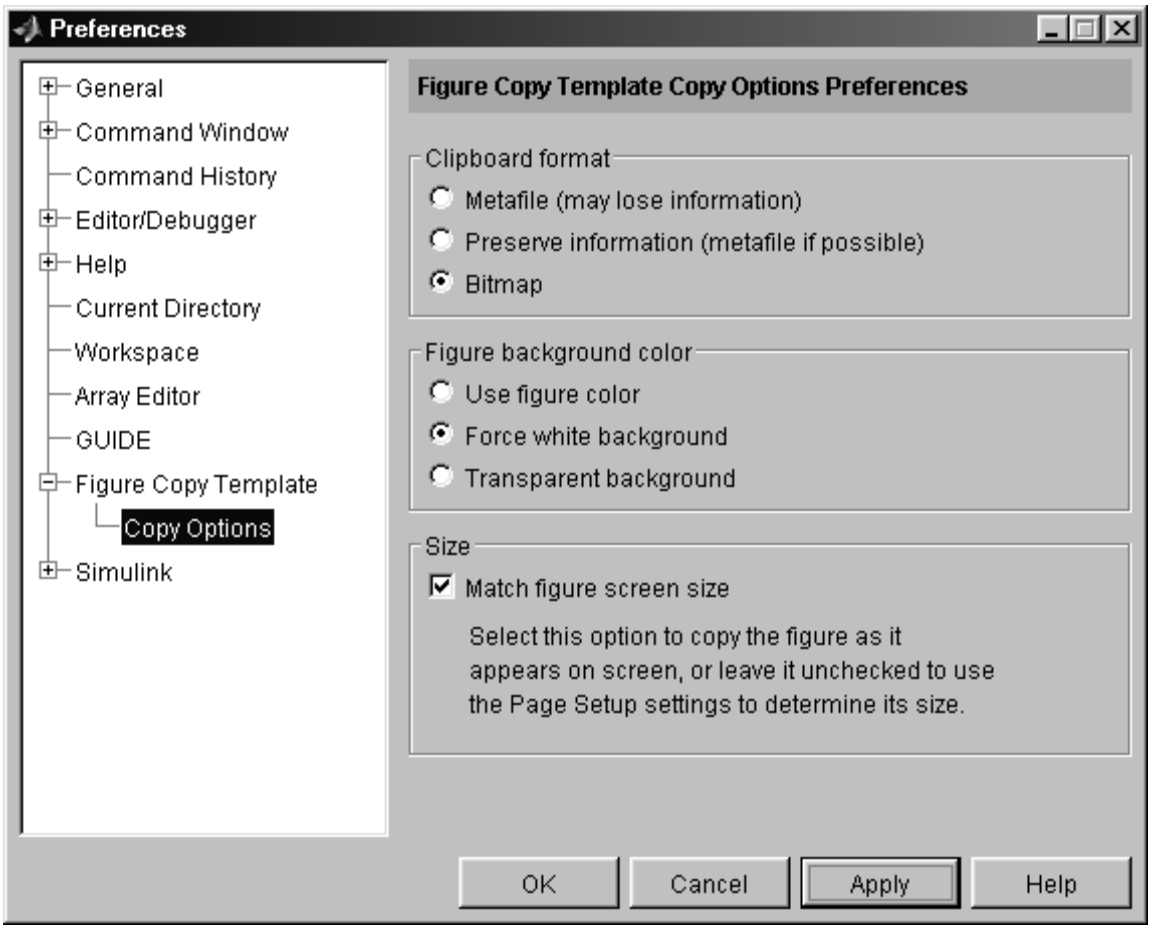


Рис. 4.8. Діалогове вікно **Preferences** для настроювання параметрів **Copy Options**

У цьому діалоговому вікні можна задати наступні параметри.

1) Формат даних для передачі в буфер обміну операційної системи:(**Clipboard format**):

- **Metafile** – створення метафайла з можливою втратою даних;
- **Preserve information** – створення метафайла, якщо це можливо (без втрати даних);
- **Bitmap** – створення формату точечного рисунка **.bmp*.

2) Колір фону (**Figure background color**):

- **Use figure color** – встановлюється поточний колір (сірий за замовчуванням);
- **Force white background** – білий фон;
- **Transparent background** – прозорий фон.

3) Опція **Match figure screen size** дозволяє копіювати графічне вікно таким, яким воно створюється (без трансформації).

Команда **Figure Properties** відкриває доступ до настроювань властивостей самого графічного вікна (рис. 4.9).

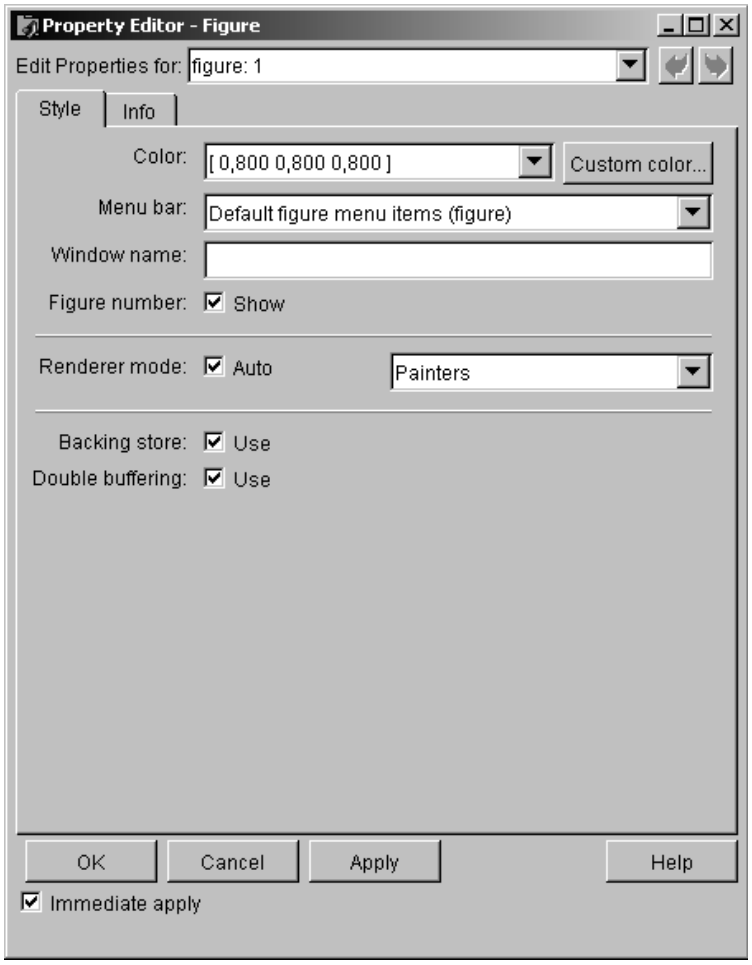
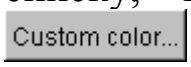


Рис. 4.9. Діалогове вікно для настроювання параметрів **Property Editor**

Саме верхнє поле цього вікна - список, що розкривається, **Edit Properties for:** у якому можна вибрати фігуру, для якої потрібно змінити властивості. Зрозуміло, для цього побудованих фігур повинно бути декілька.

За замовчуванням у вікні **Property Editor** відкривається закладка **Style** (Стиль). У цьому вікні можна змінити кольори вікна фігури. Для цього потрібно вибрати один з основних кольорів у списку, що розкривається, **Color**: або натисканням кнопки 

викликати діалогове вікно для вибору довільних кольорів.

В списку, що розкривається, **Menu bar:** вибирається одна з опцій:

- **Default figure menu items (figure)** - відображення меню і панелі інструментів у вікні обраної фігури;
- **No menu items (none)** - меню і панель інструментів у вікні фігури відсутні.

Поле введення **Window name:** призначено для введення імені вікна фігури. Після заповнення цього поля в назві графічного вікна після Figure No. 1 з'являється введений напис.

Встановлена напроти опції **Figure Number** галочка вказує на відображення номера графічного вікна в його імені.

Аж унизу вікна є опція **Immediate apply**, якщо вона відзначена, то зроблені зміни всередині діалогового вікна моментально набувають чинності, як після натискання кнопки 

- застосувати.

При виклику команди **Axes Properties** з'являється діалогове вікно властивостей осей (рис. 4.10).

Розглянемо основні закладки діалогового вікна властивостей вісей - **X**, **Y**, **Z** і **Style**. Закладки **X**, **Y**, **Z** мають однакові настроювання, тільки вони відносяться до обраної вісі, тому розглянемо їх на прикладі вісі **X**.

У полі введення **Label:** указується ім'я вісі. У списку **Color:**, що розкривається, вибирається колір вісі. У списку **Location:**, що розкривається, вказується розташування числових написів уздовж вісі. Якщо відмічено опцію **Grid:** то на графіку буде відображена координатна сітка.

Наступний набір параметрів:

- **Limits** - вказує межі шкали;
- **Ticks** - вказує, які значення шкали необхідно відобразити;
- **Labels** - вказує напис, який можна нанести уздовж вісі.

За замовчуванням перераховані параметри мають настроювання **auto**, але при необхідності можна зняти галочку і вказати в текстових полях, що активуються, потрібні параметри.

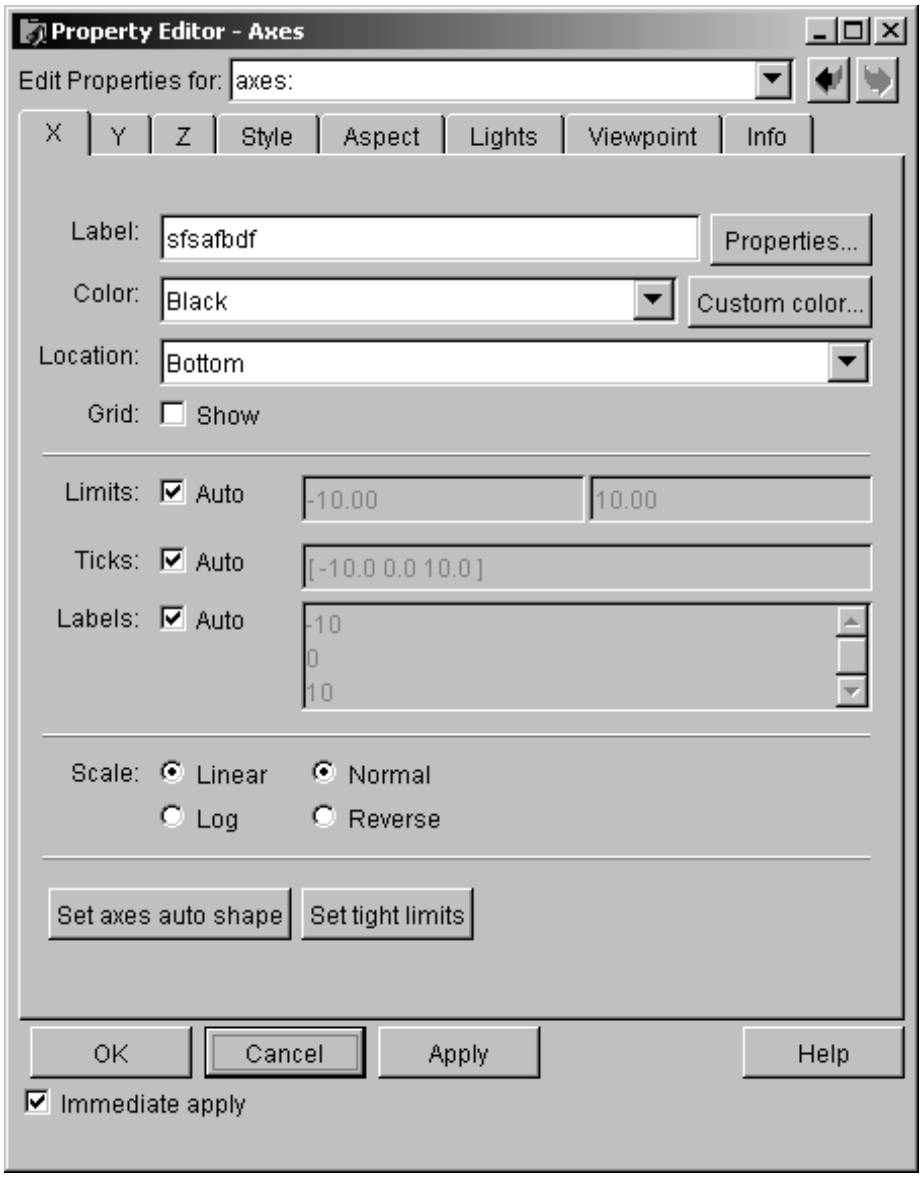


Рис. 4.10. Діалогове вікно **Axes Properties**

В наборі параметрів **Scale:** розташовані дві пари зв'язаних між собою параметрів. Перша пара:

- **Linear** - лінійна шкала;
- **Log** - логарифмічна шкала.

Друга пара:

- **Normal** - напрямок вісі з ліва направо;
- **Reverse** - напрямок вісі з права на ліво.

І, нарешті, дві кнопки: Set axes auto shape - установка параметрів за замовчуванням у режим **auto** і Set tight limits - установка параметрів шкали (**Limits**, **Ticks**, **Labels**) за розміром графіка.

Завдання для самостійної роботи

1. Побудуйте графіки наступних функцій

а) $y = 3\sin x + 2\cos x$ при $-2\pi \le x \le 2\pi$ із шагом 0,05.

б) У другому графічному вікні побудуйте графік функції $y_1 = \mathrm{tg}x_1$, де $0 \le x_1 \le \pi$, а шаг дорівнює 0,01. Для виводу нового графічного вікна використайте команду `figure`.

в) На обидва графіка нанесіть координатну сітку.

2. До графіку з завдання 1,*а* за допомогою команди `hold on` додайте графіки наступних функцій

а) $y_2 = 3\sin x - 2\cos x$;

б) $y_3 = \sin x + \cos x$.

3. Для графіків функцій $y(x)$, $y_2(x)$ та $y_3(x)$, що побудовані при виконанні попередніх завдань, встановіть наступні параметри відображення ліній

а) графік функції $y(x)$: колір графіку - чорний, тип лінії – сполошна з маркером у вигляді символу «*», ширина лінії – 1,5;

б) графік функції $y_2(x)$: колір графіку - червоний, тип лінії – пунктир, ширина лінії – 2;

в) графік функції $y_3(x)$: колір графіку - блакитний, тип лінії – штрих-пунктир, ширина лінії – 3;

4. Відкрийте нове графічне вікно, за допомогою команди `subplot` розділіть його на чотири частини та у кожній з них побудуйте по одному з графіків функцій $y(x)$, $y_1(x)$, $y_2(x)$ та $y_3(x)$.

5. М-ФАЙЛИ СЦЕНАРІЇВ І ФУНКЦІЙ

5.1 Редактор М-файлів

Як правило, рішення серйозних інженерних задач вимагає написання програм значного обсягу. При цьому робота в командному рядку вікна **Command Window** стає вкрай незручною. У цих випадках рекомендується користуватися спеціальним вбудованим редактором М – файлів, який можна викликати настуаним чином:

- набравши команду `edit` у командному рядку,
- вибравши **New→M-file** з меню **File** або
- натиснувши кнопку 🗋 на панелі інструментів.

Імена програм, написаних у цьому редакторі мають формат <*ім'я_файлу.m*>, звідси і їхня назва М – файли.

Редактор М-файлів має своє меню та панель інструментів (рис. 5.1).

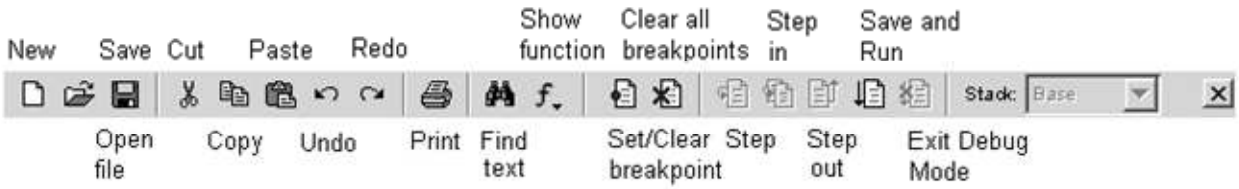


Рис. 5.1. Панель інструментів редактора/відладчика

Призначення кнопок панелі інструментів редактора наступне:

- **New** - створення нового М-файлу;
- **Open** - вивід вікна завантаження файлу;
- **Save** - запис файлу на диск;
- **Print** - друк вмісту поточного вікна редактора;
- **Cut** - вирізання виділеного фрагмента і перенос його в буфер;
- **Copy** - копіювання виділеного об'єкта в буфер;
- **Paste** - розміщення фрагмента з буфера в позиції текстового курсору;
- **Undo** - скасування попередньої операції;
- **Redo** - повтор скасованої операції;
- **Find text** - знаходження зазначеного тексту;

- **Show function** - показ функції;

- **Set/Clear Breakpoint** - установка/скидання точки переривання;

- **Clear All Breakpoints** - скидання всіх точок переривання;

- **Step** - виконання шагу трасування;

- **Step In** - пошагове трасування із заходом у викликувані М-файли;

- **Step Out** - пошагове трасування без заходу у викликувані М-файли;

- **Save and Run** - запис та виконання програми;

- **Exit Debug Mode** - вихід з режиму відладки.

Призначення М - файлів – розвантажити основну програму від другорядних розрахунків, зробити її більш чіткою і ясною.

Припустимо, що конструктор вирішує завдання побудови графіка залежності напруги Sigma (Н/м^2) у точці *A* конструкції, що зображена на рис. 5.2, від зусилля *F*(Н), прикладеного в точці *B*. При цьому визначення напруги Sigma у силу складності конструкції виконується шляхом дуже складних розрахунків.

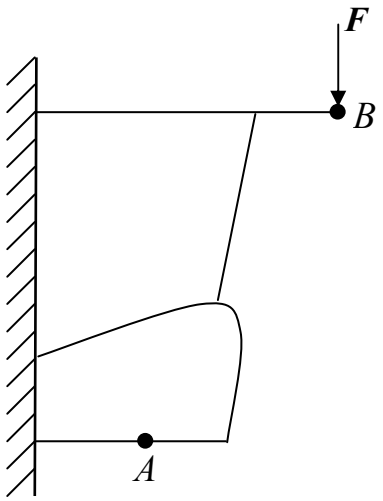


Рис. 5.2. Деяка конструкція

Нехай реально можливі зусилля *F* розташовуються в інтервалі [900 … 9900] Ньютонів. Конструктор вирішив визначити напруги Sigma при *F* = 900, 1900, 2900, 3900, 4900, 5900, 6900, 7900, 8900, 9900 Н, тобто, починаючи від мінімального до максимального *F* з інтервалом 1000 Н. Як він справедливо думав, плавна крива, що з'єднує значення Sigma при різних *F* дасть йому загальну картину

можливих напруг у точці *A* конструкції. Тоді напрошується наступна структура програми:

```
>>F=900:1000:9900;
>> Перший оператор у процедурі розрахунку Sigma;
>> Другий оператор у процедурі розрахунку Sigma;
… … … … … … … … … … … … … … … … … … … … … … …
>> n-й оператор у процедурі розрахунку Sigma;
>>plot(F, Sigma)
```

Нагадаємо, що перший оператор задає послідовність *F*, при яких будуть розраховуватися Sigma. Операторів, послідовність яких приводить до визначення Sigma при розглянутому *F*, може бути дуже багато. Оператор `plot(F,Sigma)` ініціює побудову графіка залежності Sigma = Sigma(*F*), що і потрібно конструкторові.

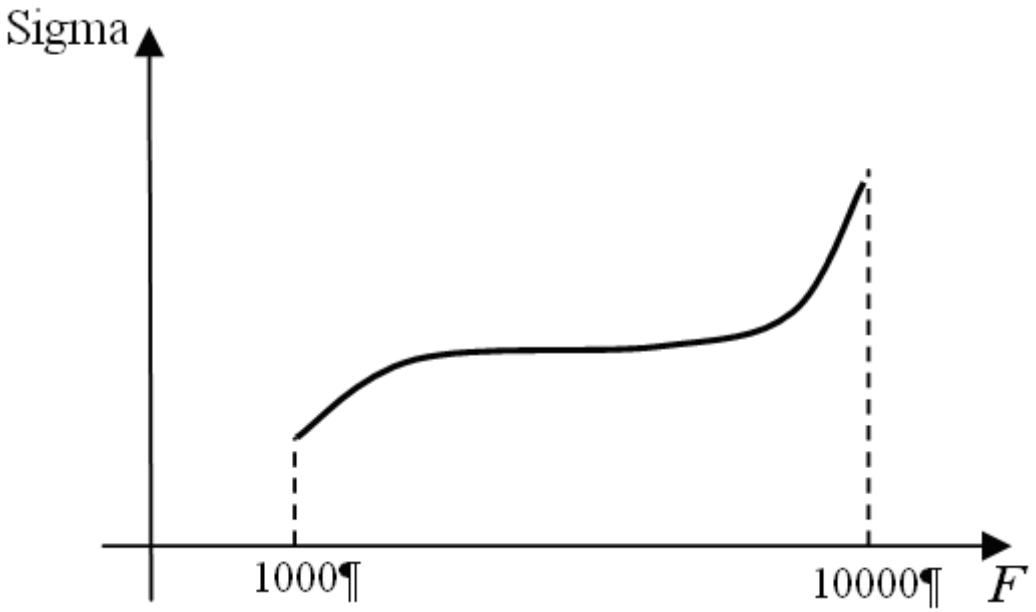


Рис. 5.3. Результат виконання програми

Якщо операторів, послідовність яких приводить до визначення Sigma при розглянутому *F*, багато - 50, 100, 150 і т.п., то можна взагалі перестати розуміти основний задум всіх розрахунків. Як говорить народне прислів'я, «за деревами не буде видно лісу». Тому у всіх високоорганізованих машинних мовах - Pascal, C, C++ і мові MATLAB прагнуть виділити розрахунки в так звані підпрограми, щоб головна програма ясно показувала, яка задача вирішується, а

підпрограма виконувала «чорнову» роботу. М-файли - це і є підпрограми мови MATLAB.

Є два типи М - файлів: файли - сценарії (які ще називають Script – файлами, *script* – якийсь офіційний документ, сценарій) і файл-функції (file-function).

Познайомимося з ними, а потім визначимо сферу застосування.

5.2 Script-файли

Script-файл є просто частиною головної програми (сесії), але виділеною в окремий блок і оформленою не в **Command Window**, а в редакторі М-файлів (у принципі, можна використати будь-який текстовий редактор). «Культурно» складені файли-сценарії мають наступну структуру:

```
1. %Основний коментар
2. %Додатковий коментар
3. Тіло файлу з будь-якими вираженнями
```

Знак % відсотка має зелені кольори і відкриває рядок коментарю. Оскільки MATLAB є англомовним продуктом, то він «недолюблює» кирилицю, тому коментарі краще вводити латинськими буквами. Але практика показує, що при невеликій кількості знаків використання кирилиці в коментарях можливо.

Зрозуміло, що створений файл повинен бути збережений на диску з якимсь ім'ям. Збереження здійснюється натисканням кнопки Save або Save and Run на панелі інструментів редактора М-файлів. Крім того можна скористатися відповідними командами головного меню або відповідними сполученнями клавіш (**Ctrl+S** або **F5**). Припустимо, що у розглянутому вище прикладі з конструктором, він вирішив дати файлу-сценарію ім'я «PasSigma». Нагадаємо, що файл буде мати розширення *.m*. Зверніть увагу, що на відміну від коментарів, у яких використання кирилиці часом «сходить з рук», застосування її в іменах файлів неприпустимо.

Script-файл працює з даними з робочої області (**Workspace**) сесії. Якщо в Script-файлі оголошуються нові змінні, які ще не оголошувалися в сесії, то ці змінні (і їхні значення) фіксуються в

Workspace і можуть далі використатися в сесії. Іншими словами, сесія і Script-файл мають загальні (глобальні) змінні. Тому Script-файлу не потрібні вхідні і вихідні змінні.

Отже, нехай програма має вигляд:

```
>>F=[900:1000:9900];
>>a=F/2
>>b=a+100
>>Sigma=500*sqrt(b)
>>plot(F,Sigma)
```

Тут послідовність розрахунків по визначенню Sigma при розглянутому *F* зведена в навчальних цілях усього до трьох операторів. Зараз не важливо, три їх або тисячі, оскільки ми освоюємо саме поняття Script-файлу. Тоді рішення завдання можна забезпечити двома програмами: «залишком» вихідної програми, з якої вилучені всі розрахунки, і Script-файлом, якому буде привласнене ім'я PasSigma:

```
>>%Основна програма в Command Window
>>F=[900:1000:9900];
>>PasSigma;  %Розширення вказувати не потрібно
>>plot(F,Sigma)
```

```
1. %Script-file PasSigma
2. %20.11.06. Конструктор Шевчук І.І.;
3. % Складна консоль. Напруга в точці А;
4.a=F/2;
5.b=a+100;
6.Sigma=500*sqrt(b);
7.disp('F='); disp(F);
8.disp('Sigma='); disp(Sigma);
9.pause(30);
10.disp('Ура! Виходить');
```

Вікно редактора М-файлів з Script-файлом PasSigma і **Command Window** з результатами виконання основної програми і Script-файлу представлені на рис. 5.4, а побудований графік залежності Sigma(*F*) – на рис. 5.5.

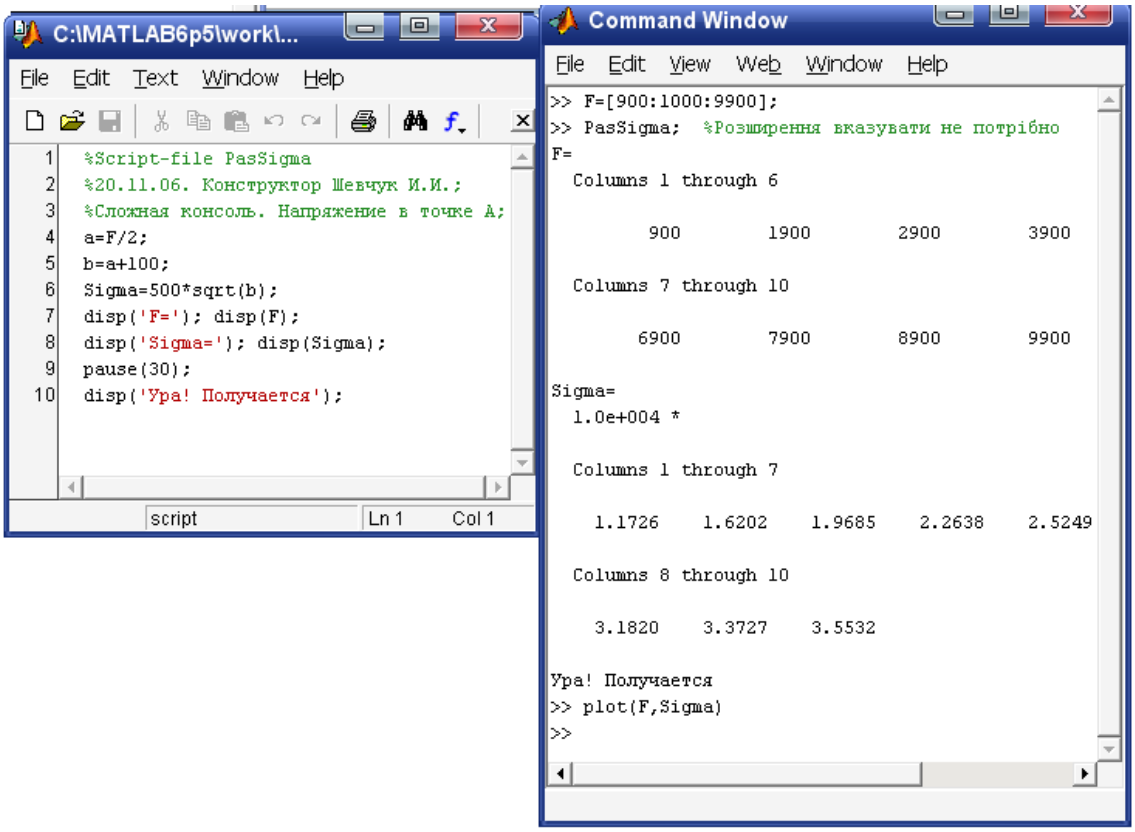


Рис. 5.4. Визначення Sigma

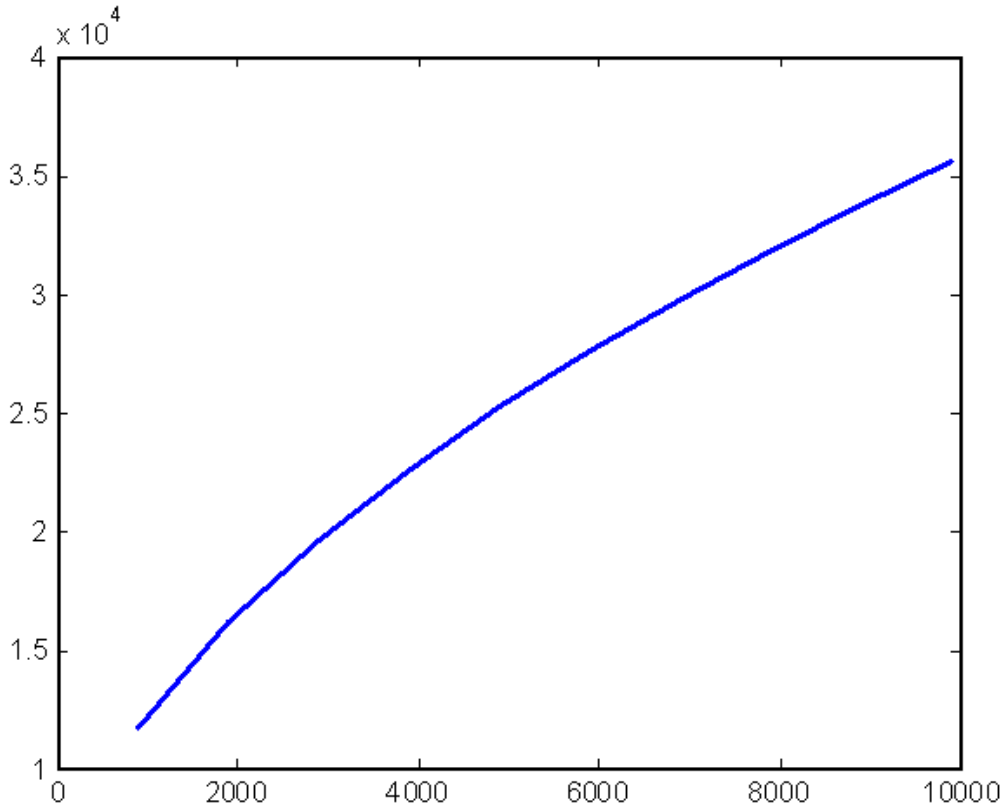


Рис. 5.5. Графік залежності Sigma(F)

Звернемо увагу на наступні деталі.

1. Програма спочатку розраховує 10 значень змінної *a*.

2. Потім розраховує 10 значень змінної *b*.

3. І тільки потім 10 значень Sigma.

4. Оператори `disp` організують вивід у командне вікно відразу всіх 10 значень *F* і всіх 10 значень Sigma.

5. Через 30 секунд з'являється коментар «Ура! Виходить!».

Наприкінці відзначимо, що іноді зручно відновлювати в пам'яті зміст Script-файлу по його коментарях, не виходячи з **Command Window**. Для цього в **Command Window** необхідно набрати команду наступного формату:

```
help шлях до файлу\ім'я файлу
```

Наприклад, якщо Script-файл PasSigma.m перебуває в папці work, то команда буде мати вигляд:

```
>> help work\ PasSigma.m  <Enter>

Script-file PasSigma
20.11.07. Конструктор Шевчук I.I.;
Складна консоль. Напруга в точці A;
```

5.3 Файли - функції

Уявимо собі, що Script-файл має наступний зміст:

```
1. %Можливий варіант Script-file
2. %Script-file PasSigma
3. %20.11.07.  Конструктор Шевчук I.I.;
4. %Складна консоль. Напруга в точці A;
5.a=F/2;
6.b=a+100;
7.c=sqrt(b);
8.z=c+1;
9.g=z+0.2;
10. %Зверніть увагу, починаються
11. %ті ж самі операції
12.a1=g/2;
13.b1=a1+100;
```

```
14.c1=sqrt(b1);
15.z1=c1+1;
16.g1=z1+0.2;
17. %Змінна g1 розраховується по g так само,
18. %як змінна g по F
19.Sigma=500*sqrt(g1);
```

Сам Script-файл втратив ясну структуру. Він має великий обсяг і у ньому двічі виконуються групи однакових операцій $F \to g$, $g \to g1$. Проблему вирішує використання файлу - функції (file-function):

```
1. %File-function Ra
2.function v=Ra(s)
3.k1=s/2;
4.k2=k1+100;
5.k3=sqrt(k2);
6.k4=k3+1;
7.v=k4+0.2;
```

Тоді Script-файлу можна надати вид:

```
1.Script-file PasSigma
2. %20.11.06.  Конструктор Шевчук I.I.;
3. %Складна консоль. Напруга в точці A;
4.g=Ra(F);
5.g1=Ra(g);
6.Sigma=500*sqrt(g1);
```

Тепер задачу побудови графіка залежності Sigma=Sigma(F) вирішують три програми, що читаються легко:

1) головна, текст якої розташовується в **Command Window**:

```
>>%Основна програма в Command Window:
>>F=[900:1000:9900];
>>PasSigma;
>>plot(F,Sigma)
```

2) Script-файл PasSigma:

```
1. %Script-file PasSigma
2. %20.11.06. Конструктор Шевчук I.I.;
```

```
3. %Складна консоль. Напруга в точці A;
4.g=Ra(F);
5.g1=Ra(g);
6.Sigma=500*sqrt(g1);
```

3) файл-функція:

```
1. %File-function Ra
2.function v=Ra(s)
3.k1=s/2;
4.k2=k1+100;
5.k3=sqrt(k2);
6.k4=k3+1;
7.v=k4+0.2;
```

Наведемо правила формування файлів-функції.

1. Перший оператор обов'язково має наведений вигляд, де спочатку пишеться слово `function`.

2. Потім через пробіл вказується ім'я змінної, значення якої розраховується у файлі-функції й повертається в програму, що викликається.

3. Далі розташовується знак рівності, ім'я файлу і у дужках – ім'я формальної змінної.

4. Останнім оператором у файлі-функції *обов'язково є оператор, в якому визначається значення змінної, що повертається.*

Файл-функція може містити коментарі.

Практика показує, що особливо важко освоюється поняття формальної змінної. *Формальна змінна - це змінна, якій при виклику файлу-функції привласнюєтьс значення «викликаючої» змінної.* У прикладі при виклику `g = Ra(F)` формальній змінній *s* привласнюється значення *F*, а при виклику `g1=Ra(g)` – значення *g*. У першому випадку всі операції, які виконуються над s, будуть фактично виконуватися над *F*, а в другому випадку всі операції, які виконуються над *s*, будуть фактично виконуватися над *g*.

Звернемо увагу на наступні властивості файлу-функції.

1. Файл-функція може використовуватися в математичних вираженнях Script-файлу у форматі `name(аргументи)`.

Наприклад, у рядках 4 і 5 розглянутого вище Script-файлу PasSigma оператори мають вигляд: `g=Ra(F)`, `g1=Ra(g)`.

2. Всі змінні, наявні в тілі файлу-функції, є локальними, тобто діють тільки в межах цього файлу-функції. Проте, правила «гарного тону» все-таки не рекомендують використовувати в Script-файлі і файлі-функції однакові імена змінних.

Наведена форма файлу-функції характерна для функції з одним вихідним параметром. Якщо вихідних параметрів більше, то вони вказуються у квадратних дужках після оператора `function`. При цьому структура файлу-функції буде мати такий вигляд:

```
 % Список вхідних параметрів:
function[var1,var2,...]=name_f
 %Основний коментар
 %Додатковий коментар
var1=вираження; %Тіло файлу з будь-якими
                % вираженнями
var2=вираження;
… … … … … … … … … … … … … … … … … … … … … … … … …
varn=вираження
```

Такий файл-функцію не можна використовувати безпосередньо в математичних висловлюваннях, оскільки він повертає не єдиний результат, а безліч результатів - по числу вихідних параметрів. Тому дана функція використовується в Script-файлі як окремий елемент програми виду

```
[var1,var2,...]=name_f(Список вхідних параметрів)
```

Після його застосування змінні виходу `var1, var2,..` стають визначеними і їх можна використовувати в наступних математичних висловлюваннях та інших сегментах програми. Якщо функція використовується в Script-файлі у форматі `name_f(Список_параметрів)`, то повертається значення тільки першого вихідного параметра - змінної `var1`.

Приклад 5.1. Визначимо значення двох змінних *var*1 та *var*2, якщо

$$var1 = a^2 + 2b,\ var2 = -\sqrt{a} - 2b.$$

Script-файл Naraz1:

```
1. %Два вихідних параметри
2. %Особливості використання file-function
3.a=2; b=3;
4.[var1,var2]=Naraz2(a,b)
5.Y=Naraz2(a,b)
```

Файл-функція Naraz2(s1,s2)

```
function[var1,var2]=Naraz2(s1,s2)
 %Два вихідних параметри
var1=s1^2+2*s2;
var2=-sqrt(s1)-2*s2;
```

Після виконання Script-файлу в командному вікні з'явиться наступна відповідь:

```
var1 =
    10
var2 =
   -7.4142
Y =
    10
```

MATLAB дозволяє вводити в програмі глобальні змінні. Для цього в Script-файлі та файлі-функції повинна бути використана інструкція:

```
global (змінні)
```

Приклад 5.2. Введемо у файли з попереднього прикладу команди для визначення змінної $x = -a + 20b$.

Script-файл Naraz1

```
1. %global variable
2.global d;
3.a=2; b=3;
4.[var1,var2]=Naraz2(a,b)
5.Y=Naraz2(a,b)
6.x=d
```

Файл-функція Naraz2(s1,s2)

```
function[var1,var2]=Naraz2(s1,s2)
 %global
d=-s1+20*s2;
var1=s1^2+2*s2;
var2=-sqrt(s1)-2*s2;
```

Після виконання Script-файлу в **Command Window** з'явиться відповідь:

```
var1 =
    10
var2 =
   -7.4142
Y =
    10
d =
    58
```

У функції системи MATLAB можна включати підфункції. Вони оголошуються і записуються в тілі основних функцій і мають ідентичну їм конструкцію. Підфункції визначені та діють локально, тобто в межах відповідного файлу-функції. Якщо функції і підфункції повинні використовувати загальні змінні, їх потрібно оголосити глобальними як у файлі-функції, так і у підфункції.

І останнє зауваження. Оскільки робота в **Command Window** вкрай важка через незручність виправлення замічених помилок, можна рекомендувати всю програму рішення якої-небудь задачі реалізовувати, використовуючи тільки Script-файл і файл-функцію.

Завдання для самостійної роботи

1. На рис. 5.6 надана система маса-пружина з демпфуванням, яка може служити моделлю автомобільного амортизатора. Тут M – маса вантажу, b – коефіцієнт тертя, c – коефіцієнт жорсткості пружини, $f(t)$ – зовнішня сила. Коливання маси за відсутності зовнішньої сили $f(t)$ описується вираженням

$$y(t) = \frac{y_0}{\sqrt{1-\zeta^2}} \cdot e^{-\zeta\omega_0 t} \sin(\omega_0 \sqrt{1-\zeta^2}\, t + \arccos\zeta, \qquad (5.1)$$

де y_0 – початкове відхилення ω_0 – власна частота коливань системи, $\omega_0 = \sqrt{c/M}$, ζ - безрозмірний коефіцієнт загасання (демпфування). Проаналізуйте вплив коефіцієнту жорсткості пружини c на коливання системи. Для цього побудуйте в єдиному вікні графіки коливань системи як реакцію на початкове відхилення y_0 при наступних параметрах

а) $c = 1$; $M = 1$, $b = 1$;
б) $c = 2$; $M = 1$, $b = 1$;
в) $c = 5$; $M = 1$, $b = 1$.

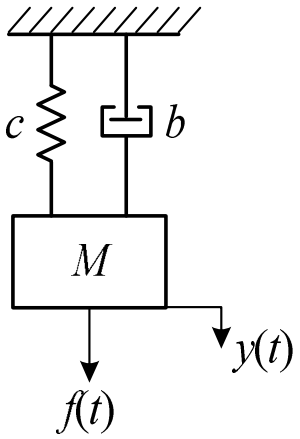


Рис. 5.6. Система маса-пружина з демпфуванням

Завдання значень c, M і b слід здійснювати в командному рядку, а останні розрахунки і команди побудови графіків помістити в Script-файлі. Час моделювання t змінюється від 0 до 10 секунд із шагом 0,1 секунда. Оскільки другий і третій множники в рівнянні (5.1) є векторами, то для їх поелементного множення необхідно використовувати операцію множення з точкою (.*).

2. Для графіків, що булі побудовані при виконанні попереднього завдання, встановіть наступні параметри відображення ліній

а) при $c = 1$: колір графіку – зелений тип лінії – сполошна без маркера, ширина лінії – 1,5;

б) при $c = 2$: колір графіку - червоний, тип лінії – пунктир з маркером у вигляді символу «х», ширина лінії – 2;

в) при $c = 5$: колір графіку - чорний, тип лінії – штрих-пунктир без маркера, ширина лінії – 3.

3. Виконаєте завдання 1 за допомогою файлу-функції.

6. КЕРУЮЧІ СТРУКТУРИ

Крім програм з *лінійною* структурою, інструкції яких виконуються строго одна за одною, існує безліч програм, структура яких *нелінійна*. Вітки в таких програмах можуть виконуватися залежно від певних умов, іноді з кінцевим числом повторень - циклів, іноді у вигляді циклів, що завершуються при виконанні заданої умови. Практично будь-яка серйозна програма має нелінійну структуру. Для створення таких програм необхідні спеціальні керуючі структури. Вони є в будь-якій мові програмування і, зокрема, в MATLAB.

6.1 Умовні оператори

Умовний оператор `if` (якщо) у найпростішому варіанті має конструкцію:

```
if умова_інструкції end
```

Наприклад

```
if x>3      y=x^2, end
   умова    інструкція
```

Поки умова виконується, оператор `if` повертає логічне значення 1 («істина») і виконуються інструкції, що становлять тіло структури `if…end`**.** При цьому оператор `end` вказує на кінець переліку інструкцій. Інструкції в списку розділяються оператором «`,`» (кома) «`;`» (крапка з комою) або натисканням клавіші <Enter>. Якщо умова не виконується (дає логічне значення 0, «неправда»), то інструкції також не виконуються.

Умови записуються у вигляді:

```
вираження_1 оператор_відносини  вираження_2
```

Причому у якості оператору відносини використовуються наступні оператори:

== – дорівнює;

< – більше;

> - менше;

<= – більше або дорівнює;

>= – менше або дорівнює;

~= - не дорівнює.

Всі ці оператори являють собою пари символів без пробілу між ними.

Розглянемо ще приклад:

```
>> a=3;
>> x=4;
>> if  x>=3  a=2, end
a =
     2
```

Розповсюдженою конструкцією умовного циклу є наступна:

```
if  Умова інструкції_1,
else Інструкція_2, end
```

Приклад:

```
>> if x>3, a=2, b=sqrt(a), else a=5, b=a^2, end
a =
     2
b =
    1.4142
```

Є одна неприємна деталь при використанні умов операторів. Вони не працюють у конструкції побудови графіків. Наприклад.

Основна програма	М-файл-функція
>>x=1:0.1:2;	Function fa=lek6(r)
>>y=lek(6)	if r<1.51 a=sin(r),else a=-r, end
>>plot(x,y)	fa=a

Всі значення *y* визначаються відповідно до умови `else`. Умова $x < 1.51$ ігнорується.

Для того щоб побудувати графік залежності $y=y(x)$ можна зробити так:

- визначити вектор *X*:

```
>>X=[1 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0];
```

- визначити вектор *Y*:

```
>>Y=[lek6(1)   lek6(1.1)   lek6(1.2)   lek6(1.3)
lek6(1.4)  lek6(1.5)  lek6(1.6)  lek6(1.7)  lek6(1.8)
lek6(1.9) lek6(2.0)];
```

- побудувати графік:

```
>>plot(X,Y)
```

Результатом виконання останньої команди є графік, представлений на рис. 6.1.

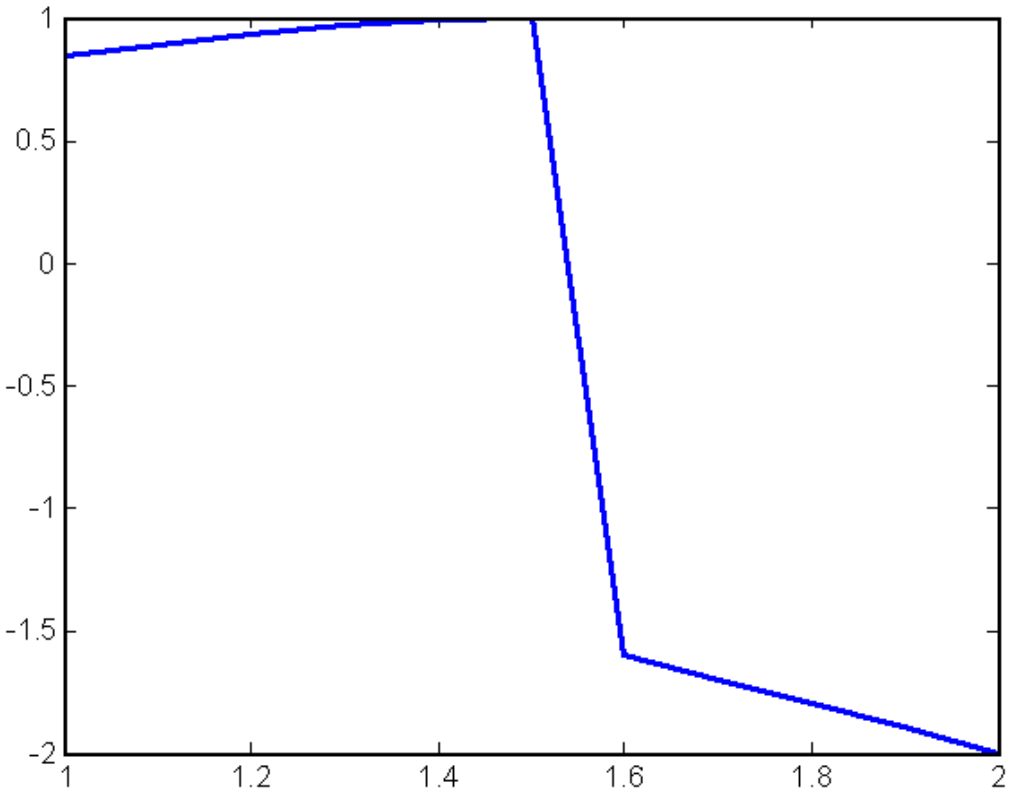


Рис. 6.1. Графік залежності `y=y(x)`

6.2 Цикли типу while...end

Оператор `while` (англ. *while* – «поки»), дозволяє організовувати більш складні цикли. Цикл виконується доти, поки виконується умова. Конструкція такого циклу має вигляд:

```
while Умова
Інструкція
end
```

Приклад. Розглядається балка (рис. 6.2) із прикладеною силою *F*(Н).

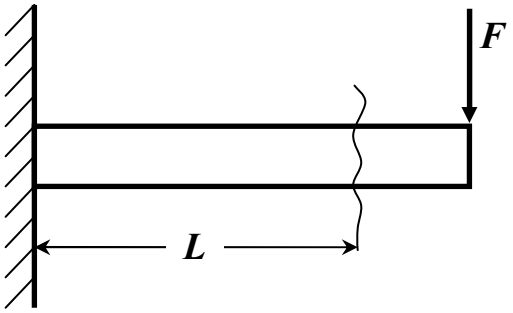


Рис. 6.2. Балка з діючої на неї силою

Знайдемо довжину *L* прикладання сили *F*, при якій момент *M* (Нм) у місці закладення балки перевищить 14750 (Нм); *F* = 9675 (Н). *M* = *F L*.

```
>>F=9675;  %Діюча сила, Ньютони
>>L=0;     %Довжина прикладання сили
>>M=0;     %Початкове значення моменту
>>while M<14750;
L=L+0.01;
M=F*L;
end
>> disp(L)
    1.5300
```

Вирішимо цю ж задачу із залученням апарата М-файлів-функцій. Складемо спочатку М- функцію.

```
1  %Балка.Сила
2  %Макимальна припустима відстань
3 function Mome=oprmome(sila,rasst)%Без «;»
4 Mome=sila*rasst;
```

Перейдемо тепер у командне вікно:

```
>>F=9675;
>>L=0;
>>M=0;
>>while M<14750;
L=L+0.01;
M=oprmoment(F,L);
end
>>disp(L)
    1.5300
```

6.3 Цикли типу for...end

Цикли типу `for…end` використовуються для організації обчислень із заданим числом повторюваних циклів. Їх ще називають циклами з параметром. Конструкція такого циклу має вигляд:

```
for var=вираження
інструкція;
end
```

Тут `var` – ім'я параметра змінної `(X, Y, lambda,…)`; вираження найчастіше записується у вигляді:

```
var(min): d: var(max)
```

де `var(min)` – мінімальне значення параметра; `d` - шаг зміни, якщо шаг не вказується, то за замовчуванням `d=1`; `var(max)` – максимальне значення параметра.

Наступний приклад пояснює застосування циклу для одержання значень $y=y(x)$ при ряді значень аргументу x.

М-файл-функція:

```
1 function fa=lekt6_3(r)
2 A=r.^2;
3 B=sin(a);
4 Z=cos(b)+0.1*exp(b);
5 fa=z;
```

Основна програма.

```
>> for k=1:10,
x(k)=0.1*k;
y(k)=lek6_3(x(k));disp(y(k));plot(x,y);end
    1.1010
    1.1033
    1.1054
    1.1046
    1.0976
    1.0808
    1.0514
    1.0086
    0.9553
    0.8983
```

Дана програма будує графік, представлений на рис. 6.3.

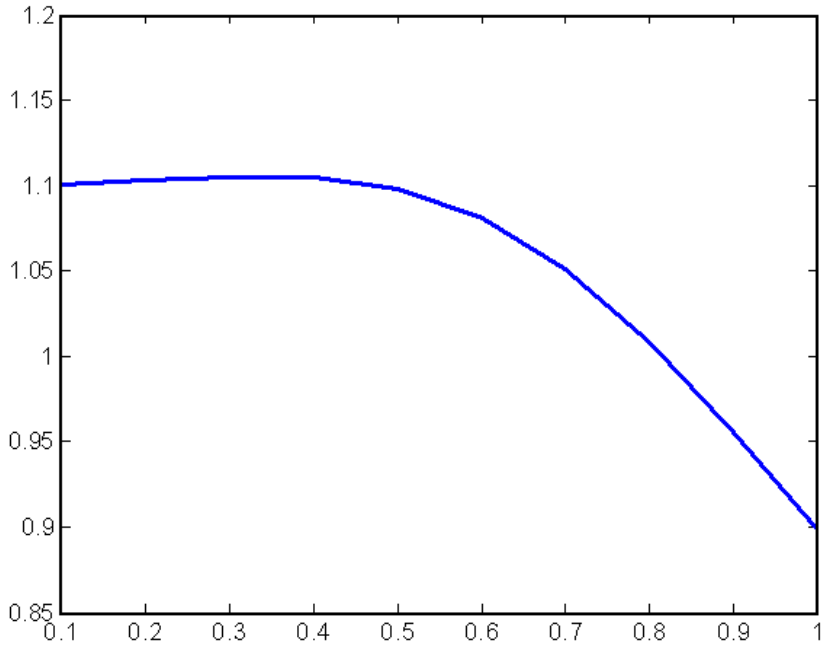


Рис. 6.3. Приклад на використання циклу `for…end`

Розглянемо тепер ще одну можливість панелі інструментів графічного вікна, про яку ми не згадували в розділі 4. Виберемо операцію **Basic Fitting** у меню **Tools**. При цьому з'явиться діалогове вікно, представлене на рис. 6.4. Це вікно дозволяє виконувати апроксимацію та інтерполяцію отриманих одномірних даних. Підберемо апроксимуючу криву 4-го порядку:

$$Y(x) = 0{,}64x^4 - 1{,}6x^3 + 0{,}97x^2 - 0{,}18x + 1{,}1$$

при x_0=0,1.

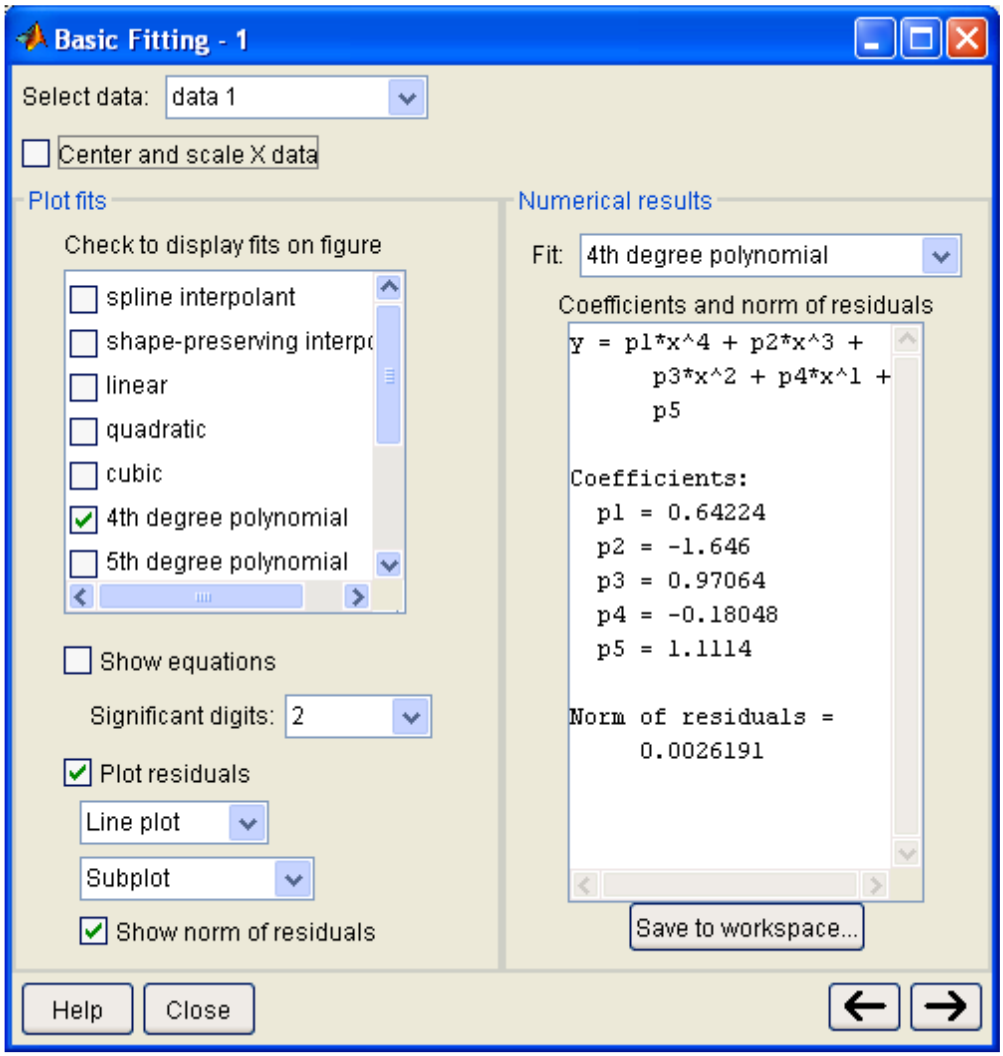


Рис. 6.4. Діалогове вікно **Basic Fitting**

Завдання для самостійної роботи

1. Виконайте завдання 5.1 за допомогою циклу `for…end`. Коефіцієнт жорсткості пружини *c* повинен змінюватися від 0,2 до 1 з шагом 0,2.

2. Побудуйте в одному графічному вікні сімейство кривих $f(x,a) = e^{-ax}\cdot\sin x$, якщо $x\in[0, 2\pi]$, а значення параметра *a* змінюються від -0,1 до 0,1.

3. Побудуйте графік функції $f(x) = e^{-0{,}1x}$, $x\in[-10, 10]$. Апроксимуйте отриману криву поліномом 4-го порядку.

7. ВЕКТОРИ ТА МАТРИЦІ

7.1 Загальні положення

MATLAB - це система, що спеціально створена для роботи з матрицями, а вектори і скаляри трактуються як частні види матриць. Розглянемо приклад найпростішої програми:

```
>> x=0.6570;
>>x^2
```

З вікон **Workspace** або **Array Editor** (рис. 7.1) добре видно, що змінна *x* і непойменована змінна, що дорівнює квадрату *x* та одержала ім'я `ans`, трактуються як матриці розміром 1×1.

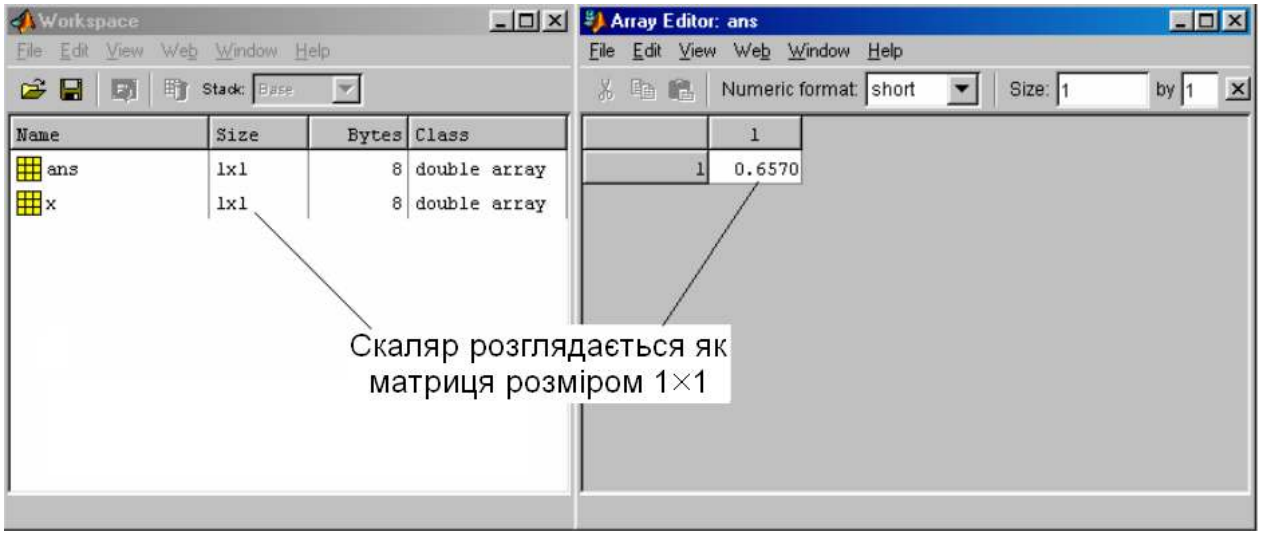


Рис. 7.1. Вікна **Workspace** і **Array Editor**

Використання матриць як основних обчислювальних одиниць дозволяє різко скоротити обсяг програм, у яких виконуються операції з векторами і матрицями, у порівнянні із програмами, написаними на інших мовах.

7.2 Введення векторів і матриць

Опишемо правила введення матриць. Матриця звичайно записується у квадратні дужки [ ] і формується з окремих

елементів. При цьому елементи стовпця відокремлюються пробілами або комами, а рядки розділяються крапками з комою або натисканням клавіші <Enter>. Розмірність матриці вказувати не потрібно, вона визначається автоматично. Припустимо, наприклад, що нам треба ввести матрицю **A**:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 4 & -1 \end{bmatrix}.$$

Можливі два варіанта введення, що принципово не відрізняються.

1. `>> A=[1,2;4,-1];`

Тут всі елементи матриці введені єдиним рядком. Елементи кожного рядка відокремлюються друг від друга комою (можливе використання пробілу). Рядки матриці відокремлюються один від одного крапкою з комою.

2. `>> A=[1 2;<Enter>`
`4 -1];`

Такий вид дає краще візуальне сприйняття матриці.

Для контролю введеної матриці її можна вивести на екран:

```
>> A        <Enter>
    1       2
    4      -1
```

Елементи матриці можна вводити у вигляді арифметичних виражень, що містять будь-які доступні у MATLAB функції. Наприклад,

```
>> M=[2+cos(pi/3) exp(-2);1 sin(pi/8)]
M =
    2.5000    0.1353
    1.0000    0.3827
```

Часто матриця **A** формується за результатами обчислень її елементів у циклі. При цьому можлива вихідна індексація $A(k,n)$ її елементів.

Розглянемо приклад. Нехай необхідно сформувати матрицю **A** розміром 5×5. Її (k,n)-ий елемент $A(k,n)$ дорівнює сумі квадратів (k^2+n^2). Але якщо сума ($k+n$)>3, тоді (k,n)-ий елемент $A(k,n)$ дорівнює $(k^2+n^2)-(k+n)$.

Відкриваємо **Command Window**. Потім відкриваємо M-file функцію, у якій будуть розраховуватися елементи $A(k,n)$. Дамо їй ім'я *fmatr*:

```
1 function a=fmatr(r,s)
2 % матриця
3 %сума квадратів з корекцією
4 d=0;
5 if (r+s)>3 d=d+s;
6    else d=0; end
7    a=r^2+s^2-d;
```

Тепер створимо M-file-функцію, який дамо ім'я *matr*:

```
1 % Формування матриці
2 % Цикли
3 for k=1:5
4    for n=1:5
5        A(k,n)=fmatr(k,n); %Елемент матриці
6                           %має індекси k,n
7    end;
8 end;
9 disp(A)
```

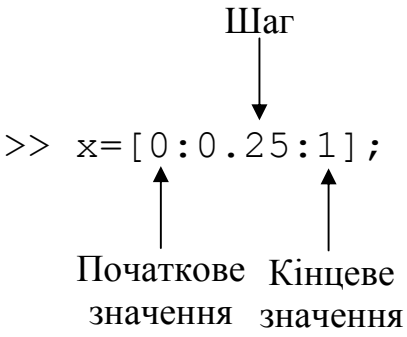
Запускаємо останній Script File кнопкою «Save and Run»  на панелі інструментів, при цьому в **Command Window** з'явиться матриця **A**:

```
>>    2     5     7    13    21
      5     6    10    16    24
      9    11    15    21    29
     16    18    22    28    36
     25    27    31    37    45
```

Вектор-рядок з N елементами MATLAB сприймає як матрицю розміром $1\times N$, тому якщо треба задати, наприклад, вектор **V** з елементами [5 3 2 1 0], то досить набрати наступну команду:

```
>> V=[5  4  3  2  1]      <Enter>
V =
     5     4     3     2     1
```

Дуже часто доводиться працювати з векторами, значення елементів яких є арифметичною прогресією. MATLAB дає можливість спрощеного введення вектора, компонентами якого є числа, що починаються зі значення x_1, закінчуються значенням x_N і йдуть слідом один за одним із заданим шагом *dx*. Для цього використовується символ двокрапка (:). Наприклад:



Якщо шаг не заданий, то він приймає значення 1.

Вектор-стовпець вводиться аналогічно вектору-рядку, але значення елементів відокремлюються знаком крапка з комою (;):

```
>> B=[1;3;5;7;9]
B =
     1
     3
     5
     7
     9
```

Про кількість елементів в одномірному масиві завжди можна довідатися за допомогою функції length:

```
>> length(B)
ans =
     5
```

7.3 Створення матриць зі специфічними властивостями

MATLAB має дуже багато вбудованих функцій для створення матриць із якими-небудь особливими властивостями. Розглянемо деякі з них.

Так, для створення матриці розміром $m \times n$, всі елементи якої - одиниці, використовується функція `ones(m,n)`:

```
>> ones(3,2)          <Enter>
ans =
     1     1
     1     1
     1     1
```

Якщо необхідно ввести матрицю з одиничними елементами розміром $n \times n$, то досить ввести `ones(n)`:

```
>> ones(3)
ans =
     1     1     1
     1     1     1
     1     1     1
```

Для створення матриці з одиничними елементами такого ж розміру, як і матриця, що задана раніше, використовується запис `ones(size(A))`. Наприклад, якщо раніше була задана матриця **A** розміром 5×5, то одержимо:

```
>> ones(size(A))
ans=
    1     1     1     1     1
    1     1     1     1     1
    1     1     1     1     1
    1     1     1     1     1
    1     1     1     1     1
```

Подібний запис рівною мірою справедливий і для інших функцій, розглянутих нижче.

Для створення матриці розміром $n \times n$, у якої всі діагональні елементи дорівнюють одиниці, а інші – нулю використовується функція `eye(n)` (читається [аі]):

```
>> E=eye(3)
E =
     1     0     0
     0     1     0
     0     0     1
```

Таку матрицю називають *одиничною*.

Функція `zeros(m,n)` (або `zeros(n)`) створює матрицю з нульовими елементами:

```
>> zeros(3,4)
ans =
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

Функція `rand(m,n)` створює матрицю розміром $m \times n$ з випадкових чисел, рівномірно розподілених у діапазоні від 0 до 1, а функція `randn(m,n)` - створює матрицю розміром $m \times n$ з випадкових чисел, розподілених за нормальним законом з нульовим математичним очікуванням і середнєквадратичним відхиленням, рівним одиниці.

Для того, що вказати на окремий елемент матриці використовується інструкція `M(n,k)`. Наприклад, якщо матриця $\boldsymbol{M}$ має вигляд

$$\mathbf{M} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix},$$

то інструкція

```
>> M(2,3)
```

дасть результат:

```
ans =
     6
```

Якщо потрібно привласнити елементу *M*(2,1), який у вихідній матриці дорівнює 4, значення 10, треба в командному рядку набрати

```
>> M(2,1)=10;
```

Саме собою зрозуміло, що привласнювати нове значення можна будь-якому елементу матриці та це нове значення може бути будь-яким.

Робота з окремим елементом вектора здійснюється за допомогою інструкції `B(k)`, де *k* – місце елемента у векторі.

7.4 Операції з матрицями

До основних матричних операцій відносяться операції, що представлені в таблиці 7.1.

Таблиця 7.1

Оператори для операцій з матрицями

+	Додавання
-	Віднімання
*	Множення
/	Праве ділення
\	Ліве ділення
'	Транспонування
^	Зведення в ступінь
inv(A)	Обернення матриці
.*	Поелементне множення
./	Поелементне праве ділення
.\	Поелементне ліве ділення
.'	Поелементне транспонування
.^	Поелементне зведення в ступінь

Нагадаємо зміст всіх наведених операцій.

1. Додавання і віднімання матриць.

Сумою (різницею) двох матриць **A** і **B** є матриця **C**, елементи якої визначаються сумами (різницями) відповідних елементів

матриць **A** і **B**. При додаванні та відніманні матриць вони повинні мати однакову розмірність.

Приклад:

```
>> A=[1 2 4; -5 1 -6]; B=[3 -1 4; 1 -1 2];
>> C=A+B

C =

     4     1     8
    -4     0    -4
```

2. Множення матриць.

Матрицю **A** можна помножити на матрицю **B**, тільки якщо число стовпців матриці **A** дорівнює числу рядків матриці **B**. Результатом множення матриці **A** розміром ($n\times m$) на матрицю **B** розміром ($m\times k$) буде матриця **C** розміром ($n\times k$), елементи якої визначаються формулою

$$C(i,j)=\sum_{r=1}^{m}\left(a_{ir}\cdot b_{rj}\right),\quad i=1,n;\ \ j=1,k\,. \tag{7.1}$$

Наприклад.

```
>> A=[1 2 4;-5 1 -6];
>> B=[2 -2 0 1;6 -1 5 4;2 1 -3 -1];
>> C=A*B

C =

    22     0    -2     5
   -16     3    23     5
```

3. Обернення матриці, тобто визначення зворотної матриці.

Зворотною матрицею $\mathbf{A}^{-1}$ матриці **A** називається матриця, застосування якої ліворуч або праворуч до матриці **A** дає одиничну матрицю **E**:

$$\mathbf{A}^{-1}\cdot\mathbf{A}=\mathbf{A}\cdot\mathbf{A}^{-1}=\mathbf{E}. \tag{7.2}$$

Обернення матриці **A** виконується за допомогою функції `inv(A)`. Наприклад.

```
>> A=[ 1  2 ; 3  4 ];
>> inv(A)
ans =
  -2.0000   1.0000
   1.5000  -0.5000
>> A*inv(A)
ans =
   1.0000         0
   0.0000   1.0000
>> inv(A)*A
ans =
   1.0000         0
   0.0000   1.0000
```

4. Праве ділення.

Якщо матриця **C** є результатом множення матриці **A** на квадратну матрицю **B**: $\mathbf{C} = \mathbf{A}\cdot\mathbf{B}$, то для визначення матриці **A** (саме матриці **A**!) потрібно застосувати матриці **C** и $\mathbf{A}\cdot\mathbf{B}$ ліворуч до матриці $\mathbf{B}^{-1}$:

$$\mathbf{C}\cdot\mathbf{B}^{-1} = \mathbf{A}\cdot\mathbf{B}\cdot\mathbf{B}^{-1}. \qquad (7.3)$$

Добуток $\mathbf{B}\cdot\mathbf{B}^{-1}$ дорівнює одиничній матриці **E.** Отже,

$$\mathbf{A} = \mathbf{C}\cdot\mathbf{B}^{-1}, \qquad (7.4)$$

або з використанням символу правого ділення:

$$\mathbf{A} = \mathbf{C}/\mathbf{B}. \qquad (7.5)$$

Праве ділення матриці **C** на квадратну матрицю **B**, тобто **C**/**B**, визначає матрицю **A**, добуток якої на матрицю **B**, визначає матрицю **C**.

Загальне правило можливості застосування правого ділення полягає в тому, що праве ділення можна застосовувати тільки при квадратній невираженій матриці **B** і матриці **C**, що має число стовпців, які збігаються з розмірністю матриці **B.**

Приклад.

```
>> A=[1 2; 3 4];
>> B=[5 6;7 8];
```

```
>> C=A*B
C =

    19   22
    43   50
>> C/B
ans =
    1.0000   2.0000
    3.0000   4.0000
```

5. Ліве ділення.

Якщо матриця **C** є добутком квадратної матриці **A** на матрицю **B**: $\mathbf{C} = \mathbf{A}\cdot\mathbf{B}$, то для визначення матриці **B** (саме **B**!) необхідно до матриці **C** и $\mathbf{A}\cdot\mathbf{B}$ застосувати зліва матрицю $\mathbf{A}^{-1}$:

$$\mathbf{A}^{-1}\cdot\mathbf{C} = \mathbf{A}^{-1}\cdot\mathbf{A}\cdot\mathbf{B}. \qquad (7.6)$$

Добуток $\mathbf{A}^{-1}\cdot\mathbf{A}$ дорівнює одиничній матриці **E.** Тому

$$\mathbf{B} = \mathbf{A}^{-1}\cdot\mathbf{C}, \qquad (7.7)$$

або з використанням символу лівого ділення

$$\mathbf{B} = \mathbf{A}/\mathbf{C}. \qquad (7.8)$$

Ліве ділення матриці **C** на квадратну матрицю **A**, тобто **A\C**, визначає матрицю **B** таку, що $\mathbf{A}\cdot\mathbf{B} = \mathbf{C}$.

Загальне правило можливості застосування лівого ділення полягає в тому, що його можна застосовувати тільки при квадратній невираженій матриці **A** и матриці **C**, що має число рядків, яке збігається з розмірністю матриці **A.**

Приклад.

```
>> A=[ 1 2 3 ;4 5 4; 5 7 6 ];
>> C=[3;8;6];
>> B=A\C

B =
   12.0000
  -12.0000
    5.0000
```

За допомогою оператора лівого ділення « \ » зручно розв'язувати системи рівнянь. Наведемо приклад. Нехай необхідно розв'язати систему двох лінійних рівнянь:

$$\begin{cases} a_{11} \cdot x_1 + a_{12} \cdot x_2 = b_1 \ ; \\ a_{21} \cdot x_1 + a_{22} \cdot x_2 = b_2 \ . \end{cases} \tag{7.9}$$

або в матричній формі:

$$\mathbf{A} \cdot x = \mathbf{B}. \tag{7.10}$$

де

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}.$$

Для розв'язання системи лінійних алгебраїчних рівнянь використовуються або метод Крамера, або метод Гауса. Скористаємося методом Гауса. Для цього помножимо друге рівняння на коефіцієнт $-a_{11}/a_{21}$ і складемо з першим рівнянням вхідної системи. Одержимо нове рівняння:

$$\left(a_{11} - a_{21} \cdot \frac{a_{11}}{a_{21}} \right) \cdot x_1 + \left(a_{12} - a_{22} \cdot \frac{a_{11}}{a_{21}} \right) \cdot x_2 = b_1 - b_2 \cdot \frac{a_{11}}{a_{21}}. \tag{7.11}$$

Ця операція виконана для того, щоб скоротити число невідомих у другому рівнянні. Дійсно, коефіцієнт $\left(a_{11} - a_{21} \cdot a_{11}/a_{21}\right)$ дорівнює нулю і вся система рівнянь прийме вид:

$$\begin{cases} a_{11} \cdot x_1 + a_{12} \cdot x_2 = b_1 \ ; \\ 0 \ \cdot x_1 + \left(a_{12} - a_{22} \cdot \dfrac{a_{11}}{a_{21}} \right) \cdot x_2 = b_1 - b_2 \cdot \dfrac{a_{11}}{a_{21}} \ . \end{cases} \tag{7.12}$$

Із другого рівняння цієї системи одержимо:

$$x_2 = \frac{b_1 - b_2 \cdot \dfrac{a_{11}}{a_{21}}}{a_{12} - a_{22} \cdot \dfrac{a_{11}}{a_{21}}}, \qquad (7.13)$$

а з першого рівняння:

$$x_1 = \frac{b_1 - a_{12} \cdot x_2}{a_{11}} = \left(b_1 - a_{12} \cdot \frac{b_1 - b_2 \cdot \dfrac{a_{11}}{a_{21}}}{a_{12} - a_{22} \cdot \dfrac{a_{11}}{a_{21}}} \right) \cdot \frac{1}{a_{11}}. \qquad (7.14)$$

Наприклад:

$$\begin{cases} 2x_1 + 3x_2 = 8\,; \\ -x_1 + x_2 = 1\,. \end{cases}$$

$$x_1 = 1\ ; \quad x_2 = \frac{8 - 1 \cdot 2/(-1)}{3 - 1 \cdot 2/(-1)} = 2.$$

В MATLAB розв'язання системи лінійних алгебраїчних рівнянь здійснюється дуже просто:

```
>> A=[2 3;-1 1]; %Введення матриці A
>> B=[8;1];      % Введення вектора-стовпця B
>> x=A\B
x =
   1
   2
```

Як бачимо, зменшення обсягу програми і часу, витраченого на її складання дійсно істотно. Додамо до цього, що користувач у принципі звільняється від знання методу Гауса і тонкостей розв'язання системи лінійних алгебраїчних рівнянь.

6. Транспонування матриці.

Транспонування матриці, тобто заміна рядків стовпцями (або навпаки) позначається символом апострофа (').

Приклад.

```
>> A=[1 2 4;-5 1 -6];
>> A'
ans =
     1    -5
     2     1
     4    -6
```

7. Зведення в ступінь.

Зведення в ступінь в MATLAB здійснюється за допомогою знака (^): `A^n`. При цьому *n* повинне бути цілим числом, оскільки вказує, скільки разів матриця буде помножена сама на себе.

7.5 Визначення характеристик матриці

1. Обчислення визначника (детермінанта) матриці.

Обчислення визначника матриці виконується за допомогою функції `det(A)`.

Приклад.

```
>> A=[5 -4 3;3 2 1;10 1 -2];
>> det(A)
ans =
  -140
```

2. Визначення власних векторів і власних чисел квадратної матриці.

Власним вектором квадратної матриці **A** називається вектор **X**, перетворення **A**·**X** якого матрицею **A** дає вектор λ**X** того ж напрямку, що і **X.** Коефіцієнт λ називається власним числом матриці.

Власні числа визначаються за допомогою функції `eig(A)` (від англійського *eigen value* - власне число матриці). Наприклад.

```
>> A=[1 2;3 4];
>> eig(A)
ans =
   -0.3723
    5.3723
```

Спільне визначення власних векторів **V** і власних чисел *D* виконується за допомогою функції

```
>> [V,D]=eig(A)
```

Для використаної вище матриці **A**

```
V =
   -0.8246   -0.4160
    0.5658   -0.9094

D =
   -0.3723         0
         0    5.3723
```

Перший власний вектор **V**(1) має складові (-0,8246; 0,5658)′ і власне число *D*(1) = -0,3723. Другий власний вектор **V**(2) має складові (-0,4160; -0,9094)′ і власне число *D*(2) = 5,3723.

Перевірка.

$$\mathbf{A}\cdot\mathbf{V} = D\cdot\mathbf{V}.$$

1) $1\cdot(-0{,}8246) + 2\cdot(0{,}5658) = -0{,}3723\cdot(-0.8246)$,
$0{,}307 = 0{,}307$

2) $3\cdot(-0{,}8246) + 4\cdot(0{,}5658) = -0{,}3723\cdot(0{,}5658)$,
$0{,}2106 = 0{,}2106$

Завдання для самостійної роботи

1. Змінна *x* змінюється від 0 до 15 з одиничним шагом. За допомогою операторів циклу `for…end` і умови `if…end` сформуйте вектор **A**, що складається з парних *x* і вектор **B**, що складається з непарних *x*. При виконанні завдання зручно користуватися командою `rem(X,Y)`, що повертає залишок ділення *X* на *Y*.

2. Змінна *x* змінюється від 0 до 15 з одиничним шагом. За допомогою операторів циклу `while…end` сформуйте квадратну

матрицю **A**, головна діагональ якої заповнена числами від 1 до 10, а всі інші елементи дорівнюють 0.

3. В матриці **A** з попереднього завдання комірки з нульовими елементами заповніть значеннями *i+j*, де *i* та *j* – номери поточного рядка і стовпця відповідно.

4. Розглянете матрицю **A** з попереднього завдання.

а) Сформуйте вектор *Y*, що містить суму елементів рядків.

б) Знайдіть максимальну суму елементів кожного рядка матриці **A**.

в) Розділіть рядок, в якому знаходиться максимальна сума, на перший рядок поелементно.

5. Дана функція $y(x)$:

$$y = \begin{cases} \dfrac{1+x^2}{\sqrt{1+x^4}}, x \le 0 \\ 2x + \dfrac{\sin^2 x}{2+x}, x > 0. \end{cases}$$

а) Знайдіть значення цієї функції в діапазоні x=[-12;12]. Результат запишіть у вигляді матриці 5×5.

б) Знайдіть максимальне число матриці.

в) Знайдіть кількість елементів, які більше нуля.

г) Знайдіть кількість елементів, які менше нуля.

д) Знайдіть суму елементів матриці.

е) Знайдіть суму елементів головної діагоналі матриці.

ж) Знайдіть середнє арифметичне елементів матриці.

з) Знайдіть середнє арифметичне елементів головної діагоналі матриці.

і) Знайдіть номер строки, яка містить 0.

к) Знайдіть номер рядка, який містить 0.

л) Виведіть значення елементу матриці з третьої строки та п’ятого стовпця.

ЧАСТИНА II
CONTROL SYSTEM TOOLBOX

8. ЗАВДАННЯ МОДЕЛЕЙ СИСТЕМ КЕРУВАННЯ

8.1 Загальні положення

Control System Toolbox (комплект інструментів систем керування), що входить до складу MATLAB, пропонує великі інструментальні засоби для аналізу і синтезу лінійних стаціонарних (linear time-invariant - LTI) систем керування. При цьому підтримуються як безперервні так і дискретні системи, які можуть бути як одномірними (single-input/single-output - SISO) так і багатомірними (multiple-input/multiple-output - MIMO).

Control System Toolbox дозволяє задавати і аналізувати лінійні моделі в чотирьох формах:

- за допомогою передаточних функцій (transfer function - TF);
- за допомогою полюсів, нулів і коефіцієнта підсилення моделі (zero-pole-gain - ZPG).
- у просторі станів (state-space - SS);
- у вигляді частотних характеристик (frequency response data - FRD);

При описі систем передаточними функціями використовуються такі функції MATLAB як `roots`, `tf`, `series`, `parallel`, `feedback`, `pole`, `zero`, `poly`, `conv`, `polyval`, `minreal`, `pzmap`, `step`.

8.2. Робота з поліномами

Розглянемо, як MATLAB дозволяє досліджувати системи, що описуються передаточними функціями. Оскільки передаточна функція представляє собою відношення двох поліномів, ми спочатку розглянемо, як MATLAB оперує з алгебраїчними поліномами. При цьому не будемо забувати, що в передаточній функції повинні бути задані обидва поліноми - і у чисельнику, і в знаменнику.

Поліноми в MATLAB представляються у вигляді векторів-рядків, що складаються з коефіцієнтів в убутному порядку ступенів. Наприклад, поліном $p(s) = s^3 + 3s^2 + 4$ задається в такий спосіб.

```
>> p=[1 3 0 4]; ←————— p(s) = s^3 + 3s^2 + 4
```

Зверніть увагу, що навіть якщо коефіцієнт при якомусь ступені дорівнює нулю, він однаково враховується при вводі полінома $p(s)$.

Якщо **p** є вектор-рядок, що складається з коефіцієнтів $p(s)$ у порядку убування ступенів, то функція `roots(p)` визначає вектор-стовпець, що містить корені цього полінома. І навпаки, якщо **r** - вектор-стовпець, що містить корені полінома, то функція `poly(r)` дає вектор-рядок з коефіцієнтів полінома в порядку убування ступенів. Так, корені полінома $p(s) = s^3 + 3s^2 + 4$ можна обчислити таким чином:

```
>> r=roots(p) ←————— Обчислення коренів p(s) = 0
r =
  -3.3553
   0.1777 + 1.0773i
   0.1777 - 1.0773i
```

Поліном можна відновити за його коренями за допомогою функції `poly`:

```
>> p=poly(r) ←————— Відновлення полінома за його коренями
p =
    1.0000    3.0000    0.0000    4.0000
```

Множення поліномів виконується за допомогою функції `conv`. Припустимо, що ми хочемо одержати поліном $n(s)$, де $n(s) = (3s^2 + +2s + 1)(s + 4)$. Ця процедура виконується так:

```
>> p=[3 2 1];q=[1 4];
>> n=conv(p,q)
n =
     3    14     9     4
```

У результаті множення одержуємо поліном

$$n(\mathrm{s}) = 3s^3 + 14s^2 + 9s + 4.$$

Для обчислення значення полінома при заданому значенні змінної використовується функція `polyval`.

```
>> v=polyval(n,-5)
```

```
v =
   -66
```

Таким чином, поліном $n(s)$ має значення $n(-5) = -66$.

8.3. Передаточні функції

Тепер навчимося задавати передаточні функції. Передаточні функції систем створюються за допомогою функції `tf`. Нехай ми маємо дві ланки з передаточними функціями

$$W_1(s) = \frac{0{,}1s + 1}{s^2 + 0{,}2s + 1} \text{ і } W_2(s) = \frac{1}{s + 1}.$$

В MATLAB їх можна задати в такий спосіб:

```
>> num1=[0.1 1]; den1=[1 0.2 1];
>> W1=tf(num1,den1)

Transfer function:
   0.1 s + 1
---------------
s^2 + 0.2 s + 1
```

Тут `num1` і `den1` – чисельник і знаменник передаточної функції $W_1(s)$ відповідно. Таким чином, в MATLAB передаточна функція задається як відношення поліномів. У принципі, чисельник і знаменник передаточної функції можна задавати не окремо, а безпосередньо з функцією `tf`:

```
>> W1=tf([0.1 1],[1 0.2 1])

Transfer function:
   0.1 s + 1
---------------
s^2 + 0.2 s + 1
```

Якщо чисельником передаточної функції є тільки вільний член (коефіцієнт підсилення), то можливий наступний варіант завдання передаточної функції:

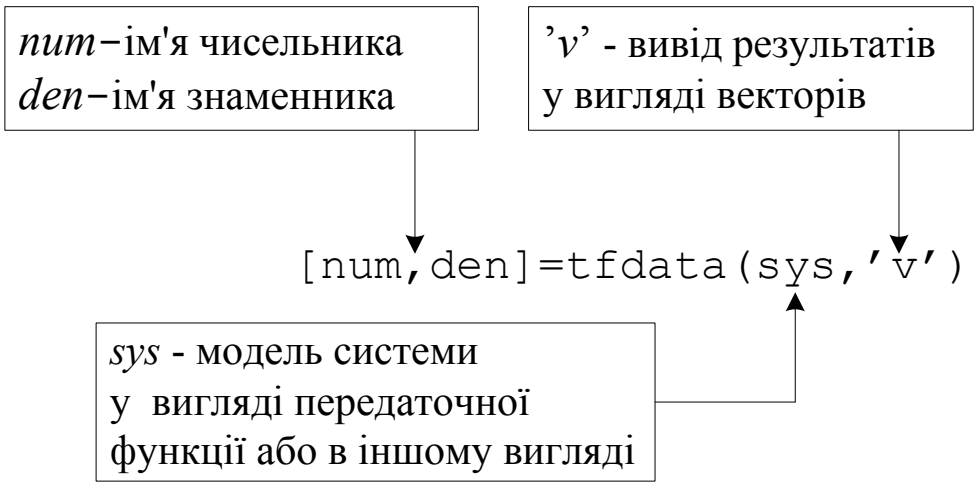
```
>> W2=tf(1,[1 1])
```

```
Transfer function:
  1
-----
s + 1
```

Передаточну функцію можна також задати як раціональну функцію змінної «*s*»:

```
>> s=tf('s');   %Завдання змінної Лапласа
>> W=s/(0.5*s^2+2*s+1)

Transfer function:
        s
-----------------
0.5 s^2 + 2 s + 1
```

Часто необхідно виділити чисельник та знаменник вже сформованої передаточної функції, наприклад, для знаходження її полюсів чи нулів. Виділення чисельника та знаменника передаточної функції можна здійснити за допомогою команди `tfdata`. Найбільш зручний синтаксис цієї команди має наступний вигляд



Команду `tfdata` можна також застосовувати якщо модель системи надана у іншому форматі – у вигляді ZPG – моделей або у просторі станів.

8.4. Завдання ZPG - моделей

ZPG-модель - це розкладена на множники передаточна функція. У цьому форматі модель характеризується коефіцієнтом підсилення *k*, нулями (коренями чисельника) передаточної функції *z* і полюсами передаточної функції *p* (коренями знаменника).

SISO, тобто одномірну модель, можна задати за допомогою команди `zpk`. Наприклад, якщо

$$W(s) = -2 \cdot \frac{s}{(s-2)(s^2-2s+2)},$$

то порядок завдання ZPG-моделі:

```
>> z=0;                 %Нулі
>> p=[2 1+j 1-j];       %Полюси
>> k=-2;                %Підсилення
>> W=zpk(z,p,k)

Zero/pole/gain:
        -2 s
--------------------
(s-2) (s^2  - 2s + 2)
```

Якщо система не має полюсів чи нулів, то на відповідному місці ставиться пуста матриця `[]`, наприклад

```
>> W=zpk([],p,k)
```

Як і у випадку з командою `tf`, ZPG-модель можна задати як раціональне вираження:

```
>> s=zpk('s');
>> W=-2*s/(s-2)/(s^2-2*s+2);
```

8.5 Завдання моделі в просторі станів

MATLAB також дозволяє задавати та аналізувати моделі систем керування в просторі станів, які описуються наступними рівняннями:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \ ; \\ \mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \ , \end{cases} \tag{8.1}$$

де **x** – вектор стану, **u** – вектор вхідних впливів, **y** – вектор вихідних сигналів, **A** – матриця коефіцієнтів системи; **B** - матриця керування; **C** – матриця спостереження виходу; **D** – матриця зв'язку. Розмірність цих матриць показана на рис. 8.1.

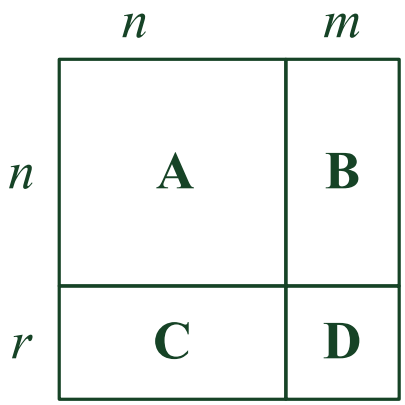


Рис. 8.1. Розмірність матриць у рівнянні (8.1)
n - кількість змінних стану *m* - число вхідних сигналів,
r - число вихідних сигналів.

Для завдання моделі системи в просторі станів використовується команда `ss`, що має наступний формат:

```
ss(A,B,C,D).
```

Наприклад,

```
>> A=[0 1;-5 -2];
>> B=[0;3];
>> C=[1 0];
>> D=0;
>> W=ss(A,B,C,D);
```

Більш докладно прийоми моделювання систем керування, представлених у просторі станів будуть розглянуті в главі 11.

8.6 Завдання FRD-моделі

FRD-модель (або частотна передаточна функція) описує поведінку системи в частотній області. Команда `frd` дозволяє визначити реакцію системи на синусоїдальний сигнал будь-якої частоти. Для цього модель системи повинна бути представлена у вигляді передаточної функції, ZPG-формі або в просторі станів, і, крім того, повинен бути зазначений вектор-рядок з частотами вхідного сигналу:

```
>> W=tf([1 0], [0.1 2 1]);
>> freq=[1 10 100 1000];%Частоти вхідного сигналу
>> H=frd(W,freq)%Реакція на вхідні впливи
From input 1 to:
  Frequency(rad/s)         output 1
  ----------------         --------
             1         0.415800+0.187110i
            10         0.415800-0.187110i
           100         0.019268-0.096243i
          1000         0.000200-0.009996i

Continuous-time frequency response data model.
```

В MATLAB також є можливість побудувати частотну передаточну функцію системи керування на підставі даних, отриманих експериментальним шляхом. Для цього необхідно задати вектор частот і вектор відгуків, збуджений цими частотами. Наприклад:

```
>> freq=[1000;2000;3000];%Вектор частот у Герцах
>> resp=[-0.8126-0.0003i;-0.1751-00016i;....
-0.0926-0.4630i];%Вектор відгуків
>> W=frd(resp,freq,'Units','Hz')
From input 1 to:
  Frequency(Hz)          output 1
  -------------          --------
     1000             -0.8126- 0.0003i
     2000             -0.1751-16.0000i
     3000             -0.0926- 0.4630i
```

8.7 Перетворення структурних схем

Одержавши моделі окремих ланок системи, необхідно об'єднати всі ці ланки в єдину структуру, створивши тим самим систему керування. За допомогою MATLAB можна виконати всі необхідні перетворення структурної схеми. Як приклад розглянемо перетворення схем, ланки яких задані у вигляді передаточних функцій.

Найпростішим є послідовне з'єднання ланок (рис. 8.2).

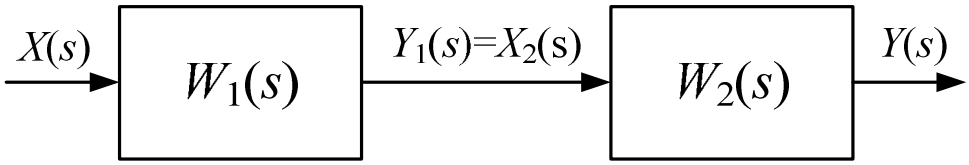


Рис. 8.2. Послідовне з'єднання ланок

Загальну передаточну функцію $W(s)$, що зв'язує $X(s)$ і $Y(s)$, можна знайти, використовуючи команду `series`. Наприклад, якщо

$$W_1(s) = \frac{2}{s+1} \text{ і } W_2(s) = \frac{0{,}5}{2s+1},$$

тоді

```
>> W1=tf(2,[1 1]);
>> W2=tf(0.5,[2 1]);
>> W=series(W1,W2)

Transfer function:
       1
--------------
2 s^2 + 3 s + 1
```

Недоліком цього способу є те, що команда `series` є функцією двох змінних. Це значно захаращує розрахунки при визначенні загальної передаточної функції послідовно з'єднаних трьох і більше ланок. Тому, на наш погляд, зручніше за все скористатися операцією множення (нагадаємо, що загальна передаточна функція при послідовному з'єднанні ланок визначається як добуток складових передаточних функцій):

```
>> W=W1*W2

Transfer function:
       1
--------------
2 s^2 + 3 s + 1
```

У структурних схемах дуже часто зустрічається паралельне з'єднання елементів (рис. 8.3).

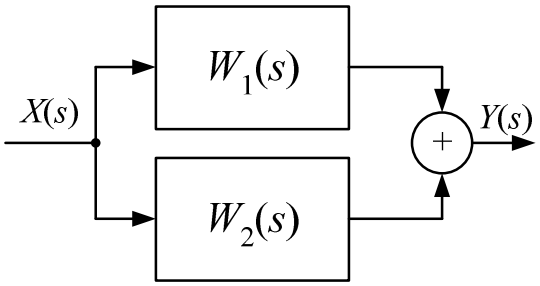


Рис. 8.3. Паралельне з'єднання ланок

У таких випадках для визначення передаточної функції з'єднання використовується функція `parallel`. Наприклад, якщо

$$W_1(s) = 1{,}2 \text{ і } W(s) = \frac{0{,}5}{s},$$

тоді

```
>> W1=tf(1.2,1);
>> W2=tf(0.5,[1 0]);
>> W=parallel(W1,W2)

Transfer function:
1.2 s + 0.5
-----------
     s
```

Зрозуміло, що можна просто скористатися операцією додавання, що, на наш погляд, навіть зручніше, оскільки число аргументів функції `parallel` повинне дорівнювати 2 або 6.

Передаточна функція замкнутої системи визначається вираженням

$$W(s) = \frac{W_1(s)}{1 \pm W_1(s)W_{зз}(s)}, \qquad (8.2)$$

де знак «+» ставится у випадку негативного зворотного зв’язку (рис. 8.4), а знак «-» - у випадку позитивного зворотного зв’язку

Обчислити передаточну функцію замкнутої системи можна за допомогою функції `feedback`. Ця функція застосовна як до одноконтурних, так і до багатоконтурних систем керування.

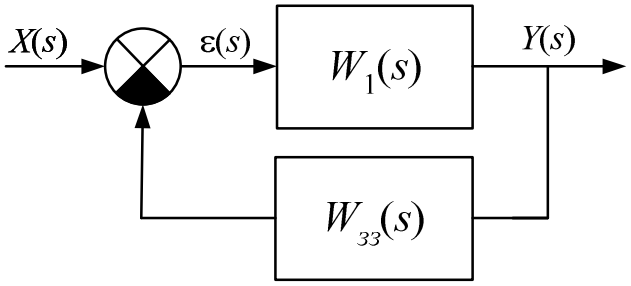


Рис. 8.4. Система керування зі зворотним зв'язком

У випадку неодиничного зворотного зв'язку функція `feedback` має наступний формат:

```
W=feedback(W1,Woc,sign),
```

де `sign = +1` у випадку позитивного зворотного зв'язку і `sign = -1` - у випадку негативного зворотного зв'язку.

Якщо в аргументах функції `feedback` не зазначений знак зворотного зв'язку `sign`, то за замовчуванням він передбачається негативним.

Часто зустрічається випадок, коли замкнута система має одиничний зворотний зв'язок. Формат функції `feedback` у цьому випадку має вигляд:

```
W=feedback(W1,1,sign)
```

Нехай, наприклад, передаточні функції об'єкта $W_o(s)$ і регулятора $W_p(s)$ на рис. 8.5 дорівнюють

$$W_o(s) = \frac{1}{2s^2 + 3s + 1} \text{ і } W_p(s) = \frac{1{,}2s + 0{,}5}{s}.$$

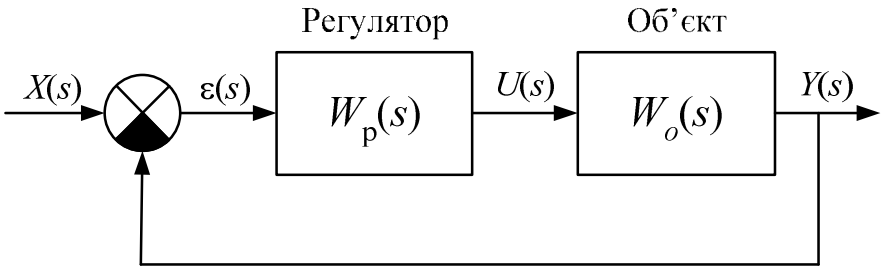


Рис. 8.5. Структурна схема зі зворотним зв'язком

Тоді передаточна функція замкнутої системи:

```
>> Wo=tf(1,[2 3 1])
>> Wp=tf([1.2 0.5],[1 0])
>> W=feedback(Wp*Wo,1)

Transfer function:
        1.2 s + 0.5
-------------------------
2 s^3 + 3 s^2 + 2.2 s + 0.5
```

У результаті перетворення структурної схеми може виникнути випадок, коли результуюча передаточна функція системи має однакові полюси і нулі. Для їхнього скорочення використовується команда `minreal`.

Нехай, наприклад, після перетворення деякої структурної схеми, ми одержали наступний результат:

$$W(s) = \frac{s^5 + 4s^4 + 6s^3 + 6s^2 + 5s + 2}{12s^6 + 205s^5 + 1066s^4 + 2517s^3 + 3128s^2 + 2196s + 712}.$$

Якщо обчислити полюси і нулі $W(s)$, то можна виявити, що поліноми в чисельнику і знаменнику мають однаковий співмножник $(s + 1)$. Ці співмножники необхідно скоротити:

```
>> W=tf([1 4 6 6 5 2],[12 205 1066 2517 3128 2196 712])

Transfer function:
       s^5 + 4 s^4 + 6 s^3 + 6 s^2 + 5 s + 2
-----------------------------------------------
12 s^6 + 205 s^5 + 1066 s^4 + 2517 s^3 + 3128 s^2 + 2196 s + 712
```

```
>> W=minreal(W)

Transfer function:
0.08333 s^4 + 0.25 s^3 + 0.25 s^2 + 0.25 s + 0.1667
-----------------------------------------------
s^5 + 16.08 s^4 + 72.75 s^3 + 137 s^2 + 123.7 s + 59.33
```

Як бачимо, після використання команди `minreal` порядки поліномів у чисельнику та знаменнику зменшилися на одиницю за рахунок скорочення одного полюса і одного нуля.

Завдання для самостійної роботи

1. Розгляньте два поліноми:

$P(s) = 0{,}05s^2 + 0{,}4s + 1$, $Q(s) = s + 1$.

За допомогою MATLAB визначте наступне.
а) Добуток $P(s)\cdot Q(s)$.
б) Полюси та нулі передаточної функції $W(s) = Q(s)/P(s)$;
в) Значення $P(s)\cdot$при $s = -2$.

2. На рис. 8.6 надана структурна схема двигуна постійного струму, що керується по ланцюгу якоря.

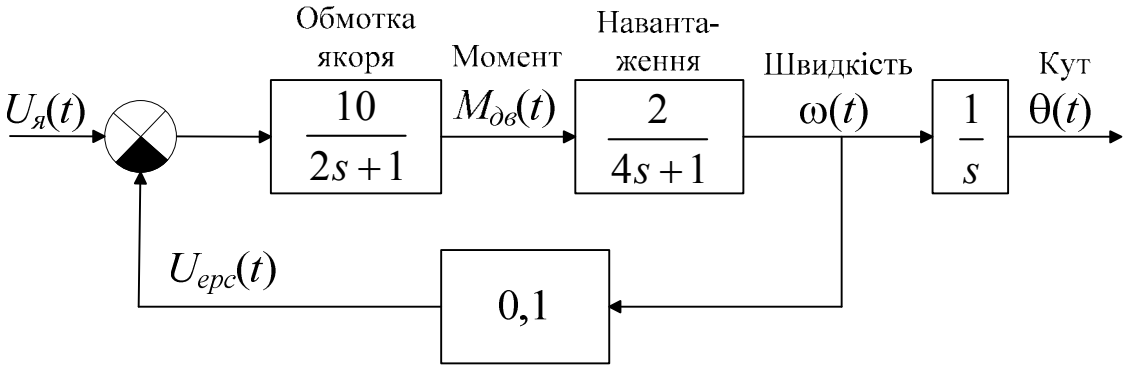


Рис. 8.6. Структурна схема двигуна постійного струму, що керується по ланцюгу якоря

а) За допомогою команд `series` та `feedback` знайдіть загальну передаточну функцію $W_\omega(s)$ двигуна, якщо вихідною величиною вважати швидкість обертання валу $\omega(t)$.

б) Мінімізуйте одержану передаточну функцію за допомогою команди `minreal`.

в) Представте результат, що одержаний у п. *б* у ZPG -формі. Для розкладання поліному на множники можна використати команду `roots`.

г) Знайдіть загальну передаточну функцію $W_\theta(s)$ двигуна, якщо вихідною величиною вважати кут повороту валу $\theta(t)$ та представте результат у ZPG.

3. На рис. 8.7 зображена замкнена система керування.

а) Визначить передаточну функцію $W(s)$ системи.

б) Вичисліть полюси $W(s)$.

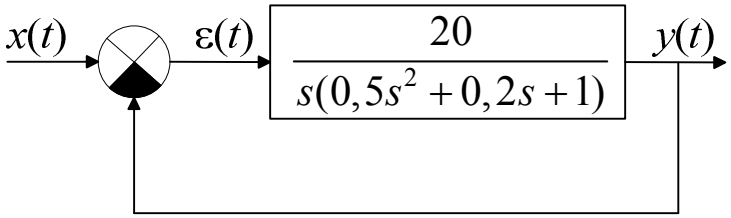


Рис. 8.7. Структурна схема замкненої системи керування

4. Передаточну функцію $W_\omega(s)$ двигуна постійного струму з завдання 2, *а* можна також представити у просторі станів за допомогою рівнянь (8.1), де

$$\mathbf{A} = \begin{bmatrix} -0,75 & 1 \\ -0,375 & 0 \end{bmatrix}, \ \mathbf{B} = \begin{bmatrix} 0 \\ 2,5 \end{bmatrix}, \ \mathbf{C} = \begin{bmatrix} 1 & 0 \end{bmatrix}, \ \mathbf{D} = 0.$$

За допомогою команды `ss` побудуйте MATLAB-модель двигуна у просторі станів.

5. Передаточна функція системи має вигляд

$$W(s) = \frac{5}{0,7s + 1}.$$

Обчисліть реакцію системи на синусоїдальний сигнал частотою 0,1; 1; 5 та 10 рад/с. Чому з підвищенням частоти вхідного сигналу амплітуда вихідного сигналу різко зменшується?

9. АНАЛІЗ СИСТЕМ КЕРУВАННЯ В CONTROL SYSTEM TOOLBOX

9.1. Аналіз стійкості в MATLAB

Тепер, коли ми вже вміємо задавати математичні моделі систем керування, можна приступати до аналізу цих систем.

Основне питання аналізу системи керування – це питання її стійкості. У більшості випадків нестійка система непрацездатна.

Стійкою є така система, що після обмеженого обурюючого впливу повертається до вхідного стану. Нагадаємо, що для стійкості лінійної системи необхідно й достатньо, щоб всі корені характеристичного полінома лежали ліворуч від мнимої вісі комплексної площини коренів. Отже, мнима вісь комплексної площини є границею стійкості. Якщо хоча б один речовинний корінь або пара комплексно сполучених коренів перебуває праворуч від мнимої вісі, то система є нестійкою. Якщо є нульовий корінь або пара чисто мнимих коренів, то система вважається нейтральною (тій, що перебуває на границі стійкості). Існують правила, або критерії, які дозволяють, не вирішуючи характеристичного рівняння, визначити, чи перебувають всі його корені в лівій напівплощині. Критерії бувають алгебраїчні (Рауса, Гурвіца), і частотні (Найквіста, Нікольса і т.д.). MATLAB має широкі можливості для аналізу стійкості систем автоматичного керування.

Нам уже відомо, як за допомогою функції `roots` знаходити корені якого-небудь полінома. Наприклад, якщо характеристичний поліном має вигляд $A(s) = s^3+s^2+2s+23$, то його корені можна визначити таким чином:

```
>> A= [1 1 2 23];
>> roots(A)
```

У результаті одержуємо колонку, що містить шукані корені

```
ans =
  -2.9558
   0.9779 + 2.6124i
   0.9779 - 2.6124i
```

Як бачимо, запропонована система буде нестійкою.

Якщо задано передаточну функцію, то можна скористатися функцією `pole`, що обчислює полюсів передаточної функції:

```
>> W=tf([1 0.2],[A])
Transfer function:
      s + 0.2
--------------------
s^3 + s^2 + 2 s + 23

>> pole(W)

ans =
  -2.9558
   0.9779 + 2.6124i
   0.9779 - 2.6124i
```

За допомогою функції `pzmap` можна вказати розташування на комплексній площині полюсів і нулів передаточної функції. Нулі на діаграмі позначаються кружечками, а полюси - хрестиками. Якщо функція `pzmap` викликається без аргументів, то діаграма будується автоматично (рис. 9.1):

```
>> pzmap(W)
```

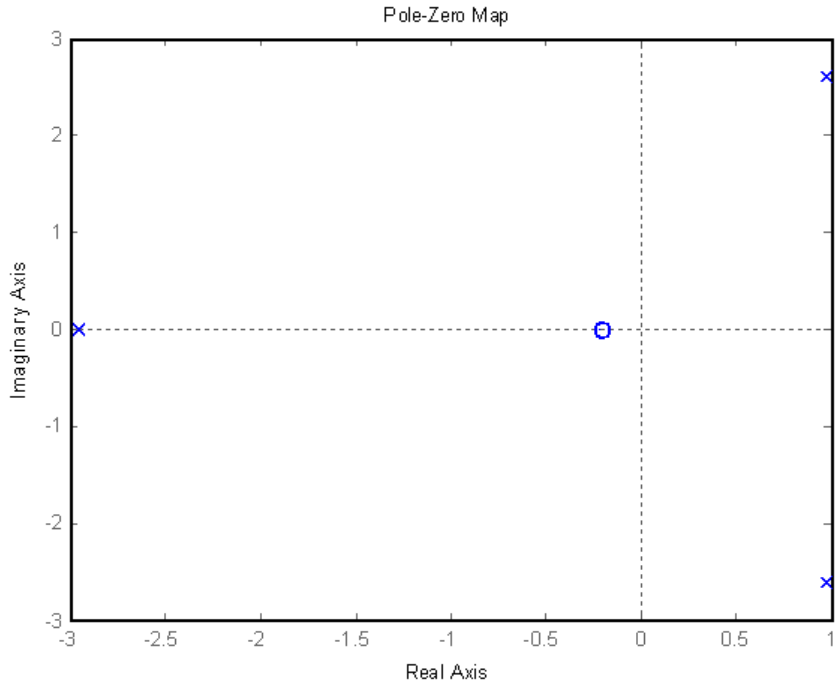


Рис. 9.1. Результат виконання команди `pzmap`

Корені характеристичного полінома можна знайти, використовуючи команду `damp(sys)`. Якщо ввести її у командному рядкувікні керування, то після виконання одержимо три колонки із цифрами. Перша (`Eigenvalue`) містить власні значення або корені характеристичного полінома, друга (`Damping`) коефіцієнти демпфірування, третя (`Freq. (rad/s)`) власні частоти (рад/с).

Розглянемо приклад системи керування числом обертів двигуна внутрішнього згоряння (рис. 9.2). Постійна часу T_1 обумовлена обмеженнями на упорскування пального в карбюратор і пропускну здатність трубопроводу. Двигун має постійну часу $T_{\text{дв}} = J/b = 3$ с, де J – момент інерції, b – коефіцієнт тертя вала. Постійна часу датчика швидкості $T_{\text{дш}} = 0{,}4$ с.

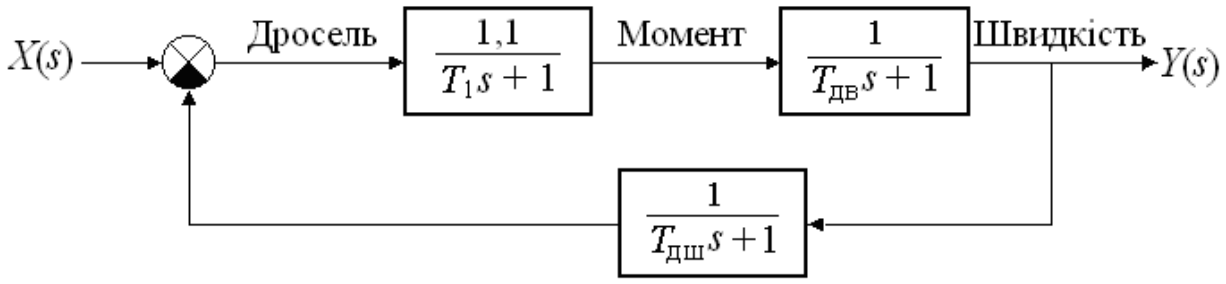


Рис. 9.2. Система керування числом обертів двигуна внутрішнього згоряння

Визначимо, чи є задана система стійкою.

```
>> W1=tf(1.1,[1 1]);
>> Wdv=tf(1,[3 1]);%Двигун
>> Wds=tf(1,[0.4 1]);%Датчик швидкості
```

Для визначення стійкості необхідно знайти корені характеристичного полінома умовно розімкнутої системи. Знайдемо, спочатку, її передаточну функцію:

```
>> Wp=series(W1,Wdv);
>> W=feedback(Wp,Wds)

Transfer function:
          0.44 s + 1.1
------------------------------
1.2 s^3 + 4.6 s^2 + 4.4 s + 2.1
```

Тепер знайдемо корені характеристичного полінома. Для цього у командному радку набираємо команду:

```
>> pole(W)
ans =
  -2.7227
  -0.5553 + 0.5782i
  -0.5553 - 0.5782i
```

Як бачимо, всі корені лежать ліворуч від мнимої вісі комплексної площини, тобто система є стійкою.

Перевіримо стійкість системи іншими способами.

1. Найбільшу наочність при судженні про стійкість дають перехідні характеристики (див. визначення стійкості):

```
>> subplot(2,1,1);step(W);
>> subplot(2,1,2);impulse(W);
```

У результаті з'явиться наступне вікно із графіками (рис. 9.3).

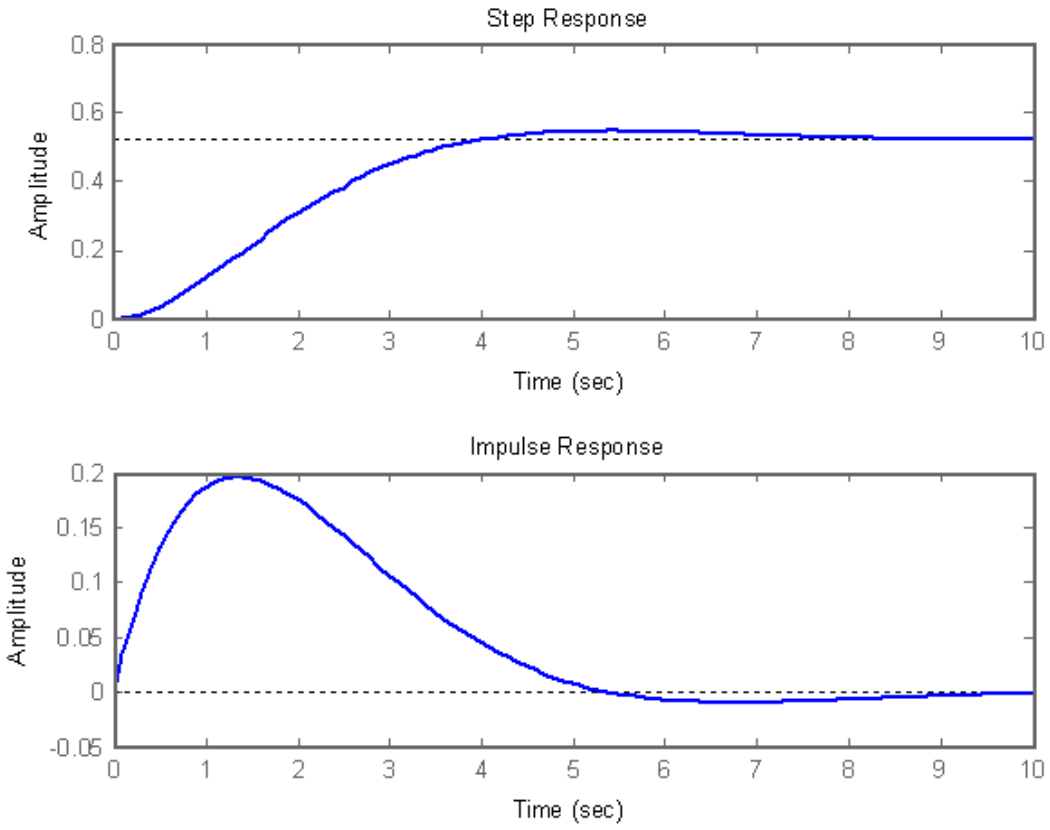
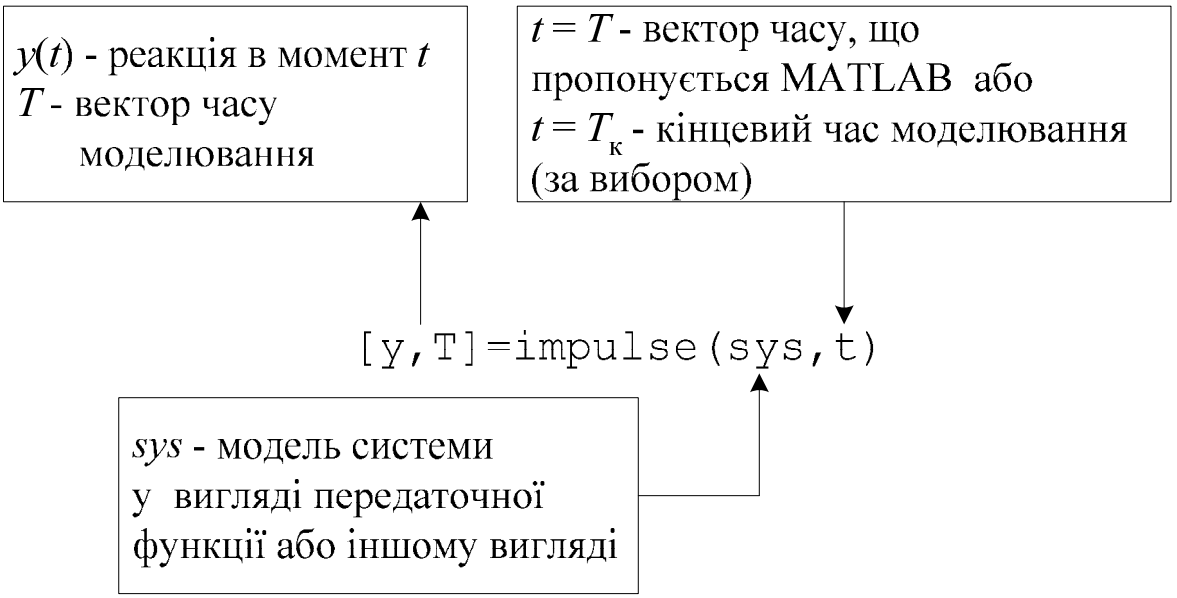


Рис. 9.3. Часові характеристики системи керування числом обертів двигуна внутрішнього згоряння

Тут ми використали нову для нас команду MATLAB `impulse`, що дозволяє будувати імпульсну перехідну функцію системи. Формат функції `impulse` наступний:



2. Згідно критерію стійкості Найквіста, стійкість замкнутої системи можна визначити за амплітудно-фазовою характеристикою (АФХ) розімкненої системи. АФХ (рис. 9.4) будується за допомогою команди `nyquist`:

```
>>Wraz=W1*Wdv*Wds;%Передаточна функція
>>                 %умовно розімкнутої системи
>>nyquist(Wraz)
```

Амплітудно-фазова характеристика не повинна охоплювати критичну точку з координатами (-1, *j*0). Це є достатньою та необхідною умовою того, щоб система була стійкою в замкнутому стані. Для зручності перегляду області в районі критичної точки (на рис. 9.4 позначена знаком «+») можна змінити масштаб вісей. Це можна зробити або вибравши в рядку меню графічного вікна (**Edit→Figure Properties**), або набравши після команди `nyquist` команду `axis([`X_{min} X_{max} Y_{min} Y_{max}`])`, де X_i і Y_i – відповідні мінімальні та максимальні значення осей (наприклад, `axis([-1.2 1.2 -1 1])`).

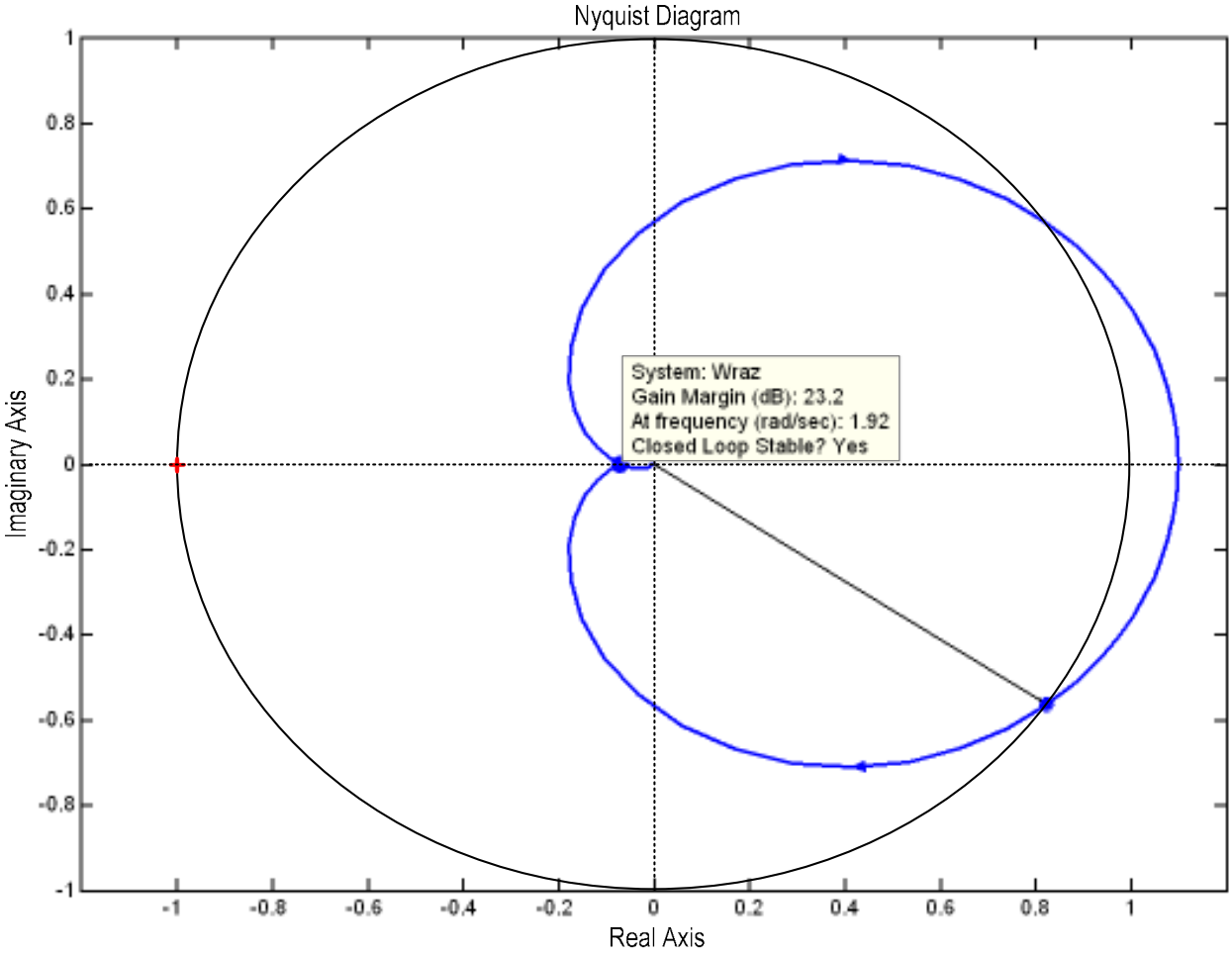


Рис. 9.4. Визначення стійкості за критерієм Найквіста

Викликавши контекстне меню (по щиглику правою кнопкою миші) і вибравши підменю **Characteristics**, можна визначити запаси стійкості системи за амплітудою та фазою. Підвівши курсор до характерних точок можна одержати чисельні значення для запасів стійкості (рис. 9.4). Зверніть також увагу на те, що MATLAB сам дає відповідь, чи стійка замкнута система (Closed Loop Stable? Yes). Це дуже зручно, тому що амплітудно-фазова характеристика може мати такий заплутаний вид, що складно визначити, чи охоплює вона критичну точку і якщо охоплює, то скільки разів.

Амплітудно-фазову характеристику можна представити і у вигляді годографа Нікольса (рис. 9.5), що являє собою графік розімкнутої системи в декартових координатах і будується за допомогою команди `nichols`:

```
>>nichols(Wraz)
```

За допомогою команди `Grid` з контекстного меню годографа Нікольса можна побудувати сітку координат для модуля і для фази. Така сітка називається діаграмою Нікольса. Діаграму Нікольса можна також побудувати за допомогою команди `ngrid`.

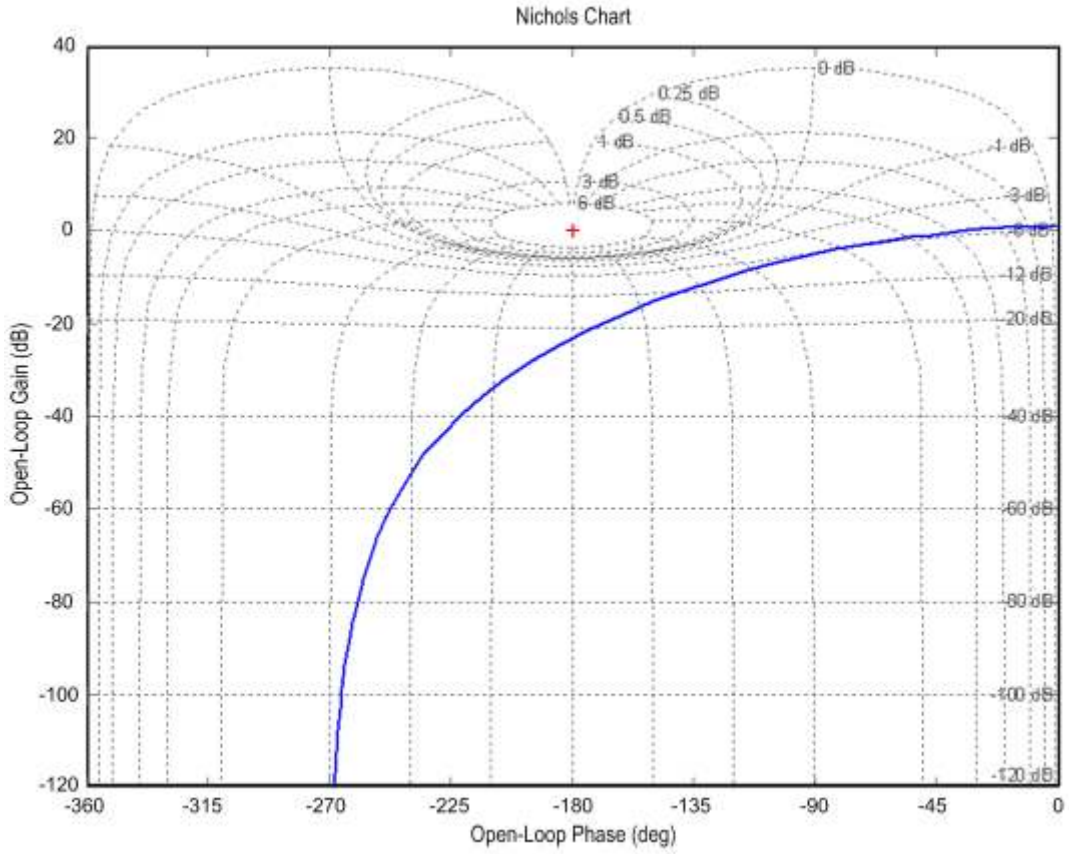


Рис. 9.5. Годограф і діаграма Нікольса

3. Про стійкість системи можна судити за логарифмічними частотними характеристиками. Система буде стійка, у тому випадку, якщо різниця між кількістю позитивних і негативних переходів ЛФЧХ через прямі $-\pi$, -3π, -5π, .. дорівнює $l/2$, де l - число коренів характеристичного рівняння розімкнутої системи, що лежать у правій напівплощині. Перехід ЛФЧХ знизу нагору, вважається позитивним, а зверху вниз - негативним.

У нашому випадку $l = 0$, тому для стійкості замкнутої системи необхідно, щоб різниця між кількістю позитивних і негативних переходів ЛФЧХ дорівнювала нулю.

Для побудови ЛАЧХ і ЛФЧХ використовують команду `bode` або `margin`. На відміну від команди `bode`, команда `margin` не лише будує логарифмічні частотні характеристики (діаграми Боде), але й відображає запаси стійкості за амплітудою та за фазою (рис. 9.6):

```
>>margin(Wraz)
```

Величина запасів за амплітудою (`Gm`) та за фазою (`Pm`) зазначена у верхній частині вікна.

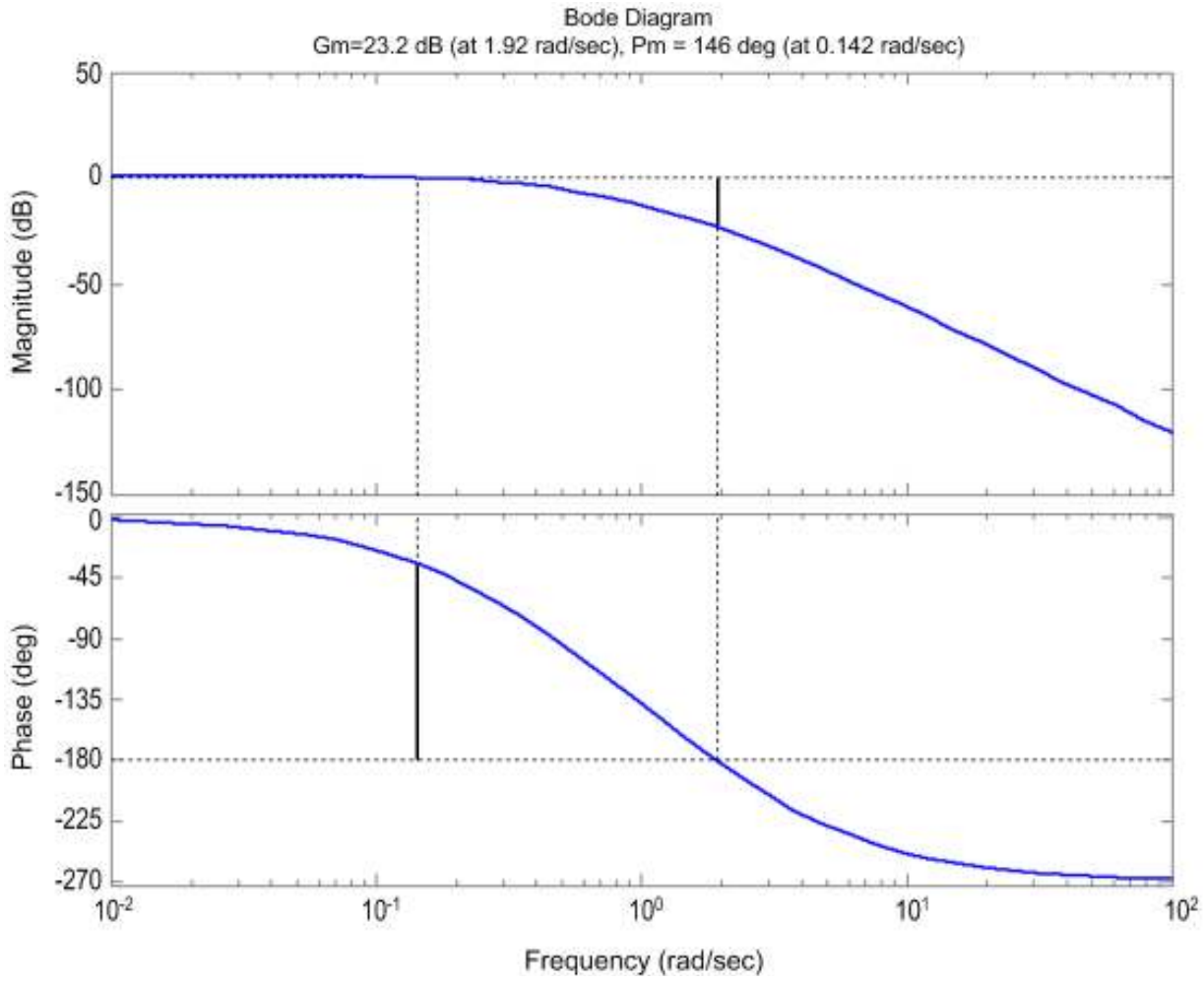


Рис. 9.6. Визначення стійкості за логарифмічними частотними характеристиками

9.2. Аналіз якості в MATLAB

Стійкість - головна, але не єдина вимога до системи керування. Стійку систему оцінюють додатково, користуючись спеціальними показниками якості.

Якість системи керування звичайно характеризується її реакцією на вхідний сигнал заданого виду. Але оскільки вхідні сигнали, які можуть реально діяти на систему, заздалегідь не відомі, то про її якість судять по реакції на типовий вхідний сигнал.

Як приклад розглянемо автопілот, призначений для автоматичного утримання літака на заданому курсі та висоті (рис. 9.7).

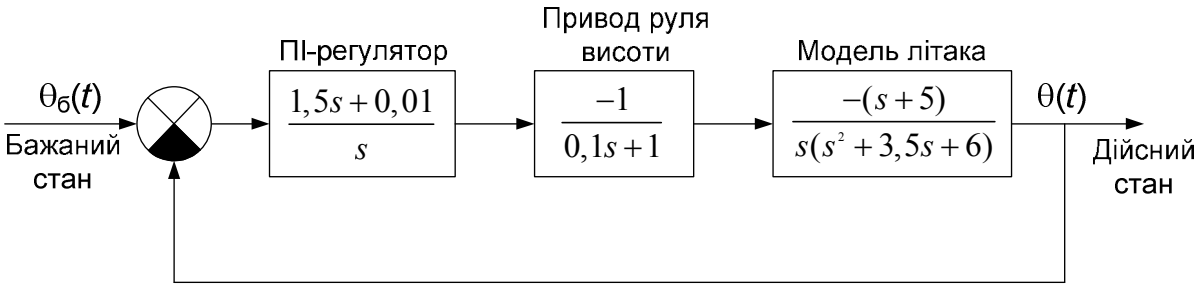


Рис. 9.7. Структурна схема системи керування літака з автопілотом

Ми вже знайомі із застосуванням функцій `step` та `impulse`. З їхньою допомогою можна побудувати реакцію системи на найпоширеніші типові сигнали - одиничний східчастий сигнал і одиничний імпульс.

Проаналізуємо якість роботи системи, приведеної на рис. 9.7, за допомогою вказаних функцій. Відповідний скріпт наведений нижче, а характеристики, побудовані з його допомогою представлені на рис. 9.8 і 9.9.

```
Wr=tf([1.5 0.01],[1 0]);%ПІ-регулятор
Wpr=tf(-1,[0.1 1]);%Привод руля висоти
Ws=tf([-1 -5],[1 3.5 6 0]);%Модель літака
Wraz=Wr*Wpr*Ws;%Передаточна функція розімкнутої САК
W=feedback(Wraz,1);%Передаточна функція замкнутої САК
step(W)
ylabel('h(t)')
```

```
figure(2)
impulse(W)
ylabel('\itw(t)'), grid
```

Викликавши контекстне меню і поставивши в підменю **Characteristics** прапорці напроти відповідних показників якості, можна одержати наочне уявлення про якість системи. Якщо підвести курсор миші до якої-небудь характерної точки, то на екрані відображається числове значення відповідного показника якості (рис. 9.8). Отримане вікно можна зафіксувати щигликом миші і потім переміщати в довільному напрямку або редагувати за допомогою контекстного меню.

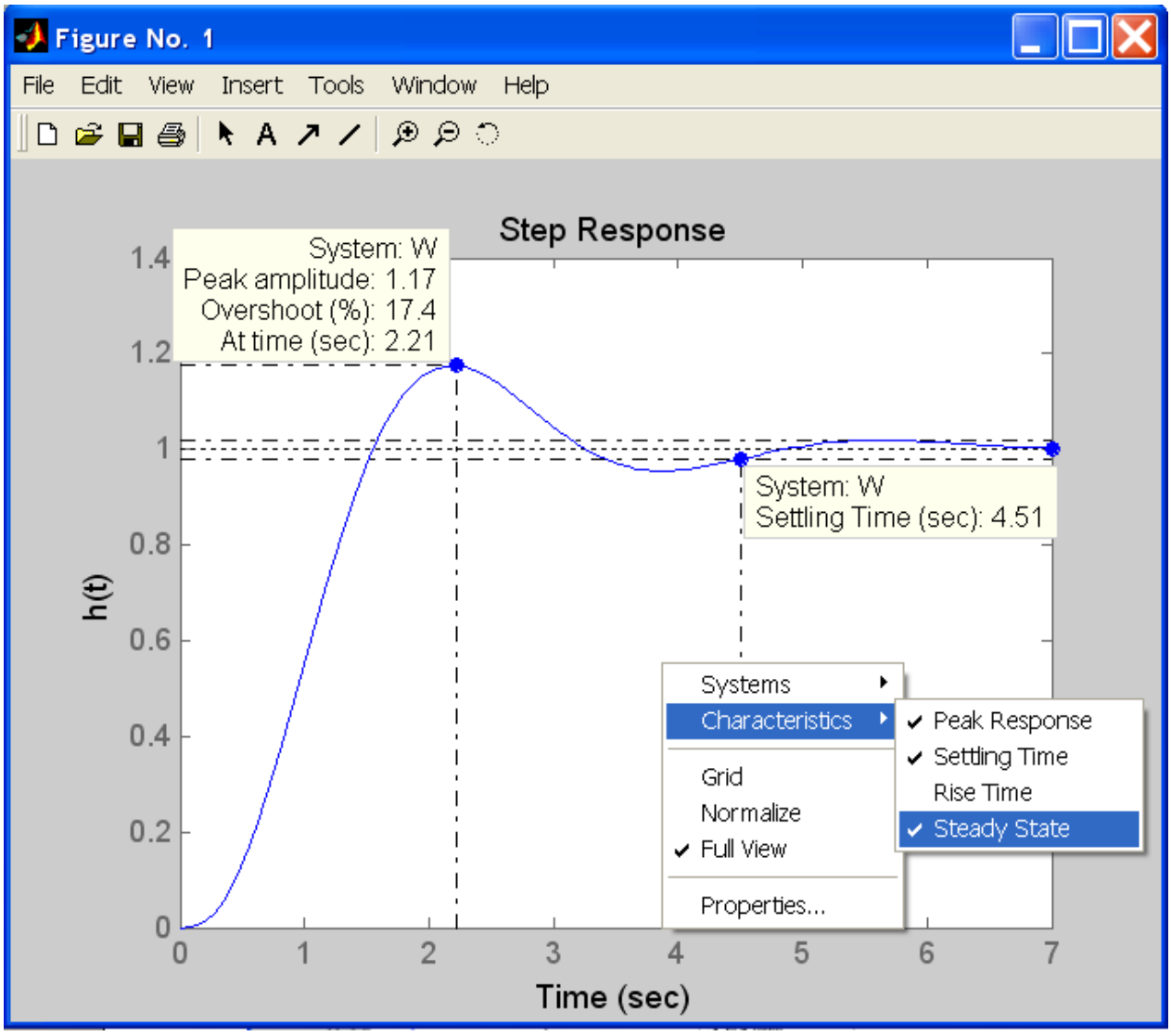


Рис. 9.8. Визначення показників якості за перехідною функцією

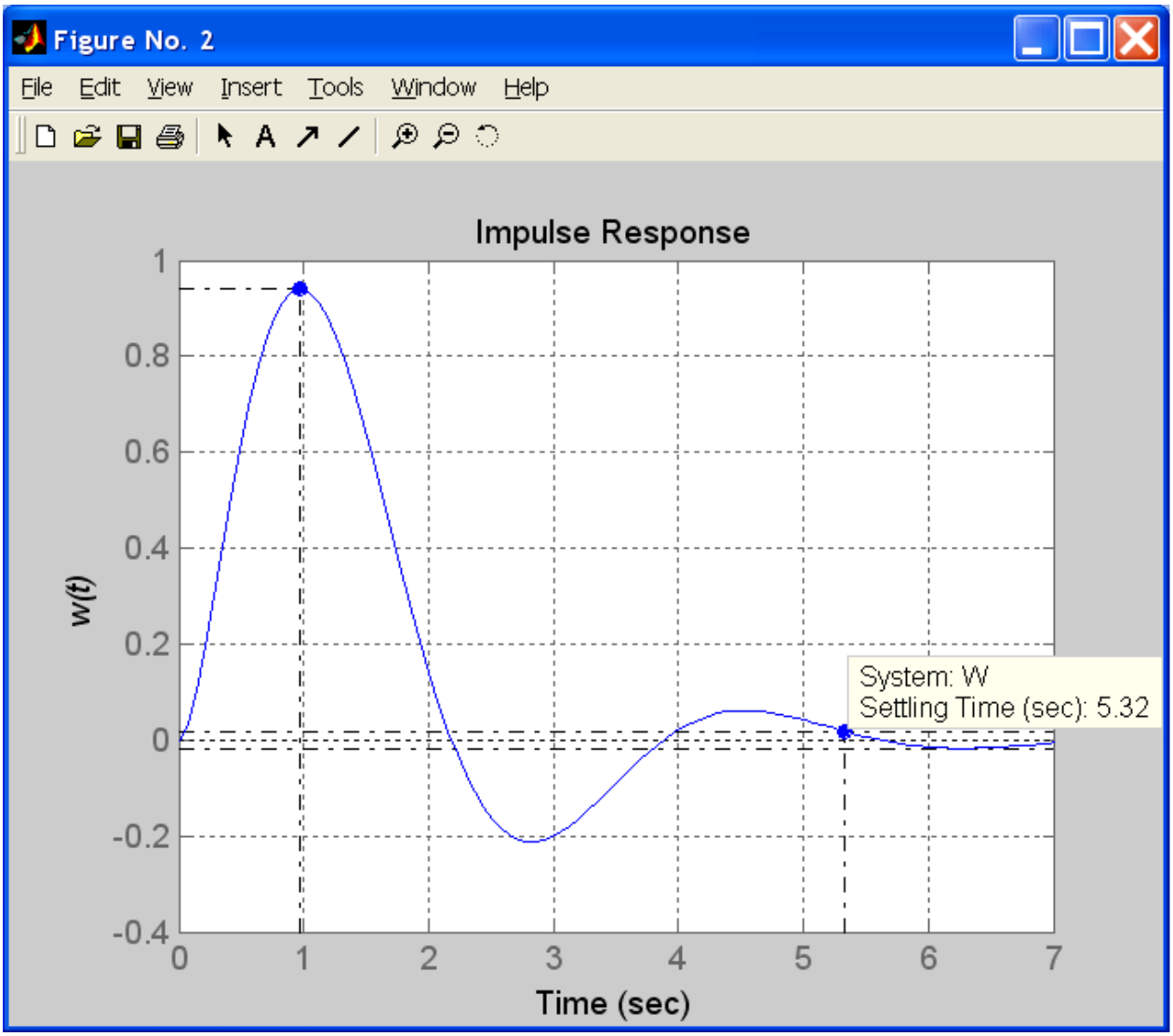
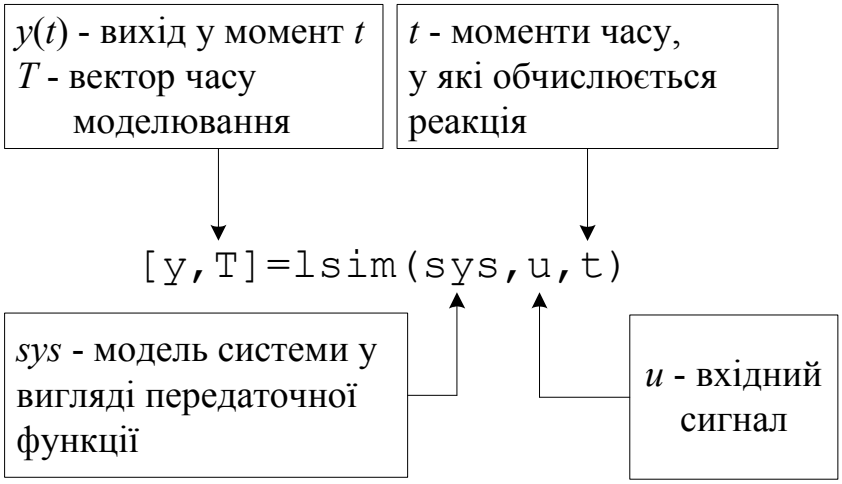


Рис. 9.9. Визначення показників якості за імпульсною перехідною функцією

Часто виникає необхідність визначення реакції системи на довільний вхідний сигнал. У цих випадках використовується функція `lsim`. З даною функцією ми ще будемо зустрічатися при моделюванні систем у просторі станів. У цьому випадку формат функції `lsim` наступний:



Звичайно, як і у випадку з функціями `step` та `impulse` можливо таке завдання команди:

```
>>lsim(sys,u,t)
```

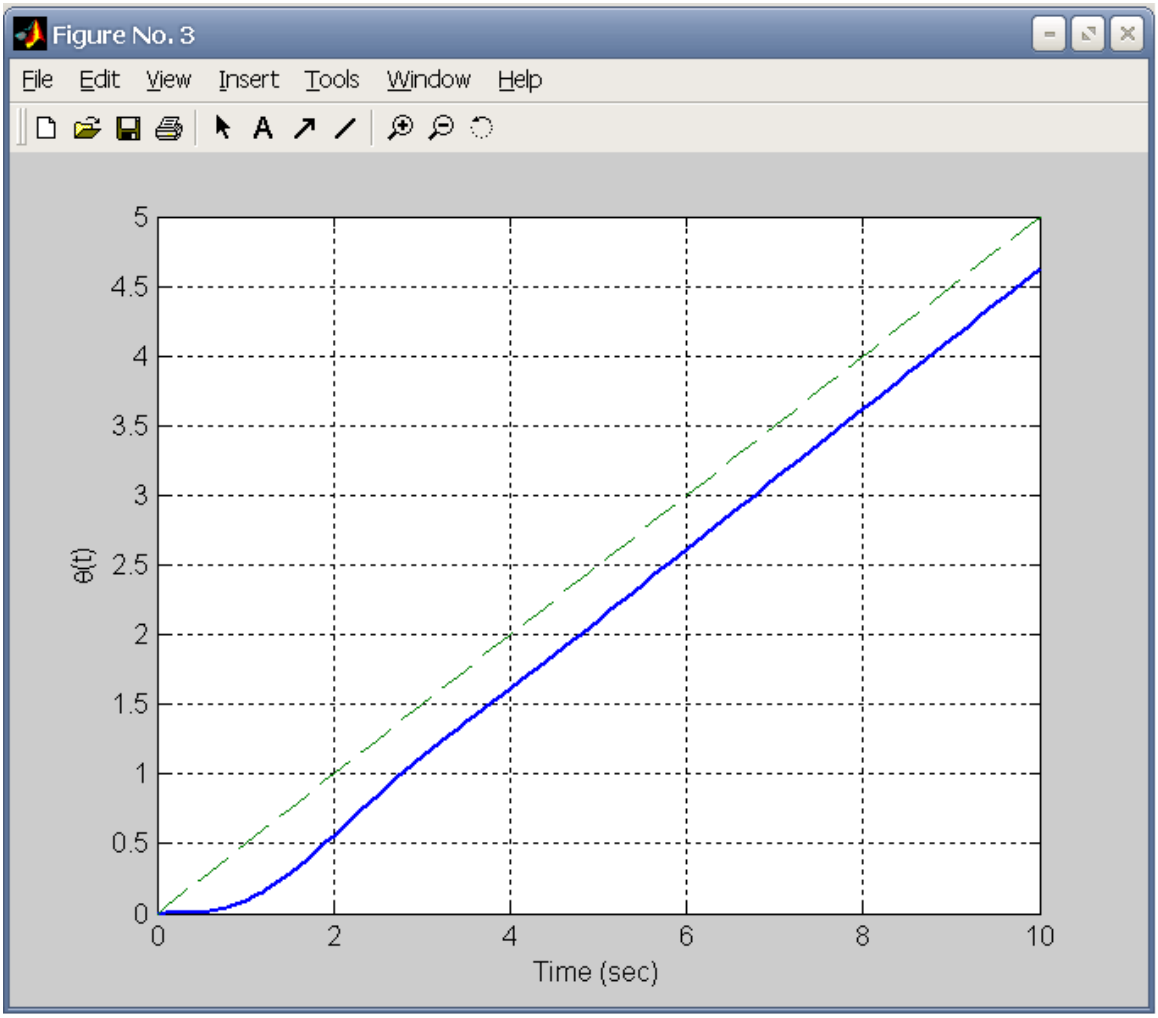


Рис. 9.10. Реакція автопілоту на лінійно наростаючий сигнал

Знайдемо за допомогою функції `lsim` реакцію автопілоту на вплив $\theta = kt$, де $k = 0{,}5°/с$ (рис. 9.10):

```
t=[0:0.1:10]'; %Час моделювання
u=0.5*t;%Вхідний сигнал
[y,T]=lsim(W,u,t);
figure(3), plot(T,y,t,u,'--'),
xlabel('Time (sec)'), ylabel('\theta(t)'), grid
```

Як видно з рис. 9.10, на перехідній характеристиці чітко простежується поява сталої помилки, тобто система має астатизм 1-го порядку.

9.3 Графічний інтерфейс LTI Viewer

Практично всі розглянуті вище характеристики можна побудувати, використовуючи графічний користувальницький інтерфейс LTI Viewer. Для цього потрібно набрати в командному рядку:

```
>>ltiview
```

або у вікні MATLAB вибрати **Start → Toolboxes → Control System → LTI Viewer**. У вікні, що з'явиться (рис. 9.11), натискаємо **File→Import**…, вибираємо передаточну функцію, яка нас цікавить, наприклад *W*, і натискаємо **OK**.

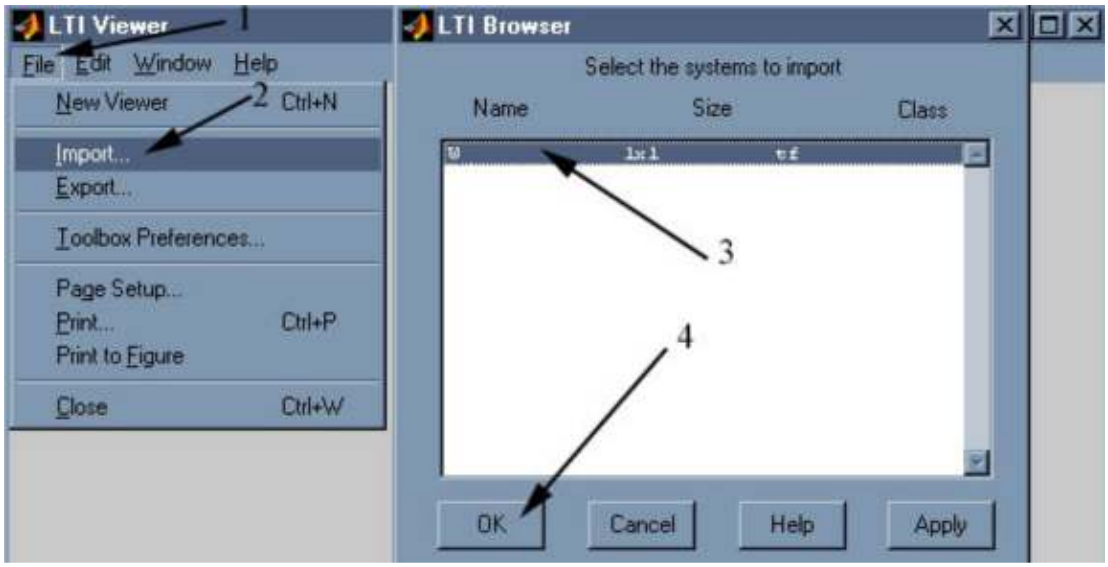


Рис. 9.11. Вибір передаточної функції в LTI Viewer

Після цього автоматично будується реакція *W* на «сходинку» і одиничний імпульс (рис. 9.12).

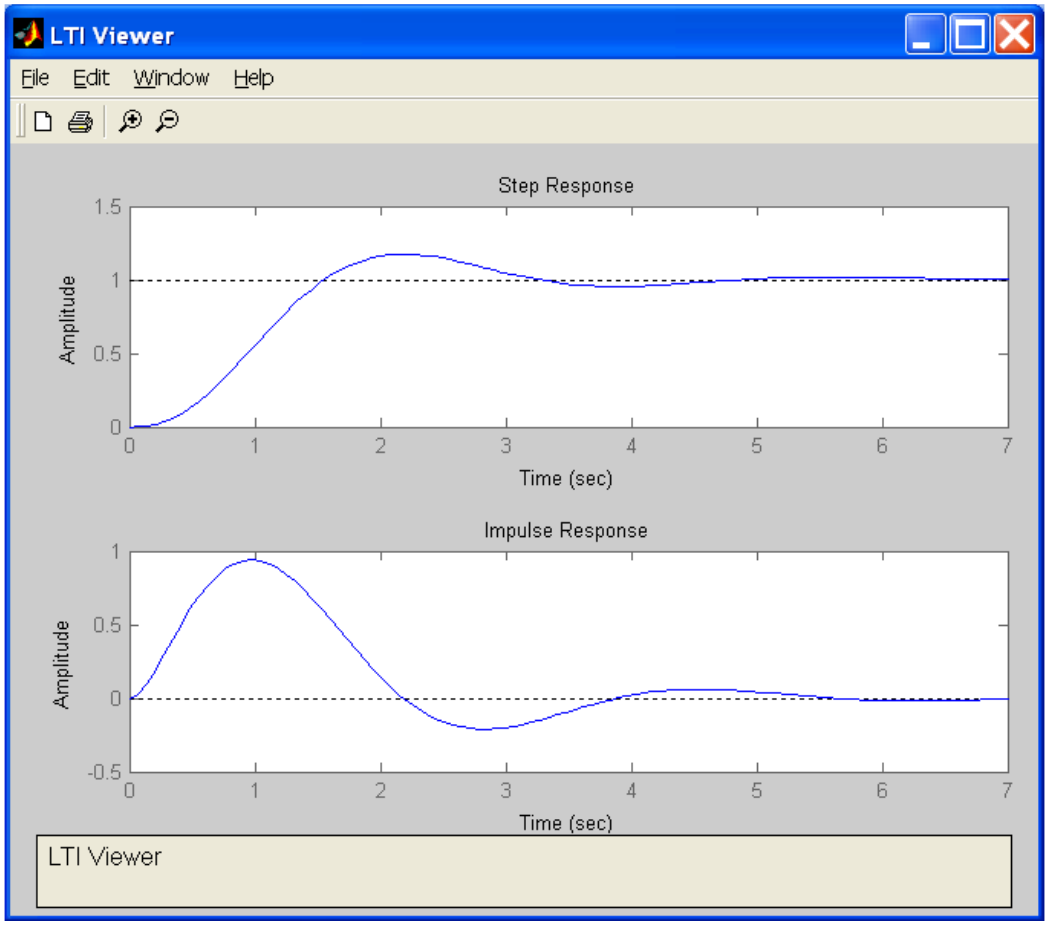


Рис. 9.12. Перехідні характеристики в LTI Viewer

Натисканням правої кнопки миші в області графіка викликається меню (рис. 9.13).

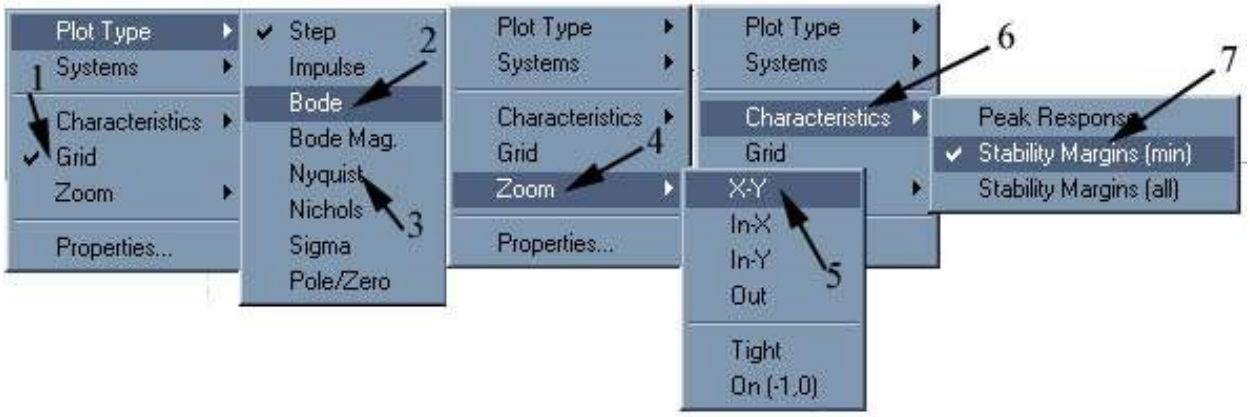


Рис. 9.13. Меню LTI Viewer

На отриманий графік можна накласти сітку - галочка навпроти **Grid** (1). Щоб побудувати логарифмічні частотні характеристики **Bode** (2) або АФХ **Nyquist** (3) потрібно вибрати відповідний рядок і натиснути на нього лівою кнопкою миші. Отриманий графік можна масштабувати, для цього вибираємо **Zoom** (4) і наприклад **X-Y** (5) і обводимо мишею при натиснутій лівій кнопці бажану область (наприклад для АФХ область у районі (-1, *j*0)). Для ЛФЧХ і АФХ можна визначити запаси по амплітуді й фазі. Для цього вибираємо **Characteristics** (6) і ставимо галочку навпроти **Stability Margins** (7). Якщо є запаси по амплітуді й фазі, то з'являються точки; наводячи на них покажчик миші, можна переглянути їхні чисельні значення.

Вибравши **File** $\rightarrow$ **Print to Figure** можна побудувати графік у стандартному графічному вікні MATLAB.

Вид графічного вікна в LTI Viewer і вибір характеристик, які необхідно в ньому побудувати, настроюється також у вікні **Plot Configurations**, що з'являється при виборі однойменного підміню в меню **Edit**.

У принципі, інтерфейс LTI Viewer простий та очевидний, і робота з ним не повинна викликати будь-яких серйозних утруднень.

Завдання для самостійної роботи

1. Система керування має наступну передаточну функцію

$$W(s) = \frac{4}{s^3 + 5s^2 + 3s + 1}.$$

а) За допомогою MATAB визначте полюси цієї передаточної функції та побудуйте комплексну площину з її полюсами та нулями. Чи є така система стійкою?

б) Перевірте зроблений Вами в п. а) висновок побудувавши перехідну функцію за допомогою команди `step`.

в) Побудуйте амплітудно- та фазово-частотні характеристики системи та знайдіть за ними запаси стійкості.

2. Дві системи керування у розімкненому стані мають наступні передаточні функції

$$W(s) = \frac{15}{(s+1)^3};\ W(s) = \frac{2}{2s+1}.$$

а) Як Ви вважаєте, яка з цих систем є стійкою, а яка ні? Чому?

б) Підтвердить свої відповіді за допомогою критерію Найквиста.

в) В різних графічних вікнах побудуйте імпульсні перехідні функції замкнених систем.

3. Розімкнена система має передаточну функцію

$$W(s) = \frac{1}{0,2s^2 + 0,4s + 1}.$$

а) За допомогою команди `step` побудуйте перехідну функцію розімкненої системи. Чому в цьому випадку дорівнює стала похибка?

б) У цьому ж графічному вікні побудуйте перехідну функцію замкненої системи з одиничним зворотнім зв'язком. Прокоментуйте одержані результати.

4. Система з передаточною функцією

$$W(s) = \frac{K}{s(s+1)}$$

охоплена одиничним негативним зворотнім зв'язком. Напишіть програму, яка б будувала в одному графічному вікні сімейство перехідних функцій системи при K = 1, 10 та 100. Визначить для кожного графіка основні показники якості: час регулювання, максимальне перерегулювання, час першого максимуму, сталу похибку та оформіть результати у вигляді таблиці. Зробіть висновки за одержаними результатами.

5. Для системи з завдання 4 побудуйте реакцію на вхідний сигнал $x(t) = t$ при $0 \le t \le 10$ с. Чому дорівнює стала похибка? Поясніть одержаний результат.

10. АНАЛІЗ І СИНТЕЗ СИСТЕМ ЗА ДОПОМОГОЮ КОРЕНЕВОГО ГОДОГРАФА

10.1 Порядок побудови кореневого годографа

У попередній главі ми розглядали питання аналізу якості систем керування. З курсу ТАК також відомо, що система керування повинна бути стійкою і мати адекватну реакцію на типові вхідні сигнали, повинна бути як можна менш чутливою до зміни параметрів, мати по можливості мінімальну сталу помилку і, нарешті, могти компенсувати вплив небажаних збурювань.

Замкнута система, що споконвічно мала би оптимальну якість без додаткової корекції її характеристик, - це досить рідкий випадок. Звичайно буває неможливо задовольнити водночас всім вимогам, що пред'являються до якості системи, тому виникає проблема пошуку компромісу між рядом вимог, серед яких можуть бути і суперечні одна одній.

Стійкість та якість перехідного процесу замкнутої системи автоматичного керування безпосередньо пов'язані з положенням коренів її характеристичного рівняння на комплексній площині. Щоб забезпечити належне розташування цих коренів, необхідне настроювання одного або декількох параметрів системи. Тому має сенс досліджувати, як переміщаються на комплексній площині корені характеристичного рівняння при зміні параметрів системи.

Кореневий годограф – це траєкторії коренів характеристичного рівняння системи на комплексній площині при зміні якого-небудь параметра системи. Зазвичай кореневий годограф будується при зміні коефіцієнта підсилення розімкнутої системи K.

Динамічні властивості замкнутої системи керування визначаються її передаточною функцією

$$W(s) = \frac{B(s)}{A(s)}, \qquad (10.1)$$

де $B(s)$ і $A(s)$ – поліноми ступеня m та n відповідно, причому $m \le n$. Корені полінома $B(s)$ називаються нулями передаточної функції, а корені полінома $A(s)$ - полюсами.

Положення полюсів $W(s)$ на комплексній площині визначає стійкість системи, а в сукупності з нулями – вид імпульсної перехідної функції $w(t)$ і перехідної функції $h(t)$.

Метод кореневого годографа дозволяє знаходити полюси і нулі передаточної функції замкнутої системи, маючи в розпорядженні полюси і нулі розімкнутої системи при зміні коефіцієнта підсилення розімкнутої системи K. Метод кореневого годографа є також методом проектування регуляторів.

Розглянемо порядок побудови кореневого годографа.

Для простої одноконтурної системи (рис. 10.1) характеристичне рівняння $A(s) = 0$ має вигляд

$$1 + K \cdot W(s) = 0, \tag{10.2}$$

де K – параметр, що варіюється.

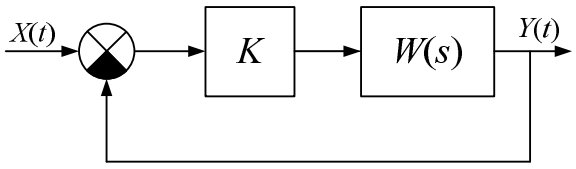


Рис. 10.1. Одноконтурна система керування

1. $W(s)$ слід представити у вигляді дробі, використовуючи полюси та нулі, тобто в ZPG-форматі:

$$1 + K \cdot \frac{\prod_{i=1}^{m}(s + z_i)}{\prod_{j=1}^{n}(s + p_j)} = 0, \tag{10.3}$$

де z - нулі, а p - полюси передаточної функції.

2. Далі варто розмістити полюси і нулі на комплексній площині. Звичайно становлять інтерес траєкторії коренів при зміні K від 0 до ∞. З виразу (10.3) одержимо

$$\prod_{j=1}^{n}(s + p_j) + K\prod_{i=1}^{m}(s + z_i) = 0. \tag{10.4}$$

При $K = 0$ корені характеристичного рівняння збігаються з полюсами $W(s)$, а при $K = \infty$ вони збігаються з нулями $W(s)$. Отже, *при зміні K від 0 до ∞ траєкторії коренів характеристичного рівняння* $1 + K \cdot W(s) = 0$ *починаються в полюсах* $W(s)$ *і закінчуються в нулях* $W(s)$. У більшості функцій $W(s)$ деякі нулі розташовуються в нескінченності на комплексній площині. Це пояснюється тим, що в цих функціях полюсів більше, ніж нулів. Якщо $W(s)$ має n полюсів та m нулів і $m \le n$, то $n - m$ віток кореневого годографа прагнуть до $n - m$ нулів, розташованих у нескінченності.

3. На наступному етапі, виділяють відрізки дійсної вісі, які належать кореневому годографу. *Ділянки кореневого годографа, що збігаються з дійсною віссю, завжди лежать ліворуч від непарного числа полюсів і нулів.*

4. Далі необхідно визначити число віток кореневого годографа. Оскільки кореневий годограф починається в полюсах і закінчується в нулях передаточної функції розімкнутої системи, то *число окремих віток годографа дорівнює числу полюсів*.

До нулів, розташованих у нескінченності, корені прагнуть уздовж асимптот із центром α_A, що утворюють із дійсною віссю кути φ_A:

$$\alpha_A = \frac{\sum_{j=1}^{n} z_j - \sum_{i=1}^{m} p_i}{n - m}, \tag{10.5}$$

$$\varphi_A = \frac{2q + 1}{n - m} \cdot \pi, \tag{10.6}$$

де q=0, 1, 2, ... , (n-m-1).

5. За допомогою критерію Рауса або Гурвіца визначають точки, у яких кореневий годограф перетинає мниму вісь (якщо такі точки є).

6. Визначають точки відриву кореневого годографа від дійсної вісі (якщо такі точки є). Для цього необхідно:

а) покласти $K = p(s)$;

б) одержати рівняння $dp(s)/ds = 0$;

в) знайти корені рівняння $dp(s)/ds = 0$ або графічно визначити корені максимуму $p(s)$.

7. Визначають кути виходу кореневого годографа з комплексних полюсів і кути входу в комплексні нулі:

$$\arg W(s) = \pi \pm q \cdot 2\pi \text{ при } s = p_j \text{ або } z_i .$$

8. Визначають положення коренів, при яких

$$\arg W(s) = \pi \pm q \cdot 2\pi \text{ при корені } s_x.$$

9. Визначають параметр K, що відповідає кожному кореню s_x:

$$K_x = \left| \frac{\prod_{j=1}^{n_p} (s + p_j)}{\prod_{i=1}^{n_z} (s + z_i)} \right|_{s=s_x} . \qquad (10.7)$$

При розташуванні віток кореневого годографа в лівій напівплощині s система стійка. При перетинанні вітками кореневого годографа мнимої вісі зліва направо система стає нестійкою. Нехай при деякому значенні $K_{кр}$ перетинання кореневого годографа із мнимою віссю відбудеться в деякій точці $j\omega_{кр.}$ Значення коефіцієнта підсилення $K_{кр}$, при якому система стає нестійкої, називають критичним, а величину $\omega_{кр}$ - критичною кутовою частотою.

Метод кореневого годографа дозволяє вибрати коефіцієнт підсилення системи керування, підібрати розташування полюсів і нулів передаточної функції коригувальних ланок, визначити параметри домінуючих полюсів системи (найближчих до початку координат площини s).

10.2. Побудова кореневого годографа в MATLAB

У якості прикладу побудуємо кореневий годограф (рис. 10.2) для системи керування автопілотом, що розглянута в попередній главі. Кореневий годограф можна використати для уточнення коефіцієнта системи з метою підвищення її якості. Передаточна функція цієї системи в розімкнутому стані $W_{raz}(s)$ дорівнює:

$$W_{raz}(s) = K\frac{1{,}5s^2 + 7{,}51s + 0{,}05}{s^2\left(0{,}1s^3 + 1{,}35s^2 + 4{,}1s + 6\right)}.$$

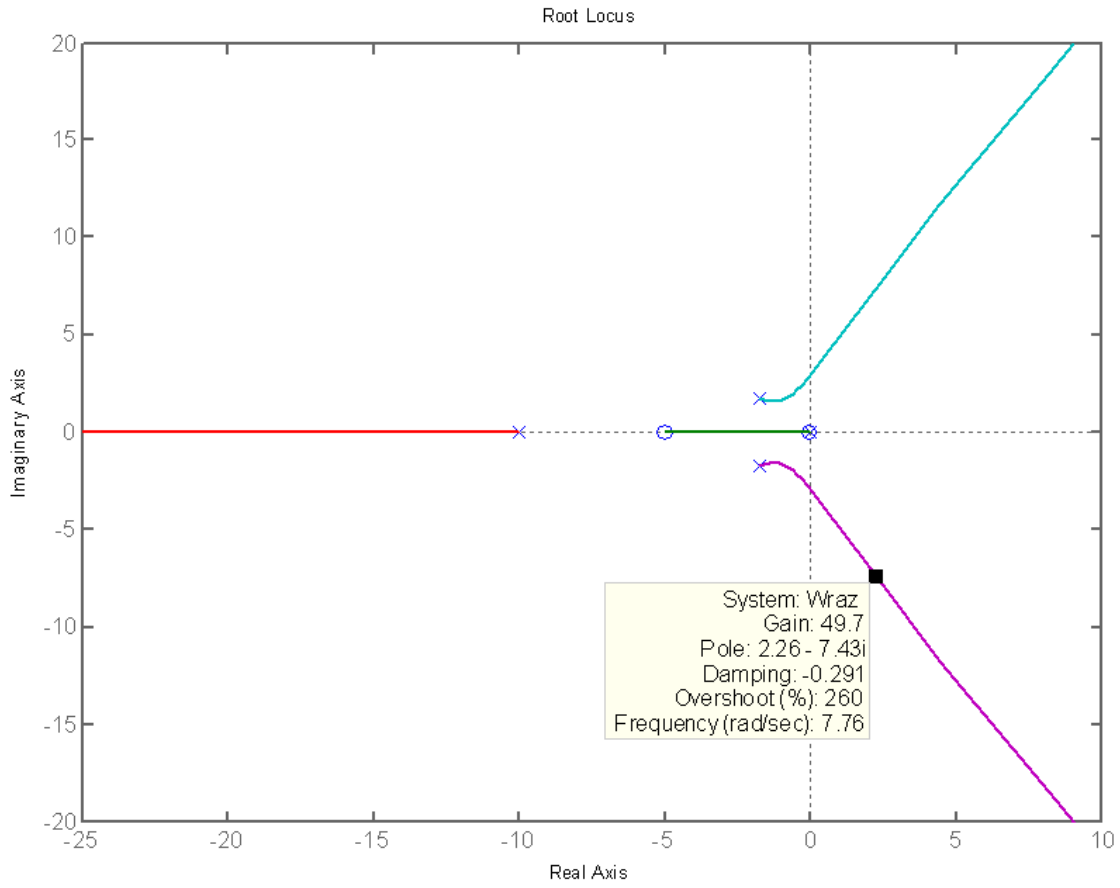


Рис. 10.2. Приклад кореневого годографа

Кореневий годограф, показаний на рис. 10.2, був побудований за допомогою команди `rlocus`, що застосовується до характеристичного рівняння виду (10.2). Порядок використання команди `rlocus` ілюструє наступний скріпт:

```
Wr=tf([1.5 0.01],[1 0]);
Wpr=tf(-1,[0.1 1]);
Ws=tf([-1 -5],[1 3.5 6 0]);
Wraz=Wr*Wpr*Ws;
rlocus(Wraz)
```

Викликавши контекстне меню і клацнувши у вікні годографа правою кнопкою миші, можна додати або видалити сітку, змінити масштаб або викликати **Редактор Властивостей** (**Property Editor**) для настроювання графіка. Клацнувши також на будь-якій лінії

годографа можна одержати інформацію про значення коефіцієнта підсилення, полюса, коефіцієнта демпфірування, перерегулювання і частоти в обраній точці (рис. 10.2).

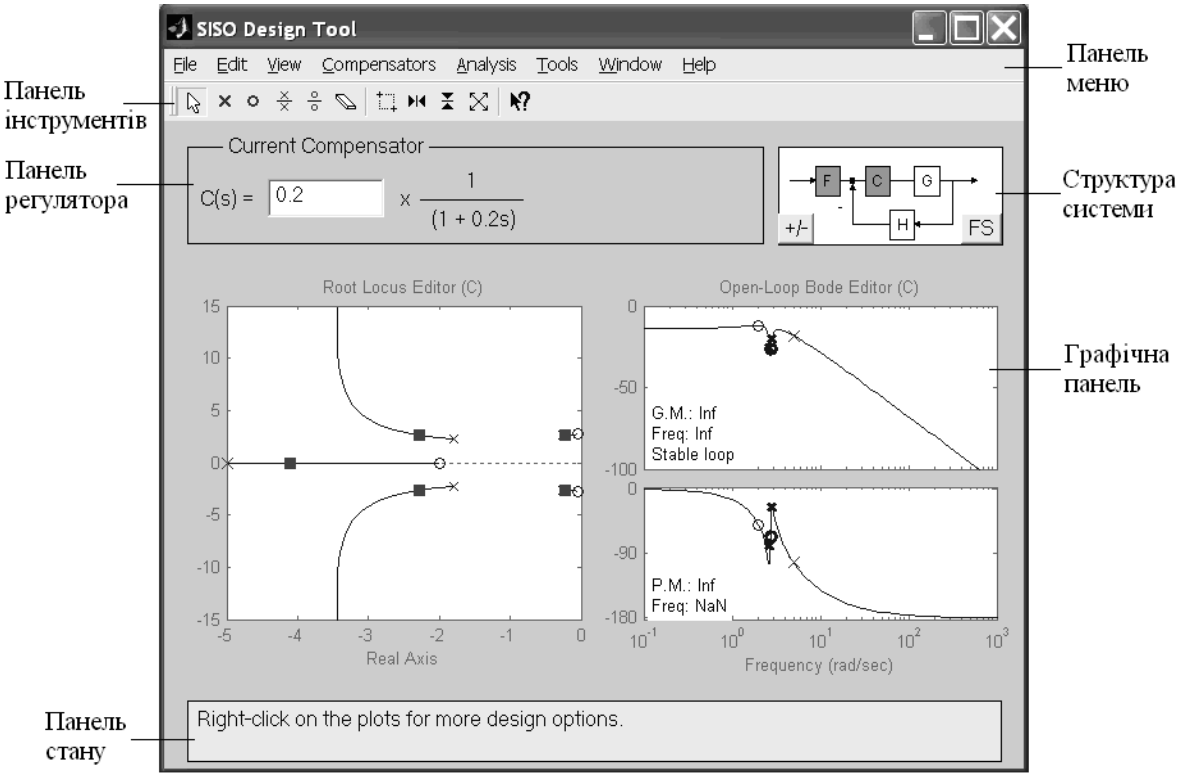


Рис. 10.3. Вікно SISO-Design Tool

Для побудови кореневих годографів зручніше використовувати GUI-інтерфейс SISO-Design Tool (GUI - графічний інтерфейс користувача, скорочення від англ. *graphical user interface*). SISO-Design Tool призначений для аналізу і синтезу одномірних лінійних систем автоматичного керування (SISO - Single Input/Single Output - Один Вхід/Один Вихід).

Запуск графічного інтерфейсу SISO-Design Tool (рис. 10.3) здійснюється командою `sisotool` або вибором **Start → Toolboxes → Control System → SISO Design Tool**.

Вікно графічного редактора (рис. 10.3) має панель меню, панель інструментів і основне робоче вікно. У верхній частині робочого вікна можна задавати структуру системи керування та коефіцієнт підсилення регулятора K. У середній частині розташована графічна панель, а внизу - невелика панель стану, у якій відображаються підказки.

Вибір необхідної структури системи керування робиться за допомогою кнопки FS (Feedback Structure – структура системи зі зворотним зв'язком) у правому верхньому куті SISO-Design Tool; тип зворотного зв'язку вибирається натисканням кнопки +/-. При цьому в спеціальному вікні відображається обрана структура, де F - попередній фільтр, C - регулятор з коефіцієнтом підсилення *K*, G - об'єкт керування, H - датчик.

Розглянемо у якості прикладу процедуру побудови в SISO-Design Tool структури системи керування числом обертів двигуна внутрішнього згоряння, зображену на рис. 9.2. Для цього в робочій області MATLAB ще раз задамо передаточні функції

```
>> W1=tf(1.1,[1 1]);
>> Wdv=tf(1,[3 1]);
>> Wds=tf(1,[0.4 1]);
```

Для завантаження в SISO-Design Tool даних з робочого простору MATLAB необхідно у вікні графічного інтерфейсу використати меню **File→Import**, у результаті чого з'являється діалогове вікно **Import System Data** (рис. 10.4).

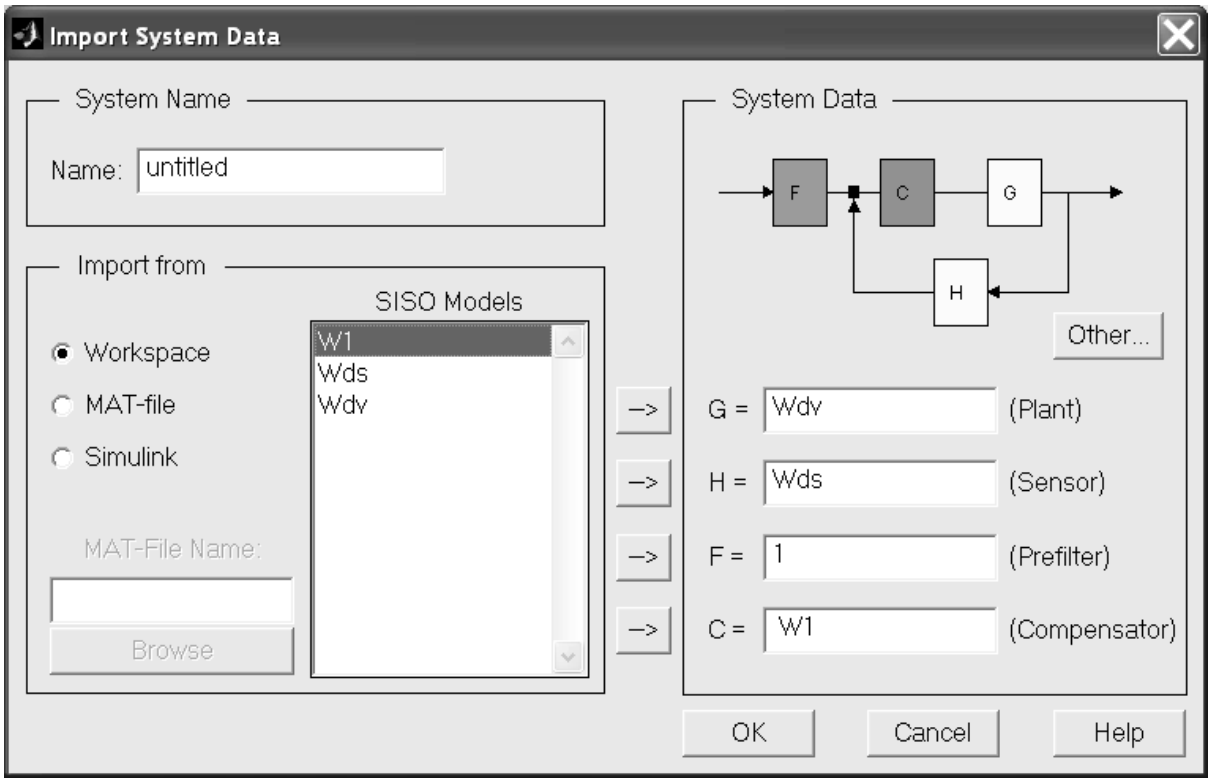


Рис. 10.4. Діалогове вікно **Import System Data**

У вікні SISO Models відображаються моделі, які задані в робочій області MATLAB (можливий також імпорт моделей з MAT-файлу і Simulink). Для того, щоб поставити яку-небудь модель ланки у відповідність елементам схеми F, C, G або H, необхідно у вікні SISO Models вибрати потрібну модель і натиснути кнопку --> напроти відповідної позиції. Якщо якого-небудь пристрою на схемі немає, то йому привласнюється значення 1. Тут же натисканням кнопки Other... можна змінити структуру системи. Таким чином, у результаті імпортування даних можна одержати необхідну схему системи керування.

За замовчуванням SISO Design Tool відображає кореневий годограф і діаграми Боде (ЛАЧХ і ЛФЧХ). Змінити графіки, які треба відобразити у вікні, можна в меню **View**. Знімемо в меню **View** прапорець напроти позиції **Open-Loop Bode,** залишивши, тим самим, лише зображення кореневого годографа (рис. 10.5).

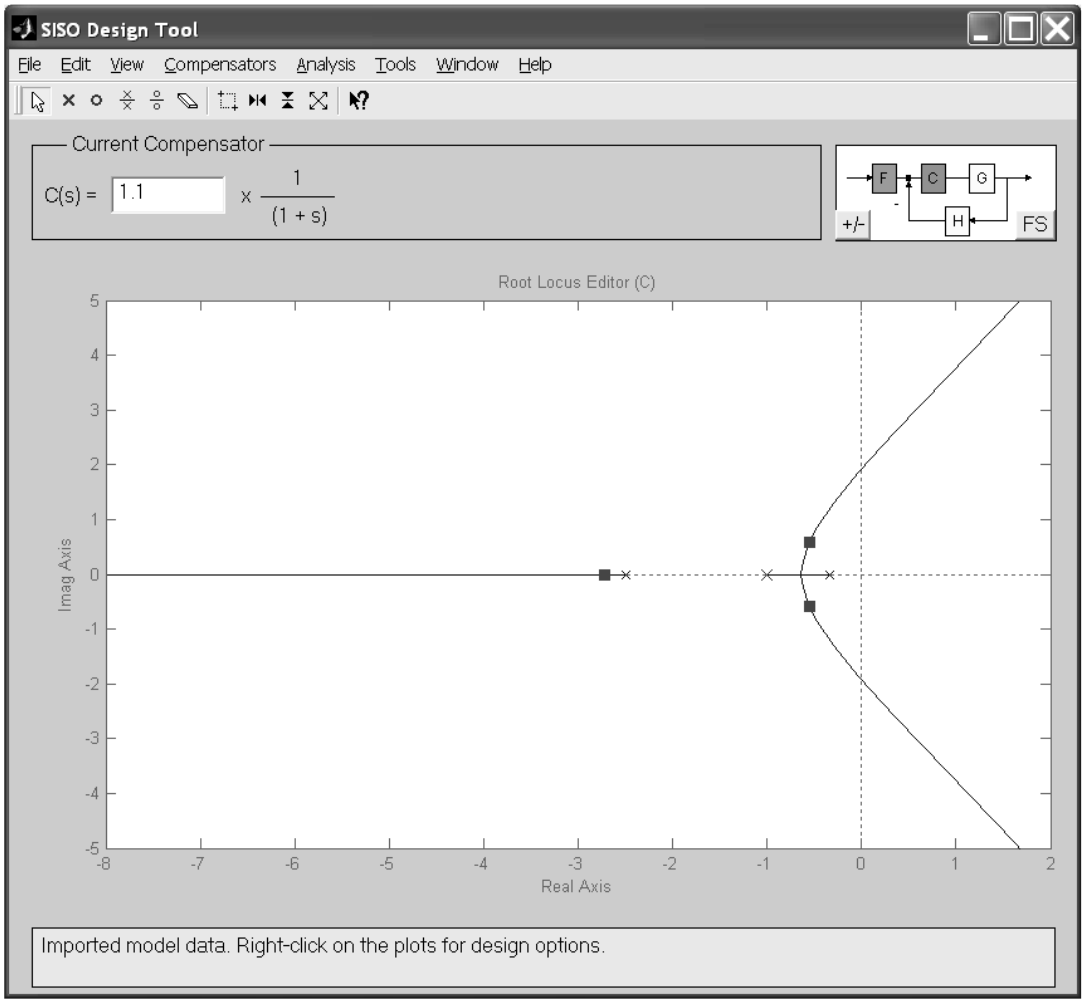


Рис. 10.5. Кореневий годограф, побудований в SISO Design Tool

Знайдемо критичний коефіцієнт підсилення $K_{кр}$, при якому система виявиться на границі стійкості. Для цього пересунемо мишею червоні квадратики, що позначають полюси замкнутої системи, по кореневому годографу до перетинання віток із мнимою віссю (рис. 10.6). Пересування курсору відбувається також при введенні значення коефіцієнта підсилення K у відповідне поле введення у верхній частині вікна SISO Design Tool.

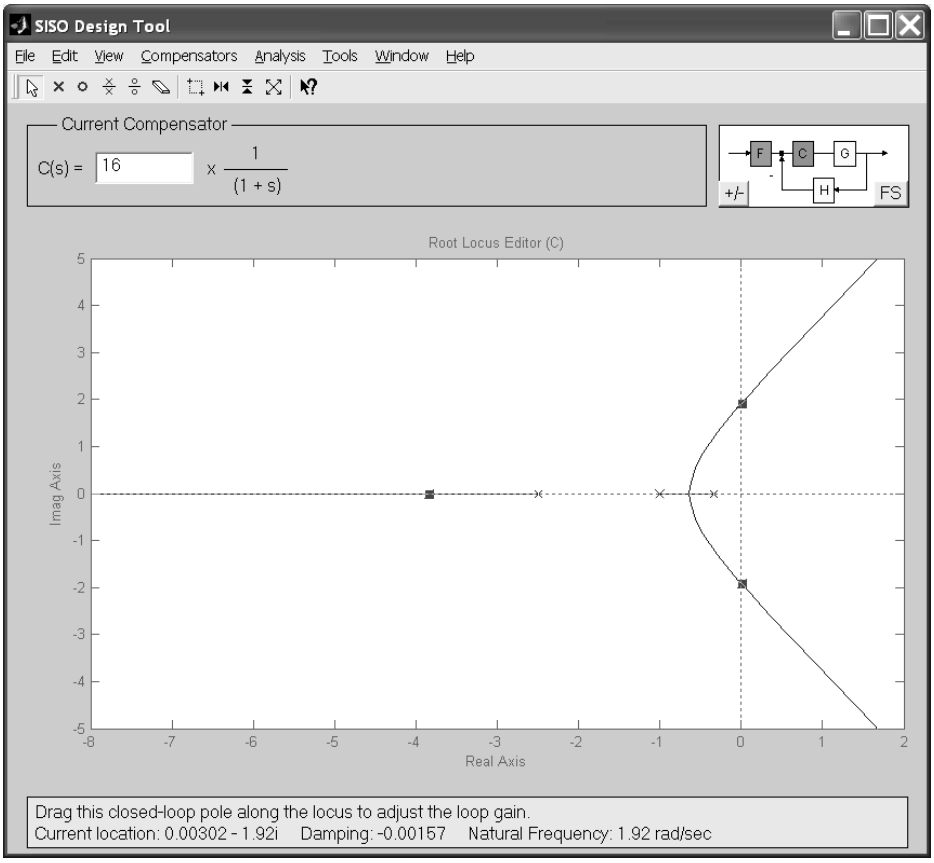


Рис. 10.6. Визначення границі стійкості системи

У нашому випадку $K_{кр} \approx 16$ (рис. 10.6). Значення виділеного полюса при цьому коефіцієнті підсилення, а також відповідну йому частоту можна переглянути на панелі стану в нижній частині інтерфейсу або вибравши в меню пункт **View→Closed-Loop Poles**. При цьому з'явиться вікно, яке зображене на рис. 10.7.

SISO Design Tool тісно пов'язаний з LTI Viewer. Це дозволяє змінювати параметри системи і відразу спостерігати результат у вікні LTI Viewer.

Побудуємо перехідну функцію замкнутої системи, наприклад, для значення $K = 0{,}5 \cdot K_{кр} = 8$.

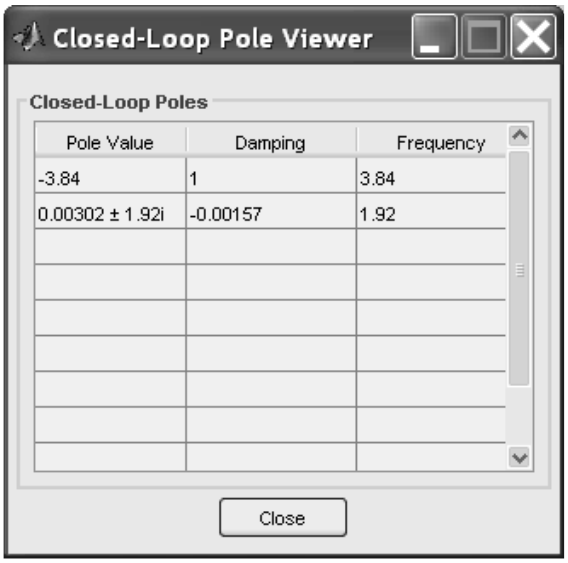


Рис. 10.7. Closed-Loop Poles Viewer

Для цього необхідно вибрати в меню пункт **Analysis → Response to Step Command**, після чого на екрані з'явиться вікно LTI Viewer for SISO Design Tool (рис. 10.8) із двома графіками: вихідним сигналом *y* (синій) і керуючим сигналом *u* (зелений).

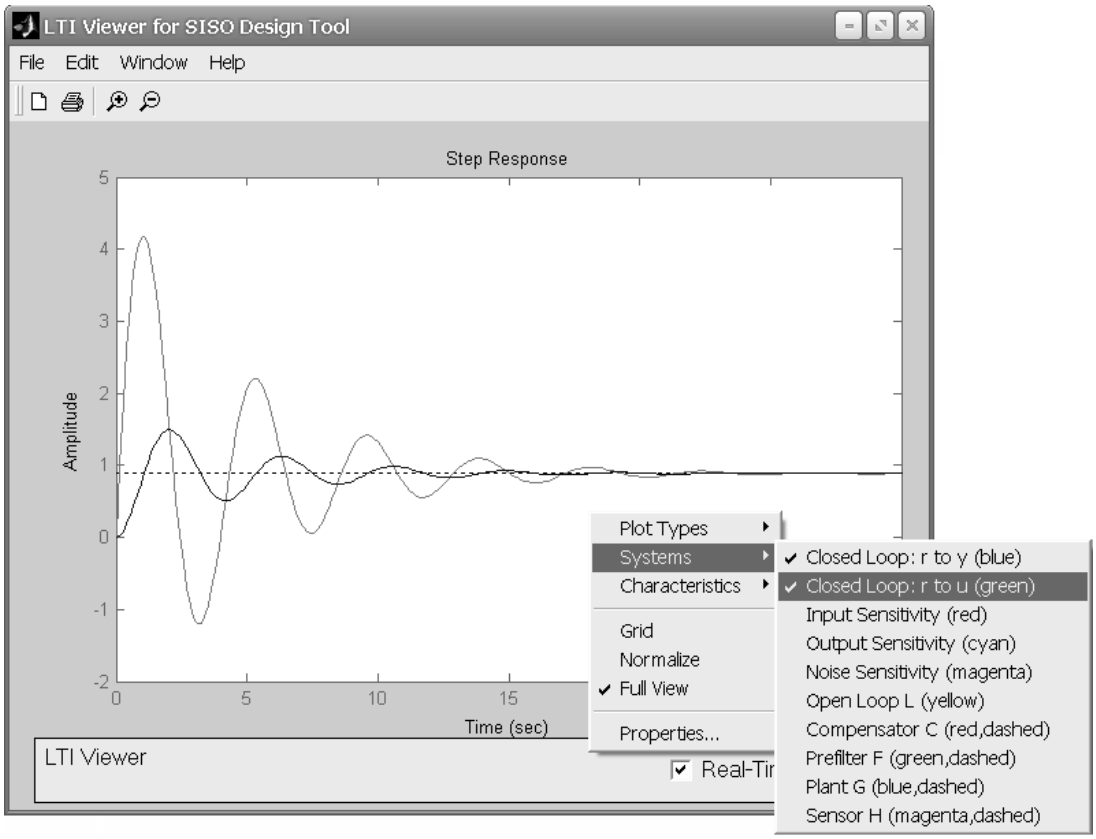


Рис. 10.8. Часові характеристики замкнутої системи при $K = 0{,}5 \cdot K_{\text{кр}}$

Щоб залишити тільки одну криву або додати ще один графік, потрібно клацнути правою кнопкою миші в графічному вікні та у контекстному меню, що з'явиться, у підменю **Systems** поставити або зняти прапорець напроти потрібної позиції. Наприклад, якщо зняти прапорець навпроти **Closed Loop: r to u (green)**, то в графічному вікні залишиться лише перехідна функція (рис. 10.9).

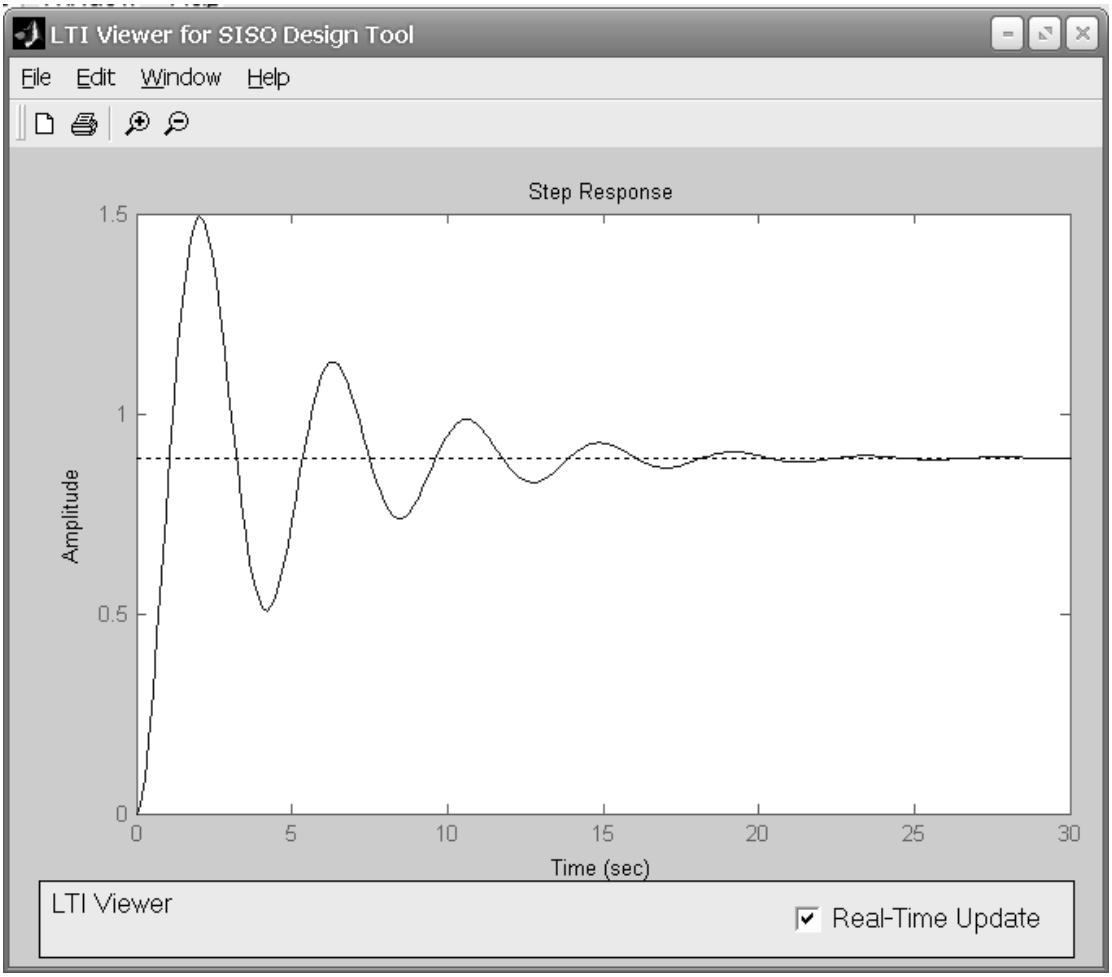


Рис. 10.9. Вид вікна LTI Viewer for SISO Design Tool при виборі позиції **Closed Loop: r to y (blue)**

Змінюючи значення K, можна побачити відповідну зміну перехідної функції або інших характеристик системи в динаміці (при встановленому прапорці **Real-Time Update** на панелі стану в правому нижньому куті). Таким чином, змінюючи значення коефіцієнта підсилення, можна не тільки визначити стійкість системи, але й підібрати бажані показники якості. Така процедура відповідає процесу синтезу П-регулятора.

Завдання для самостійної роботи

1. Передаточна функція розімкненої системи керування має вигляд

$$W(s) = \frac{K(2s+1)}{4s^2 + 2s + 1}.$$

а) За допомогою команди `rlocus` побудуйте кореневий годограф системи.

б) Визначте значення коефіцієнта підсилення K, при якому полюси замкненої системи відповідають коефіцієнту демпфування $\xi = 0{,}707$.

2. Система з одиничним зворотнім зв'язком має у розімкненому стані передаточну функцію

$$W(s) = \frac{2K}{s(2s+1)(5s+1)}.$$

а) Побудуйте кореневий годограф у SISO Design Tool.

б) Визначте значення коефіцієнта підсилення K, при якому система знаходиться на межі стійкості.

в) Визначте діапазон значень K при яких система стійка, а полюси її передаточної функції є речовинними.

г) Перевірте результати пп. *б*) та *в*) за перехідною функцією.

3. Передаточна функція розімкненої системи керування має вигляд

$$W(s) = \frac{K(2s+1)}{4s^2 + 2s + 1}.$$

а) Побудуйте кореневий годограф у SISO Design Tool.

б) Визначте значення коефіцієнта підсилення K, при якому коефіцієнт демпфування $\xi = 0{,}707$.

в) Визначте значення K, при якому система має критичне демпфування ($\xi = 1$).

г) При яких значеннях полюсів передаточної функції перехідний процес стає аперіодичним.

11. МОДЕЛЮВАННЯ СИСТЕМ У ПРОСТОРІ СТАНІВ

11.1. Основні положення

У попередніх главах ми розглянули основи аналізу і синтезу систем автоматичного керування за допомогою MATLAB. При цьому в основному ми працювали з передаточними функціями, що зв'язують вхід і вихід системи.

Для побудови передаточних функцій використовувалось перетворення Лапласа, що дозволяє перейти від диференціальних рівнянь, що описують поводження системи, до алгебраїчних рівнянь відносно комплексної змінної *s*. На підставі цього алгебраїчного рівняння ми і одержували передаточні функції. Цей метод привабливий тим, що він дозволяє використати структурні схеми для встановлення зв'язків між підсистемами.

У даній главі ми розглянемо альтернативний метод моделювання систем керування, заснований на їхньому поданні в просторі станів. Опис систем у просторі станів лежить в основі сучасної теорії керування та методів оптимізації і застосовується для дослідження і проектування нелінійних, нестаціонарних і багатомірним систем. Нагадаємо, що нестаціонарна система - це система, у якій один або більше параметрів є функціями часу. Наприклад, маса ракети змінюється по мірі витрати палива під час польоту. Багатомірна система - це система з декількома входами і виходами.

Перш ніж приступити до вивчення можливостей MATLAB для аналізу і синтезу систем керування, розглянемо поняття «стан системи» і «простір станів».

11.2 Опис систем у просторі станів

Розглянемо багатомірну систему керування (рис. 11.1).

Під *станом* системи розуміється мінімально-необхідний набір змінних величин системи $x_1, x_2,...,x_n$, здатних однозначно і єдиним образом визначити положення системи в будь-який момент часу *t*. Це означає, що якщо в момент часу t_0 відомі початкові значення

$[x_1(t_0),\ x_2(t_0),\ ...,\ x_n(t_0)]$ і вхідні сигнали $u_1(t),\ldots,u_m(t)$, то цієї інформації досить, щоб визначити майбутні значення всіх змінних стану і вихідних змінних.

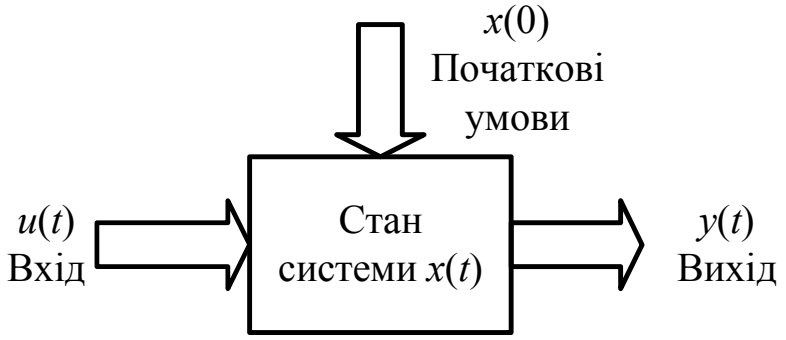


Рис. 11.1. Модель багатомірної системи керування

Змінні стану описують поведінку системи в майбутньому, якщо відомі поточний стан, зовнішні впливи і рівняння динаміки системи. Простим прикладом є електрична лампочка. Вона може бути в одному із двох положень - «увімкнена» або «вимкнена», тому її стану відповідає одне з двох можливих значень. Якщо відомо, у якому стані (положенні) перебуває лампочка в момент часу t_0, і якщо ми прикладаємо до вимикача вплив, то можна визначити майбутній стан лампочки.

Якщо до змінних стану $x_i(t)$ додати ще одну $y_0 = t$, то ці $(n+1)$ координат утворять *простір станів* системи R^n.

Відомо, що динаміка лінійної системи описується звичайним рівнянням n-го порядку. За допомогою змінних стану можна перейти до системи з n диференціальних рівнянь першого порядку:

$$\begin{cases} \dfrac{dx_1}{dt} = a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n + b_1u_1 + ... + b_{1m}u_m \ ; \\ \dfrac{dx_2}{dt} = a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n + b_2u_1 + ... + b_{2m}u_m \ ; \\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots; \\ \dfrac{dx_n}{dt} = a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n + b_nu_1 + ... + b_{nm}u_m \ . \end{cases} \qquad (11.1)$$

Звичайно рівняння стану записують у матричній формі, що значно спрощує роботу з ними

$$\frac{d}{dt}\cdot\begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}\cdot\begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} + \begin{bmatrix} b_{11} & \dots & b_{1m} \\ b_{21} & \dots & b_{2m} \\ \dots & \dots & \dots \\ b_{n1} & \dots & b_{nm} \end{bmatrix}\cdot\begin{bmatrix} u_1 \\ \dots \\ u_m \end{bmatrix}. \quad (11.2)$$

У такому вигляді модель вже цілком придатна для комп'ютерного аналізу.

Вектор-стовпець, що складається із змінних стану, називається *вектором стану* і позначається **x**, де напівжирне креслення символу означає вектор. Вектор вхідних сигналів позначається як **u**. Тоді систему (11.1) можна описати в компактному вигляді диференціальним рівнянням стану

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \quad (11.3)$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}. \quad (11.4)$$

Ці рівняння несуть великий обсяг інформації щодо динамічних властивостей системи с m входами і r виходами. Рівняння (11.3) часто називають просто *рівнянням стану*. Рівняння стану зв'язує швидкість зміни стану системи із самим станом і вхідними сигналами. Вихідні сигнали лінійної системи пов'язані зі змінними стану і вхідними сигналами *рівнянням виходу* (11.4). У рівняннях (11.3) - (11.4):

x – вектор стану розмірності ($n\times1$), компонентами якого є змінні стану системи n - го порядку;

u - вектор-стовпець керуючих впливів, ($m\times1$),

A – матриця коефіцієнтів системи, ($n\times n$);

B - матриця входу, ($n\times m$);

y - вектор-стовпець вихідних сигналів, ($p\times1$);

C – матриця виходу ($p\times n$);

D – матриця обходу, що визначає пряму залежність виходу від входу. Зазвичай **D**=0, тому що у фізичних системах у всіх каналах між входами і виходами присутні динамічні ланки. Якщо **D**≠0, то принаймні один прямий шлях від входу до виходу представлений звичайним коефіцієнтом підсилення.

Якщо в системи тільки один вхід, то матриця **B** має вигляд стовпця, а вектор **u** перетворюється в скалярну змінну. Якщо ж система має лише один вихід, то вектор **y** перетворюється в скалярну змінну, а матриця **C** приймає вид рядка. Розмірність матриць **A**, **B**, **C**, **D** зручно визначати з рис. 8.1.

11.3 Приклади побудови моделей систем у просторі станів

Поняття про змінні стану, що описують динамічну систему, можна проілюструвати на прикладі механічної системи «маса-пружина» із загасанням, зображеної на рис. 11.2.

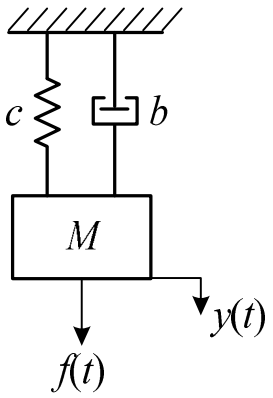


Рис. 11.2. Системи «маса-пружина»

Число змінних стану, обраних для опису системи, повинне бути по можливості мінімальним, щоб серед них не було зайвих. Для даної системи цілком достатньо мати дві змінні стану - положення маси $y(t)$ і швидкість її руху $dy(t)/dt$. Таким чином, ми приймемо в якості змінних наступні параметри:

$$x_1(t) = y(t) \text{ і } x_2(t) = \frac{dy(t)}{dt} \tag{11.5}$$

Диференціальне рівняння, що описує поведінку системи, має вигляд

$$M\frac{d^2y(t)}{dt^2} = f(t) - b\frac{dy(t)}{dt} - cy(t), \tag{11.6}$$

а передаточна функція дорівнює

$$W(s) = \frac{y(s)}{f(s)} = \frac{1}{Ms^2 + bs + c}. \qquad (11.7)$$

Запишемо рівняння (11.6) з урахуванням введених вище змінних стану:

$$\dot{x}_2(t) = -\frac{b}{M}x_2(t) - \frac{c}{M}x_1(t) + \frac{1}{M}f(t). \qquad (11.8)$$

З урахуванням (11.5), одержимо:

$$\begin{cases} \dot{x}_1(t) = x_2(t); \\ \dot{x}_2(t) = -\frac{c}{M}x_1(t) - \frac{b}{M}x_2(t) + \frac{1}{M}f(t); \\ y(t) = x_1(t), \end{cases} \qquad (11.9)$$

або в матричній формі:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{c}{M} & -\frac{b}{M} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{M} \end{bmatrix} \cdot f(t);$$
$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \qquad (11.10)$$

Ці рівняння, по суті, описують поведінку системи в термінах швидкості зміни кожної змінної стану.

Змінні стану, що описують систему, не є єдиними, і завжди можна вибрати альтернативну комбінацію таких змінних. Наприклад, для такої системи другого порядку, як маса-пружина в якості змінних стану можна вибрати будь-які дві лінійно незалежні комбінації $x_1(t)$ і $x_2(t)$.

У якості ще одного прикладу побудови моделі в просторі станів розглянемо двигун постійного струму (рис. 11.3). Двигун перетворює електричну енергію постійного струму в механічну енергію обертового руху. Основна частина моменту, що створюється ротором (якорем) двигуна, використовується для керування зовнішнім інерційним навантаженням.

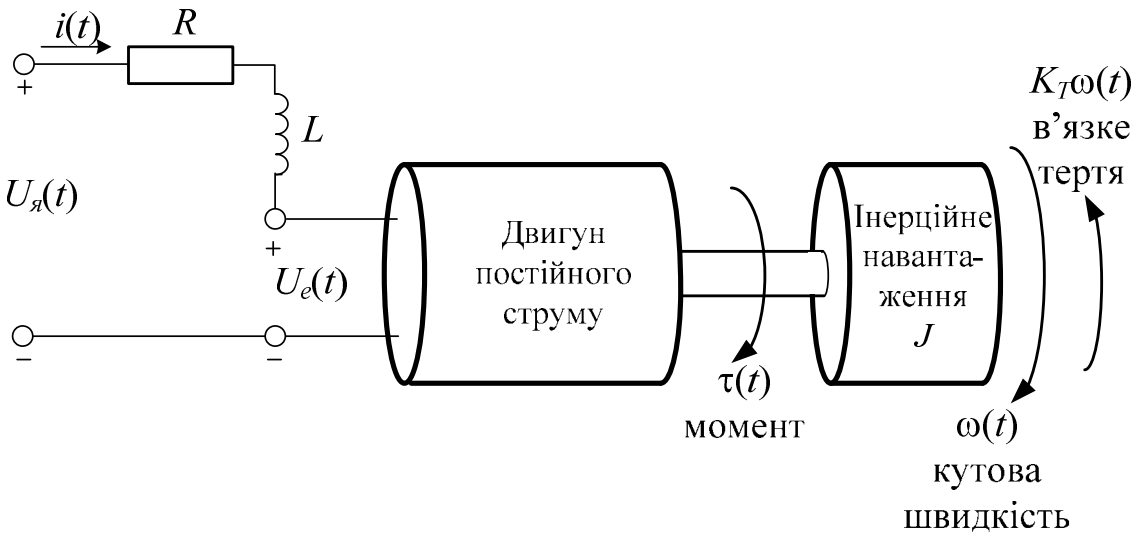


Рис. 11.3. Двигун постійного струму з навантаженням

Побудуємо математичну модель двигуна. Вхідною величиною є напруга $U_я$, що подається на якір, а в якості вихідної приймемо кутову швидкість $\omega(t)$ інерційної маси. Будемо розглядати ідеалізовану модель двигуна, зневажаючи такими другорядними ефектами, як гістерезис і спадання напруги на щітках, а також будемо вважати магнітне поле постійним. Опір та індуктивність кола якоря дорівнюють відповідно R і L.

Обертаючий момент τ на валу двигуна пропорційний току $i(t)$, що індуцується прикладеною напругою

$$\tau(t) = K_я\, i(t), \qquad (11.10)$$

де $K_я$ – постійна якоря, що залежить від фізичних властивостей двигуна, таких як напруженість магнітного поля, числа витків дроту в котушці та ін.

ПротивоЕРС $U_e(t)$, що виникає в обмотці якоря в результаті його обертання в магнітному полі, пропорційна швидкості обертання вала:

$$U_e(t) = K_д \cdot \omega(t), \qquad (11.11)$$

де $K_д$ – постійний коефіцієнт, що також залежить від певних фізичних властивостей двигуна.

Опис механічної частини двигуна ґрунтується на законі Ньютона:

$$J\frac{d\omega(t)}{dt} = \sum \tau_i = -K_T\omega(t) + K_{\ddot{y}}i(t), \qquad (11.12)$$

де $K_T\omega(t)$ – лінійна апроксимація для в'язкого тертя.

Остаточно, електрична частина двигуна описується наступним рівнянням

$$U_{\ddot{y}}(t) - U_e(t) = L\frac{di(t)}{dt} + Ri(t), \qquad (11.13)$$

або

$$U_{\ddot{y}}(t) = L\frac{di(t)}{dt} + Ri(t) + K_{\ddot{a}}\omega(t). \qquad (11.14)$$

Остаточно

$$\frac{di(t)}{dt} = -\frac{R}{L}i(t) - \frac{K_{\ddot{a}}}{L}\omega(t) + \frac{1}{L}U_{\ddot{y}}(t). \qquad (11.15)$$

З вираження (11.12) одержимо:

$$\frac{d\omega(t)}{dt} = -\frac{1}{J}K_T\omega(t) + \frac{1}{J}K_{\ddot{y}}i(t). \qquad (11.16)$$

Тепер можна представити модель двигуна у формі простору станів. Станами системи є струм i та кутова швидкість ω. Прикладена напруга $U_я$ є входом системи, а кутова швидкість ω - виходом.

Стандартна форма рівнянь стану має вигляд (11.3) - (11.4). Стосовно до нашої системи

$$\frac{d}{dt}\begin{bmatrix} i \\ \omega \end{bmatrix} = \begin{bmatrix} -\frac{R}{L} & -\frac{K_{\ddot{a}}}{L} \\ \frac{K_{\ddot{y}}}{J} & -\frac{K_T}{J} \end{bmatrix} \cdot \begin{bmatrix} i \\ \omega \end{bmatrix} + \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix} \cdot U_{\ddot{y}}(t);$$
$$y(t) = \begin{bmatrix} 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} i \\ \omega \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} \cdot U_{\ddot{y}}(t). \qquad (11.17)$$

11.4 Побудова MATLAB - моделі в просторі станів

Порядок моделювання системи керування в просторі станів розглянемо на прикладі двигуна постійного струму.

Отже, ми маємо систему диференціальних рівнянь, які описують нашу систему. Використовуючи прості команди Control System Toolbox ми можемо побудувати в MATLAB модель двигуна постійного струму в просторі станів, а також перетворювати один вид подання в іншій. Задамося, спочатку, номінальними значеннями різних параметрів двигуна постійного струму

```
R= 2.0; % Оми
L= 0.5; % Генри
Km = 0.015; % Постійна якоря Кя
Kb = 0.015; % Постійна двигуна Кд
Kf = 0.2; % Кт, Нмс
J= 0.02; % кг.м^2/c^2
```

Задавшись цими значеннями, можна побудувати модель двигуна в просторі станів. Для цього використаємо функцію `ss` (див. п. 8.5).

```
A = [-R/L -Kb/L; Km/J -Kf/J];
B = [1/L; 0];
C = [0 1];
D = [0];
Dc_ss = ss(A,B,C,D)
```

У результаті виконання останньої команди видаються наступні значення:

```
a =
             x1        x2
    x1       -4    -0.03
    x2     0.75       -10

b =
             u1
    x1        2
    x2        0
```

```
c =
               x1        x2
    y1          0         1

d =
               u1
   y1           0
```

11.5 Перетворення між поданнями моделі

MATLAB дозволяє дуже просто перейти від одного виду подання інформації до іншого. Наприклад, використовуючи команду `tf`, можна перейти від подання моделі в просторі станів до передаточної функції:

```
Dc_tf = tf(Dc_ss)

transfer function:
       1.5
------------------
s^2 + 14 s + 40.02
```

За допомогою функції `zpk` можна перетворити модель у вигляді передаточної функції або простору станів в zero/pole/gain формат. Так, для двигуна постійного струму:

```
Dc_zpk = zpk(Dc_ss)

Zero/pole/gain:
        1.5
-------------------
(s+4.004) (s+9.996)
```

Для побудови часових і частотних характеристик системи зручно користуватися графічним інтерфейсом LTI Viewer. Рекомендації роботи з LTI Viewer наведені в главі 10. Так, наприклад, на рис. 11.4 показане вікно із чотирма характеристиками двигуна постійного струму з навантаженням, отримані в LTI Viewer. Вид вікна і тип характеристик був настроєний у вікні **Plot Configurations** (рис. 11.5), викликаному командою **Edit → Plot Configurations…**

Після цього характеристики були експортовані в стандарне графічне вікно MATLAB вибором меню **File → Print to Figure**.

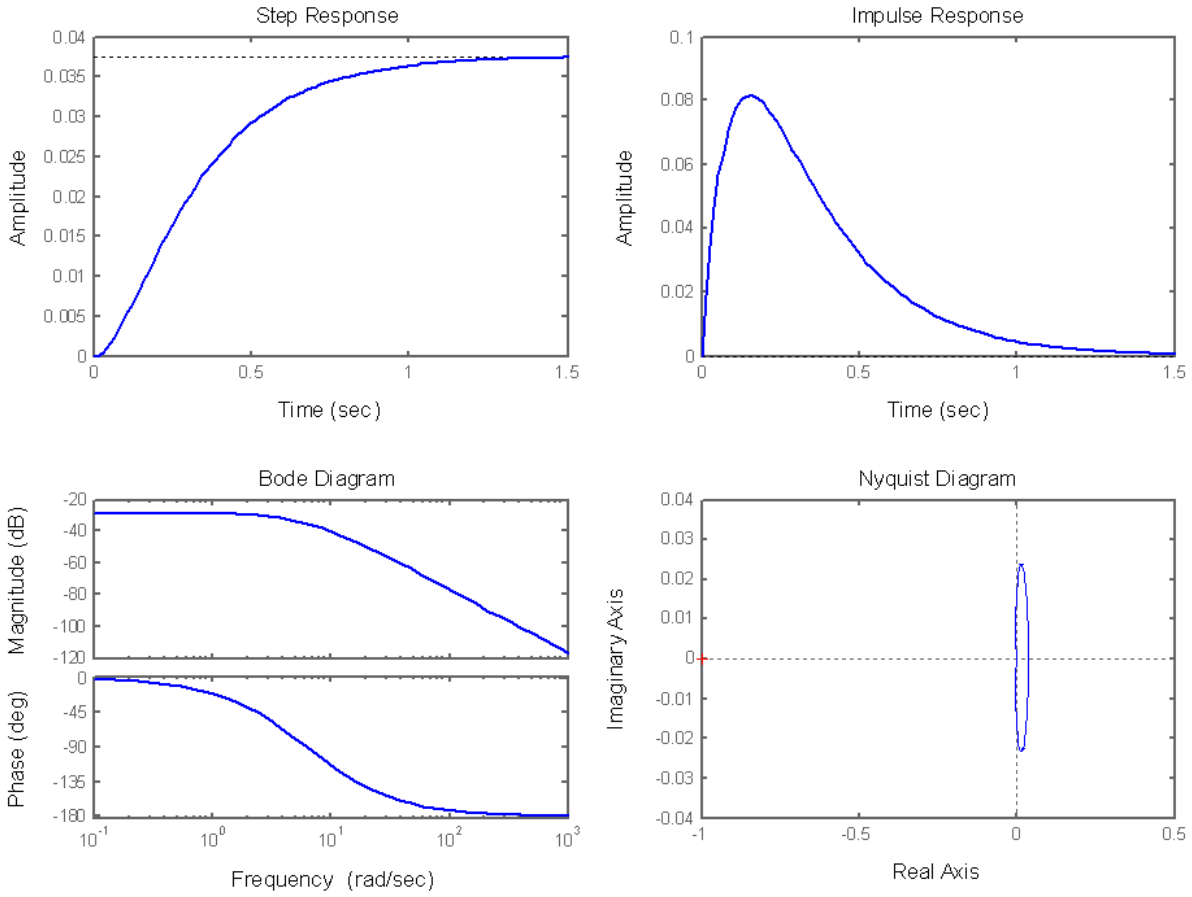


Рис. 11.4. Характеристики двигуна з навантаженням, побудовані в LTI Viewer

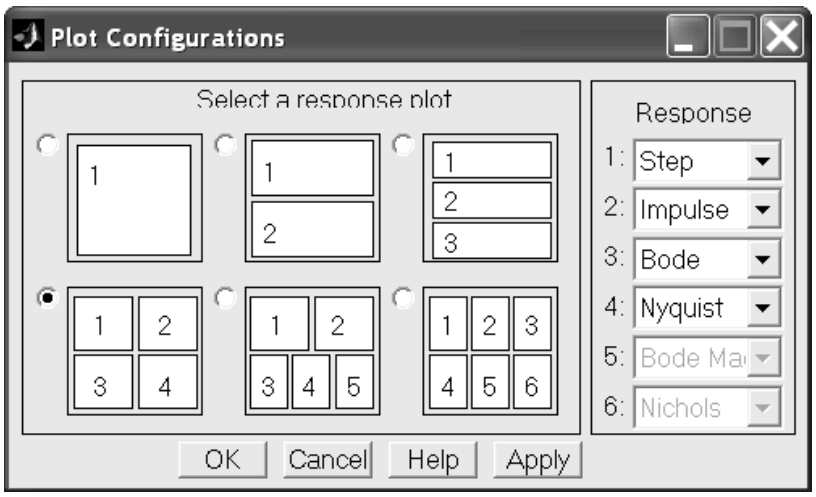
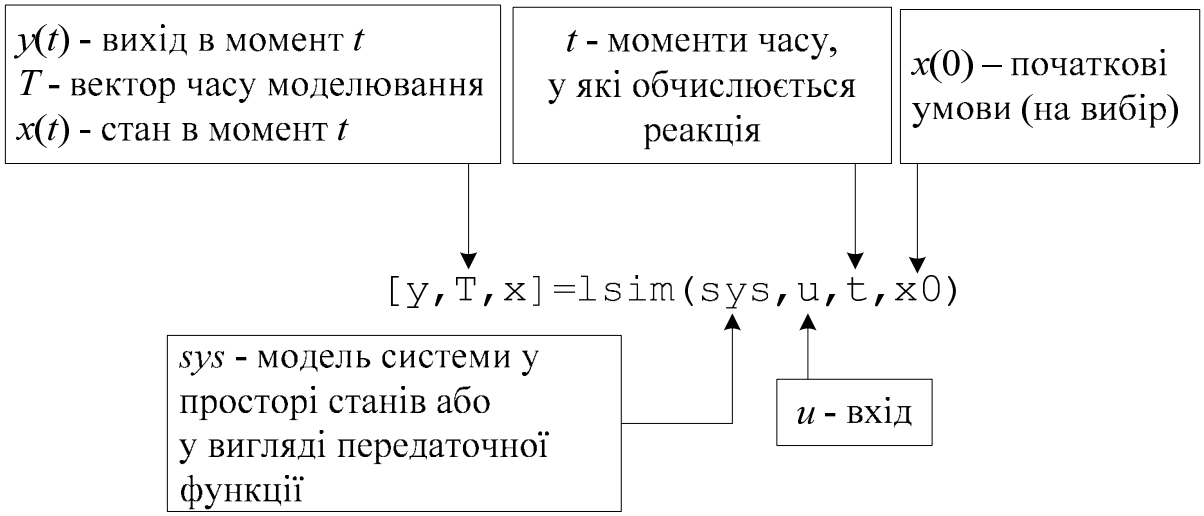


Рис. 11.5. Настроювання LTI Viewer у вікні **Plot Configurations**

Зупинимося докладніше на функції `lsim`, що також дозволяє побудувати часові характеристики системи. Її достоїнством є те, що вона допускає як завдання ненульових початкових умов, так і довільної вхідної функції:



Нижче представлений скріпт, що використовує функцію `lsim` для визначення повединку двигуна при ненульових початкових умовах і відсутності керуючого впливу.

```
x0=[1;1];%Початкові умови
t=0:0.01:1;%Вектор часу
u=0*t;%Керування дорівнює нулю
[y,T,x]=lsim(Dc_ss,u,t,x0);
subplot(1,2,1),plot(T,x(:,1));
xlabel('Час, с'),ylabel('X_1')
subplot(1,2,2),plot(T,x(:,2))
xlabel('Час, с'),ylabel('X_2')
```

Результат виконання цього скріпта представлений на рис. 11.6.

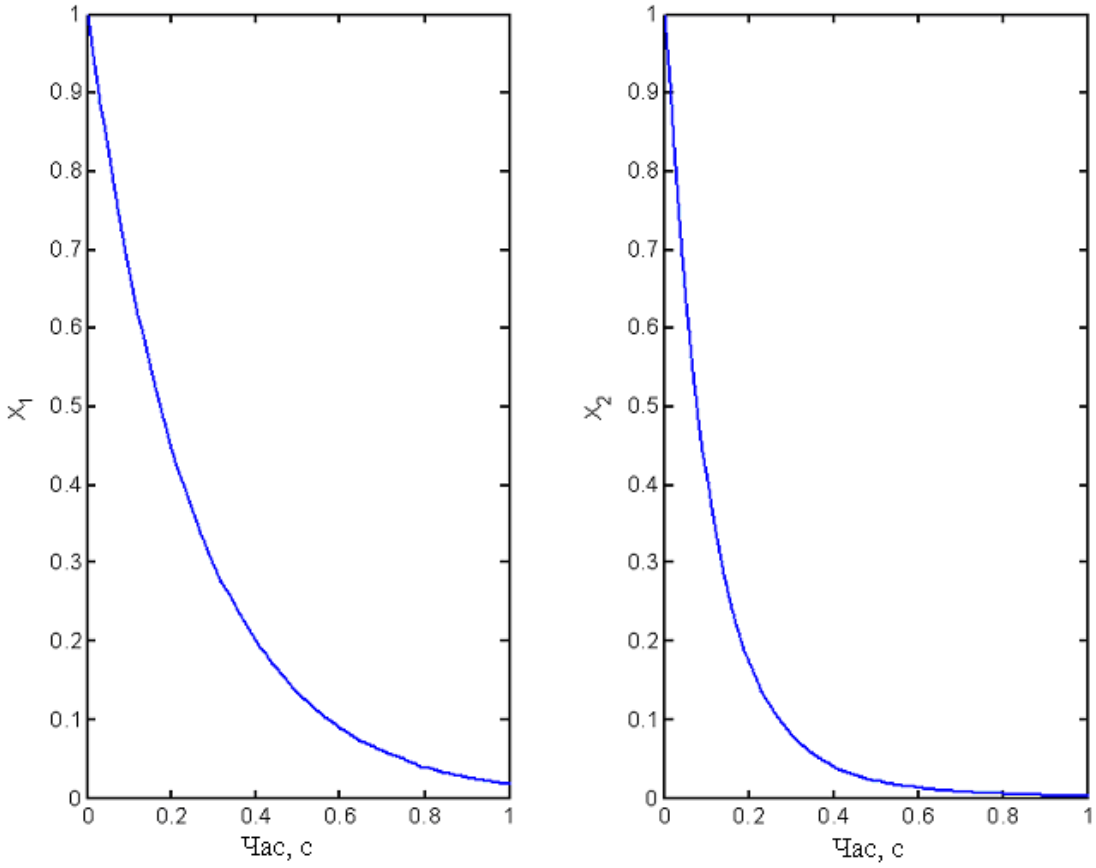


Рис 11.6. Часові характеристики двигуна постійного струму

Завдання для самостійної роботи

1. Задані моделі розімкнених систем у вигляді наступних передаточних функцій

$$W(s)=\frac{5}{2s+1};\ W(s)=\frac{4s+2}{s^2+2s+1};\ W(s)=\frac{2}{0,3s^3+1,4s^2+4s+1}.$$

а) За допомогою команди `ss` знайдіть моделі замкнених систем у просторі станів.

б) Для кожного випадку побудуйте у одному графічному вікні перехідні функції та переконайтеся, що ці форми надання моделей еквівалентні.

2. Модель системи у просторі станів описується наступними матрицями

$$\mathbf{A} = \begin{bmatrix} 0 & -2 \\ 1 & -3 \end{bmatrix},\ \mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix},\ \mathbf{C} = \begin{bmatrix} 1 & 0 \end{bmatrix},\ \mathbf{D} = \begin{bmatrix} 0 \end{bmatrix}.$$

а) Побудуйте модель системи у MATLAB.

б) За допомогою команди `tf` перетворіть її в модель у вигляді передаточної функції.

в) За допомогою графічного інтерфейсу LTI Viewer побудуйте перехідну та імпульсну перехідну функції системи.

3. Система керування описується наступними рівняннями

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -3 & -5 & -4 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u \text{ та } y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \mathbf{x}.$$

а) Побудуйте графіки вільного руху змінних $x_1(t)$, $x_2(t)$ та $x_3(t)$ при початкових умовах $\mathbf{x}(0) = [1\ 2\ 0]^T$ при $0 \le t \le 10$.

б) За допомогою команди `tf` визначить передаточну функцію системи.

4. Система керування у розімкненому стані описується наступними рівняннями

$$\dot{\mathbf{x}} = \begin{bmatrix} 3 & -0,5 & 0 \\ 4 & 0 & 0 \\ 0 & 2 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} u \text{ та } y = \begin{bmatrix} 1 & -0,5 & -0,25 \end{bmatrix} \mathbf{x}.$$

а) За допомогою команди `feedback` побудуйте модель у просторі станів для замкненої системи з одиничним зворотнім зв’язком.

б) Методом кореневого годографу знайдіть максимальний коефіцієнт підсилення системи, при якому вона ще буде стійкою.

12. ДИСКРЕТНІ СИСТЕМИ АВТОМАТИЧНОГО КЕРУВАННЯ

12.1 Поняття дискретної системи

Дотепер ми розглядали безперервні системи автоматичного керування. Однак в останні десятиліття безперервні системи витісняються дискретними або точніше, безперервно-дискретними системами. Вони складаються як з елементів, поведінка яких описується безперервними функціональними залежностями, так і з елементів, поведінка яких описуються дискретними функціями часу. Більшість з таких систем керування - цифрові системи, у яких регулювання здійснюється за допомогою ЕОМ (рис. 12.1).

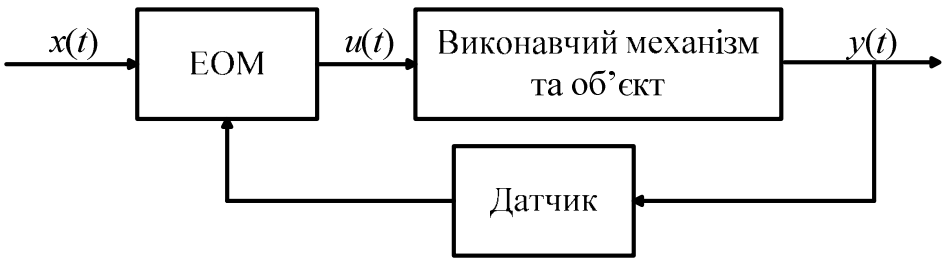


Рис. 12.1. Система керування з цифровим регулятором

ЕОМ по певній програмі обробляє сигнал помилки, представлений у цифровій формі; керуючий вплив також являє собою цифровий сигнал. Оскільки переважна більшість об'єктів і виконавчих механізмів - безперервні пристрої, а ЕОМ може приймати і видавати інформацію лише в дискретні моменти часу, то вхідним інтерфейсом ЕОМ є аналого-цифровий перетворювач (АЦП), а вихідним інтерфейсом – цифро-аналоговий перетворювач (ЦАП) (рис. 12.2).

АЦП перетворює аналогові вхідні сигнали (задаючі впливи, сигнал помилки, сигнали зворотного зв'язку з датчиків) у цифрову форму (двійковий код). Звичайно АЦП виконує це перетворення періодично з деяким інтервалом T, що називається *періодом квантування* (sampling time). Таким чином, з безперервного сигналу вибираються дискретні значення, які можна представити у вигляді числової послідовності $\varepsilon(0)$, $\varepsilon(T)$, $\varepsilon(2T)$ і т.д. або $\varepsilon[k] = \varepsilon(kT)$, де $k = 0,1,2\ldots$ Цей процес називається *квантуванням*.

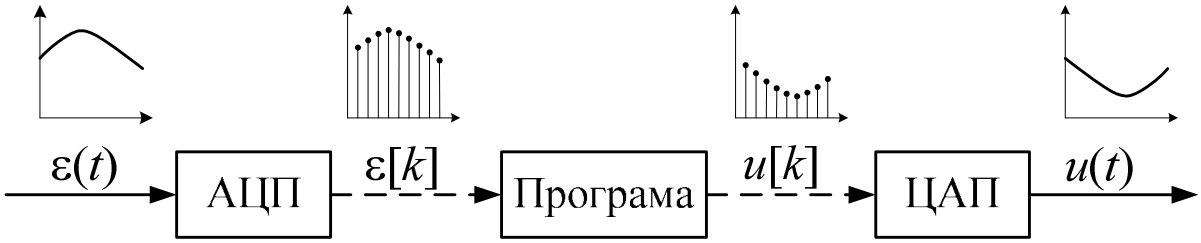


Рис. 12.2. Блок-схема цифрового комп'ютера

ЕОМ відповідно до закладеної програми перетворює вхідну числову послідовність $\{\varepsilon[k]\}$ у керуючу послідовність $\{u[k]\}$. По цій послідовності ЦАП відновлює безперервний сигнал керування $u(t)$. Найчастіше ЦАП працює з тим же періодом, що і АЦП на вході ЕОМ. Однак для розрахунку чергового керуючого сигналу потрібно якийсь час, через це виникає так називане *обчислювальне запізнювання*. На практиці прийняте це запізнювання відносити до безперервної частини системи та вважати, що АЦП і ЦАП працюють не тільки синхронно (з однаковим періодом), але й синфазно (одночасно).

Існує кілька способів відновлення безперервного сигналу. У найпростішому випадку ЦАП, одержавши новий керуючий сигнал від ЕОМ, зберігає його (фіксує) протягом періоду квантування T до появи наступного значення вхідного сигналу:

$$u(t) = u[t],\ k \le t < (k+1)T. \qquad (12.1)$$

Такий елемент називається *екстраполятором нульового порядку* (англ. *zero-order hold, ZOH*) або *фіксатором*, його робота ілюструється на рис. 12.3.

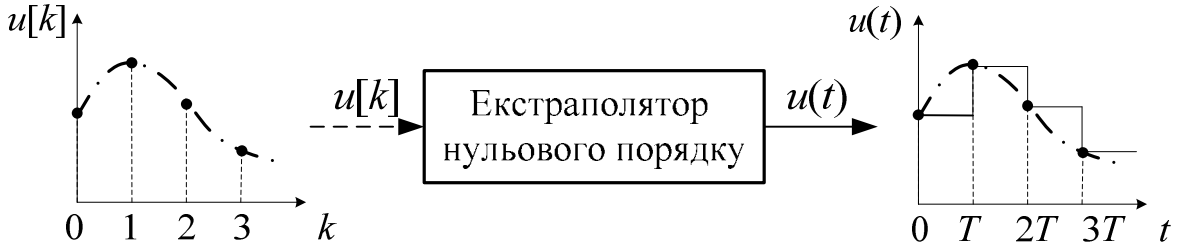


Рис. 12.3. Екстраполятор нульового порядку

Передаточна функція екстраполятора нульового порядку:

$$H_0(s) = \frac{1 - e^{-Ts}}{s}. \qquad (12.2)$$

Цей спосіб відновлення даних легко реалізується в ЦАП і найбільш часто використовується на практиці.

Іноді використовують *екстраполятор першого порядку* (англ. *first-order hold*, *FOH*), що виконує лінійну екстраполяцію на основі двох попередніх значень дискретного сигналу (рис. 12.4).

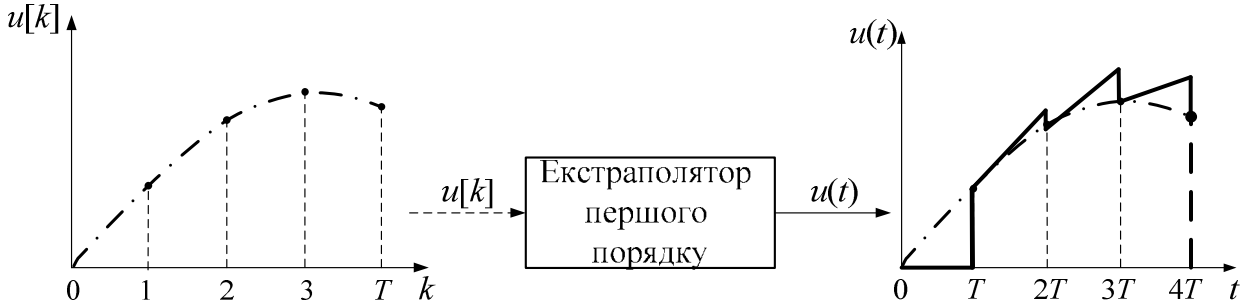


Рис. 12.4. Екстраполятор першого порядку

Безперервний сигнал на інтервалі $kT \le t < (k + 1)\ T$ відновлюється за законом лінійної екстраполяції:

$$u(t) = u(k) + \frac{t - kT}{T}(u[k] - u[k-1]). \qquad (12.3)$$

Передаточна функція екстраполятора першого порядку

$$H_1(s) = \left(\frac{1 - s^{-Ts}}{s}\right)^2 \frac{Ts + 1}{T}. \qquad (12.4)$$

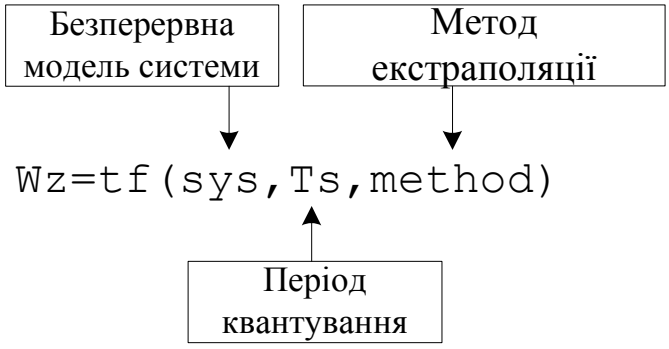
Існують й інші, менш розповсюджені на практиці, методи відновлення безперервного сигналу з дискретного.

Слід також зазначити, що дискретні системи керування не обмежуються тільки лише цифровими системами. Дискретною, наприклад, є радіолокаційна система спостереження. В ній відбитий від цілі сигнал повертається до антени лише через час, що залежить від відстані до цілі. При радіо- і телекерування застосовують дискретні способи передачі інформації.

12.2 Завдання моделей дискретних систем

Дослідження дискретних систем керування значно складніше, ніж дослідження лінійних. Однак аналіз і синтез дискретних систем істотно полегшується при використанні інтерактивних комп'ютерних засобів.

Багато функцій MATLAB, що застосовуються для створення моделей безперервних систем, можна використовувати і для формування моделей дискретних систем керування. Для цього необхідно до аргументів цих функцій додати період дискретності `Ts` у секундах. Якщо точніше, то, якщо параметр `Ts` не зазначений, він вважається рівним нулю, що відповідає безперервній системі. Наприклад, дискретну передаточну функцію в MATLAB можна одержати за допомогою функції `tf`:



Тут `'method'` - параметр, що визначає метод екстраполяції:

`zoh` – екстраполятор нульового порядку;

`foh` - екстраполятор першого порядку;

`tustin` – білінійна апроксимація Тастіна;

`prewarp` – апроксимація Тастіна з корекцією;

`matched` – метод погоджених нулів і полюсів.

Якщо метод екстраполяції не вказаний, то за замовченням вважается, що використовується екстраполятор нульового порядку. Наприклад, наступний оператор формує дискретну передаточну функцію з періодом квантування 0,1 с. та екстраполятором нульового порядку:

```
>> W=tf([1 -0.2],[1 -0.5],0.1)

Transfer function:
```

```
z - 0.2
-------
z - 0.5

Sampling time: 0.1
```

Подібним же чином можна задавати дискретні моделі в ZPG-формі та у просторі станів. Так, оператор

```
Wss=ss(A,B,C,D,Ts)
```

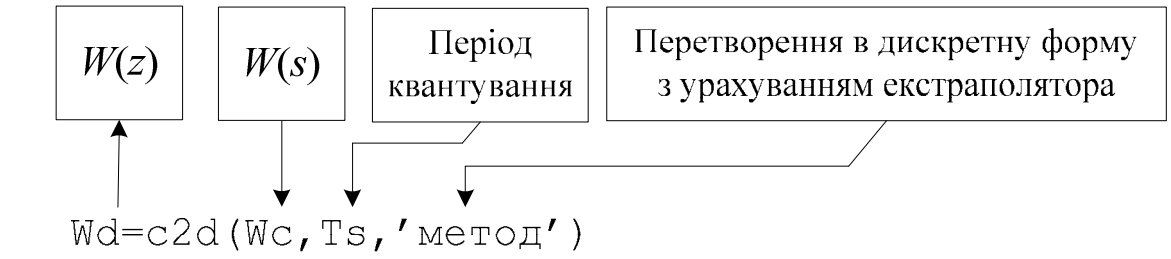
формує модель дискретної системи в просторі станів з періодом квантування T_s наступного виду:

$$\begin{cases} \mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k); \\ \mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k). \end{cases} \qquad (12.5)$$

В (12.5) вектори **x**, **u** та **y** позначають змінні станів, входи і виходи для k-1-го і k-го моментів часу. Наприклад:

```
>> A=[0.1 0.01;-0.2 0.1];B=[0;0.001];
>> C=[1 0];D=0;
>> Wss=ss(A,B,C,D,0.25)

a =
         x1    x2
   x1   0.1  0.01
   x2  -0.2   0.1

b =
          u1
   x1      0
   x2  0.001

c =
       x1  x2
   y1   1   0

d =
       u1
   y1   0

Sampling time: 0.25
Discrete-time model.
```

MATLAB дозволяє легко перетворити безперервну модель у дискретну та навпаки. Для цього використовуються функції `c2d` і `d2c`. Функція `c2d` перетворює безперервну систему в дискретну, а функція `d2c` – дискретну в безперервну. Нижче представлені формати цих функцій.



Розглянемо як приклад привод магнітної головки жорсткого диска комп'ютера. Схема, що пояснює роботу привода, представлена на рис. 12.5.

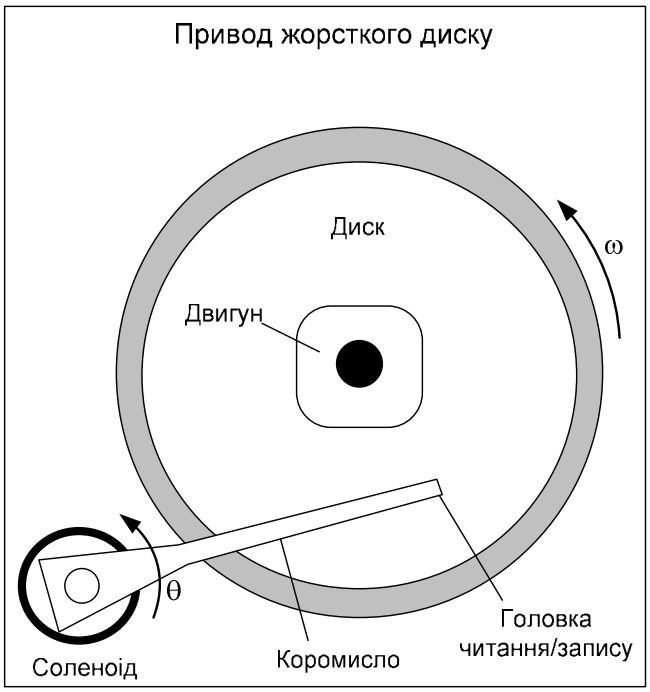


Рис. 12.5. Схема привода жорсткого диска комп’ютера

Диференціальне рівняння, що описує динаміку магнітної головки на підставі другого закону Ньютона, має такий вигляд:

$$J\frac{d^2\theta}{dt^2}+C\frac{d\theta}{dt}+K\theta=K_i i, \qquad (12,6)$$

де J – момент інерції магнітної головки; C – коефіцієнт в'язкого тертя в підшипниках; K – коефіцієнт жорсткості пружини; K_i – моментний коефіцієнт двигуна; θ - кутове положення головки; i – струм якоря двигуна.

Перетворивши (12.6) по Лапласу, одержимо передаточну функцію системи від керуючої змінної i до вихідної змінної θ:

$$W(s)=\frac{K_i}{Js^2+Cs+K}. \qquad (12.7)$$

Прийнявши J = 0,001 кгм^2, C = 0,004Нм/(рад./с), K = 10 Нм/рад., K_i = 0,05 Нм/А, побудуємо MATLAB-модель привода магнітної головки:

```
>> J=0.01;C=0.004;K=10;Ki=0.05;
>> W=tf(Ki,[J C K])

Transfer function:
         0.05
----------------------
0.01 s^2 + 0.004 s + 10
```

Тепер перейдемо від безперервної моделі до дискретної з періодом квантування T_s = 0,005 с і екстраполятором нульового порядку:

```
>> Wd=c2d(W,0.005,'zoh')

Transfer function:
6.233e-005 z + 6.229e-005
------------------------
  z^2 - 1.973 z + 0.998

Sampling time: 0.005
```

12.3 Аналіз дискретної системи

При аналізі дискретних систем керування використовуються вже відомі нам функції `step`, `impulse`, `lsim`, `bode` та ін. Зручно також користуватися графічним інтерфейсом LTI Viewer. На рис. 12.6 і 12.7 представлені перехідна функція дискретної моделі та логарифмічні частотні характеристики для безперервної і дискретної моделей привода магнітної головки жорсткого диска.

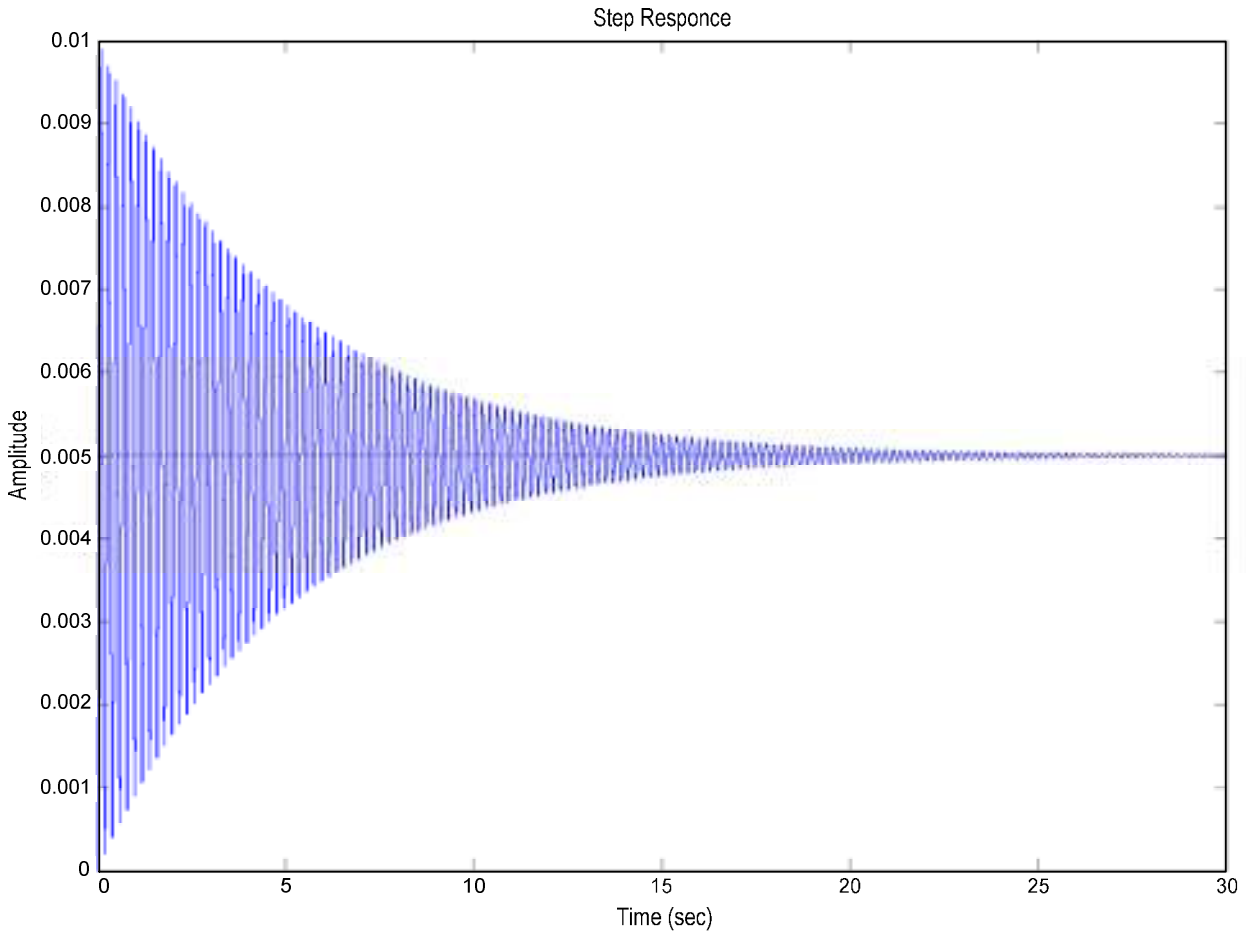


Рис. 12.6. Перехідна функція дискретної моделі

Для побудови цих графіків використовувалися наступні команди:

```
>> step(Wd)
>> figure(2)
>> bode(Wd,'-',W,':')
```

Як бачимо з рис. 12.6, система має яскраво виражену коливальність, отже, необхідна її корекція.

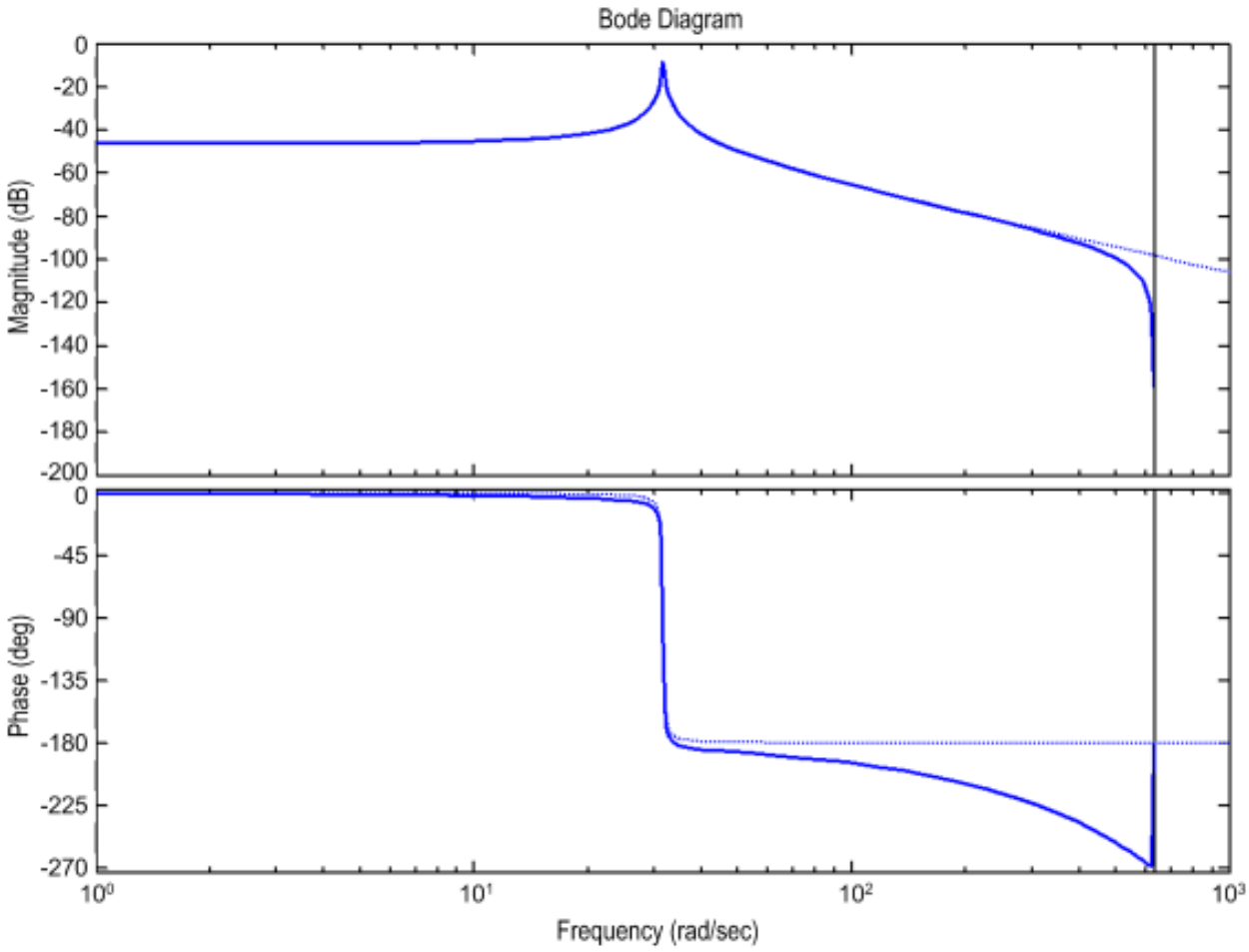


Рис. 12.7. Логарифмічні частотні характеристики для безперервної і дискретної моделей привода магнітної головки жорсткого диска

12.4 Синтез цифрового регулятора

Продовжимо розгляд приклада привода магнітної головки жорсткого диска комп'ютера (рис. 12.5). Проведений вище аналіз показав, що в системі присутні коливання, причиною яких є малий коефіцієнт демпфування. У цьому можна переконатися, побудувавши кореневий годограф дискретної моделі. Для цього можна використати функцію `rlocus` або скористатися додатком SISO Design Tool (рис. 12.8).

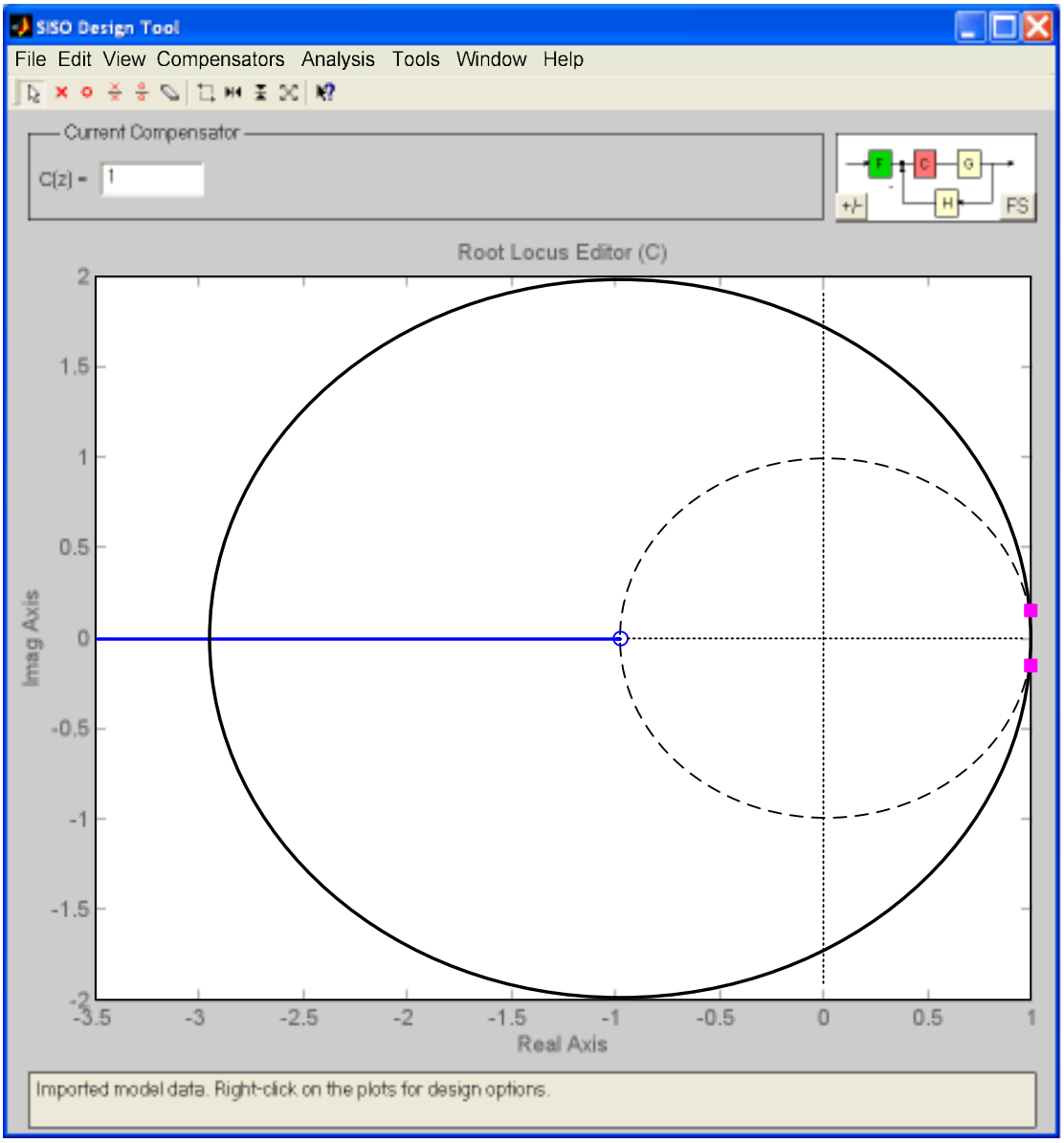


Рис. 12.8. Кореневий годограф привода магнітної головки жорсткого диска

З рис. 12.8 видно, що полюси розташовані поблизу одиничної окружності (яка для цифрових систем є областю стійкості) і мають мале демпфування. Необхідно розрахувати регулятор, що збільшить демпфування цих полюсів. Найбільш зручний спосіб - змінити коефіцієнт підсилення системи. Однак з кореневого годографу видно, що навіть невелике збільшення коефіцієнта підсилення приводить до того, що полюси замкнутої системи виходять за межі одиничної окружності отже система стає нестійкою. Тому необхідно використати більш складний регулятор, чим пропорційний.

Спробуємо застосувати ПД-регулятор. Передаточну функцію цифрового ПД-регулятора можна записати в наступному виді:

$$D(z) = K_P + \frac{K_D(z-1)}{Tz} = \frac{(K_P T + K_D)z - K_D}{Tz}. \qquad (12.8)$$

Передаточну функцію (12.8) можна представити так:

$$D(z) = K\frac{z - z_0}{z}, \qquad (12.9)$$

де

$$K = \frac{K_P T + K_D}{T}; \qquad (12.10)$$

$$z_0 = \frac{K_D}{K_P T + K_D}. \qquad (12.11)$$

Нехай z_0 = 0,85. Для визначення K можна скористатися методом кореневого годографа. Для початку приймемо K = 1.

```
>> D2=tf([1 -0.85],[1 0],0.005)

Transfer function:
z - 0.85
--------
   z

Sampling time: 0.005)
```

Побудуємо кореневий годограф системи з регулятором в SISO Design Tool (рис. 12.9).

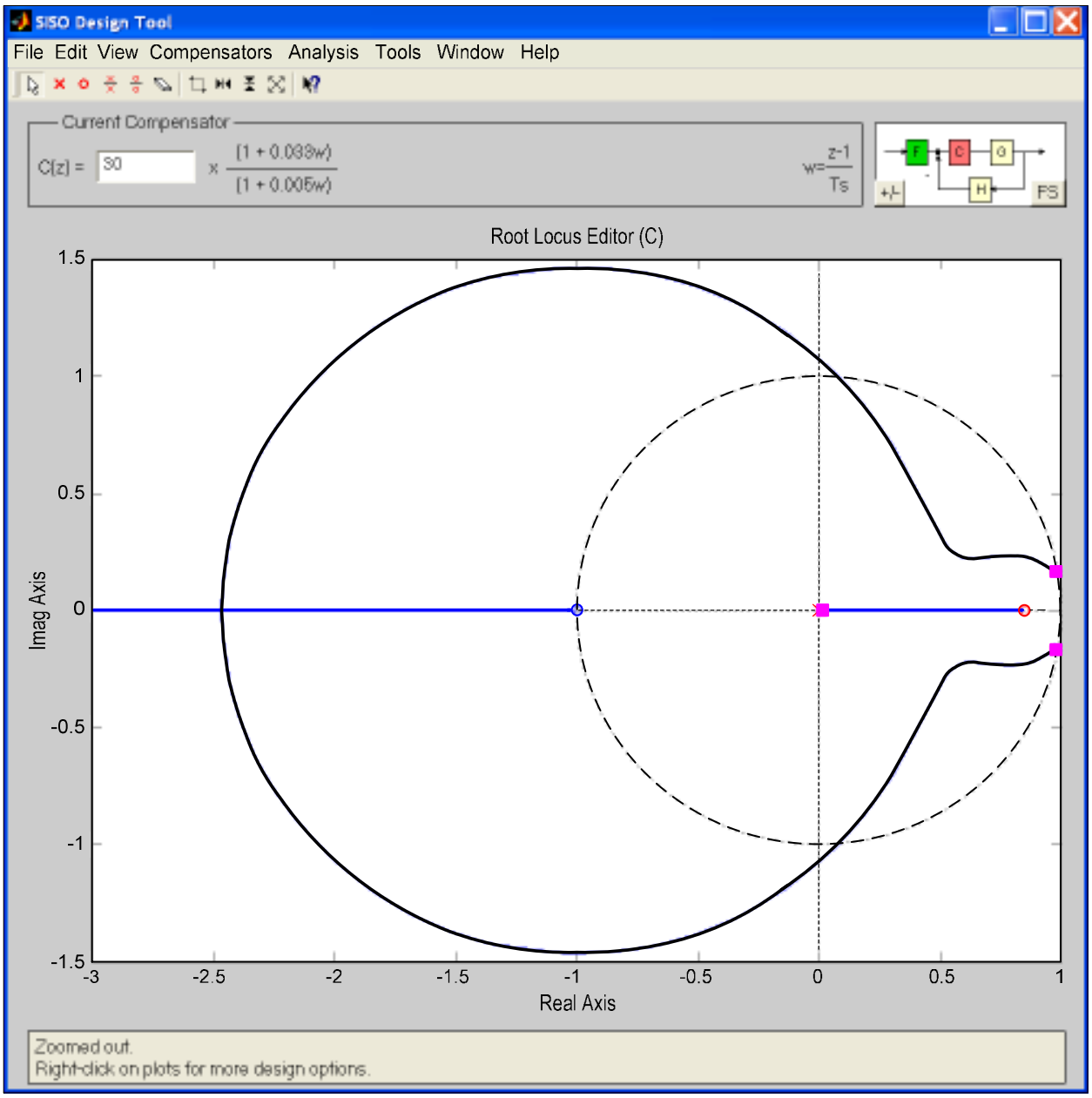
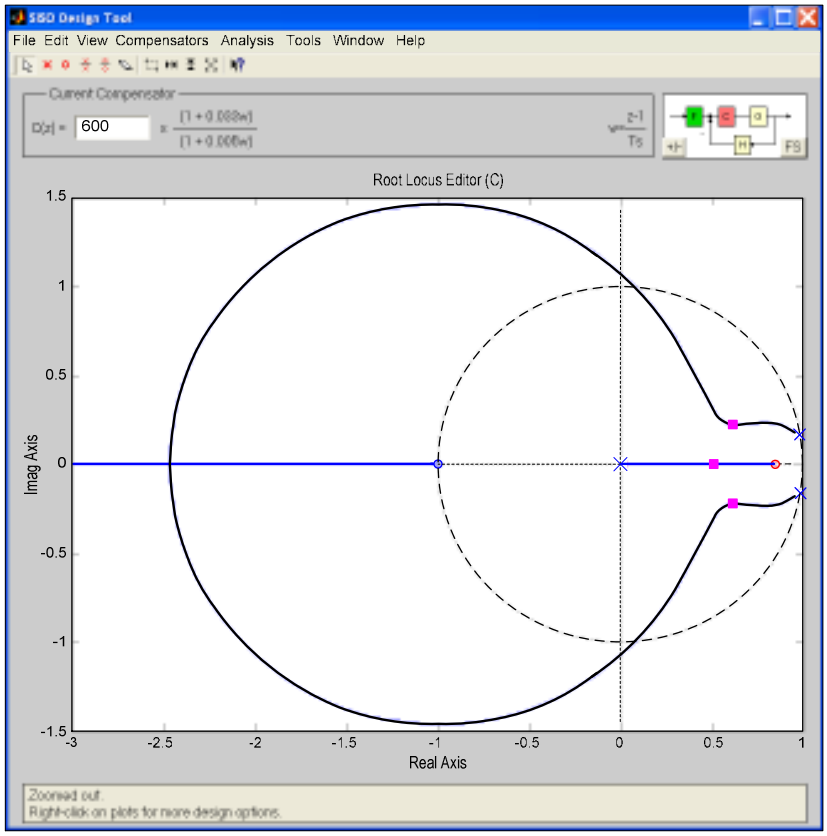


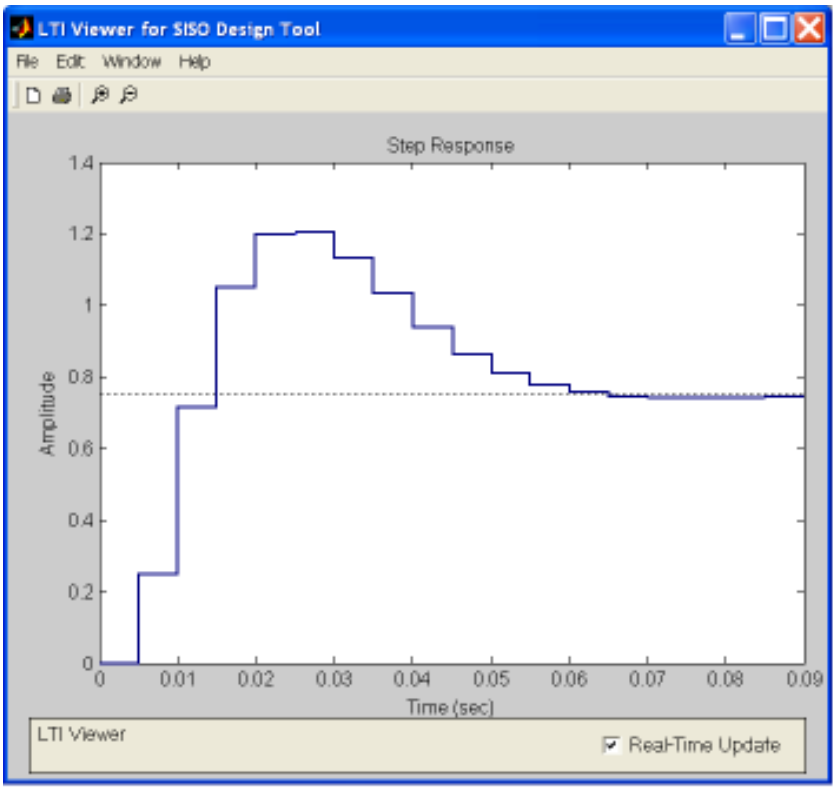
Рис. 12.9. Визначення границі стійкості
за допомогою SISO Design Tool

Переміщуючи на кореневому годографі полюси усередину одиничної окружності, можна підібрати значення K, що відповідає бажаній якості системи керування. Процес підбора коефіцієнта K показаний на рис. 12.10.

Якщо бажаної якості досягти не вдається, то необхідно використати інший регулятор, додаючи полюси та нулі за допомогою інструментів, вбудованих в SISO Design Tool.



а



б

Рис. 12.10. Настроювання цифрового регулятора в SISO Design Tool: *а* – розміщення полюсів при K = 600; *б* – відповідна перехідна функція

Завдання для самостійної роботи

1. Перетворіть наступні безперервні передаточні функції у дискретні. Вважатиме, що у дискретній системі період квантування $T = 0{,}25$ с та використовується екстраполятор нульового порядку.

$$а)\ W(s) = \frac{3}{0{,}2s+1};\ б)\ W(s) = \frac{4s+2}{s^2+2s+1};\ в)\ W(s) = \frac{1}{s};$$

$$г)\ W(s) = \frac{1}{s(s+1)};\ д)\ W(s) = \frac{1}{s(4s^2+5s+1)};\ е)\ W(s) = \frac{0{,}5s+3}{s^2+0{,}3s+0{,}5}.$$

2. Передаточна функція замкненої дискретної системи має вигляд

$$W(z) = \frac{0{,}05z+0{,}05}{1-1{,}698z+0{,}7408}.$$

а) Визначить еквівалентну $W(z)$ безперервну передаточну функцію.

б) Побудуйте перехідну функцію дискретної системи.

в) У тому ж графічному вікні побудуйте перехідну функцію безперервної системи та порівняйте одержані графіки.

3. Дискретна система у просторі станів моделюється наступними рівняннями

$$\begin{cases} \mathbf{x}(k+1) = \begin{bmatrix} 0 & 1 \\ -0{,}1 & 0{,}8 \end{bmatrix}\mathbf{x}(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix}u(k); \\ \mathbf{y}(k+1) = \begin{bmatrix} 0 & 1 \end{bmatrix}\mathbf{x}(k). \end{cases}$$

а) За допомогою команди `ss` сформуйте модель системи у MATLAB. Період квантування $T = 0{,}05$ с.

б) Отримайте еквівалентну передаточну функцію.

в) Побудуйте кореневий годограф системи та визначить діапазон значень коефіцієнту підсилення K, при якому система буде стійкою.

ЧАСТИНА ІІІ
РОБОТА В SIMULINK

13. ОСНОВИ SIMULINK

13.1 Призначення пакета Simulink

До складу системи MATLAB входить потужний засіб дослідження і моделювання систем автоматичного керування - пакет Simulink. При цьому модель системи керування в Simulink представляється у вигляді звичної структурної схеми.

Для побудови структурних схем об'єктів, що моделюються, Simulink має велику бібліотеку блокових компонентів, що включає джерела сигналів, лінійні та нелінійні ланки з різними передаточними функціями, реєструючі пристрої та ін., а також засоби анімації і звукового супроводу. Важливою властивістю Simulink є його здатність моделювати системи у реальному масштабі часу. Завдяки цим достоїнствам Simulink широко використовується для моделювання складних динамічних систем у всіх галузях науки і техніки.

13.2 Початок роботи з Simulink

Запустити Simulink можна двома способами (рис. 13.1):

- натиснути на кнопку Simulink  на панелі інструментів MATLAB;
- ввести слово `simulink` у командному рядку.

Після цього на екрані з'являється вікно браузера бібліотек (**Simulink Library Browser**), що показане на рис. 13.2. Вікно має заголовок, меню, панель інструментів, поле інформаційних повідомлень, вікна з деревом бібліотек і компонентами виділеного розділу бібліотек і рядок стану. Щоб розкрити будь-яку з віток дерева бібліотек, потрібно натиснути лівою кнопкою миші на значок «+» напроти відповідної піктограми в лівій частині вікна браузера бібліотек або двічі клацнути по цій піктограмі. У цьому випадку в правій частині вікна з'явиться список компонентів зазначеної бібліотеки.

У принципі вікно має стандартний інтерфейс «під Windows» і на початковому етапі робота в Simulink не повинна викликати істотних утруднень.

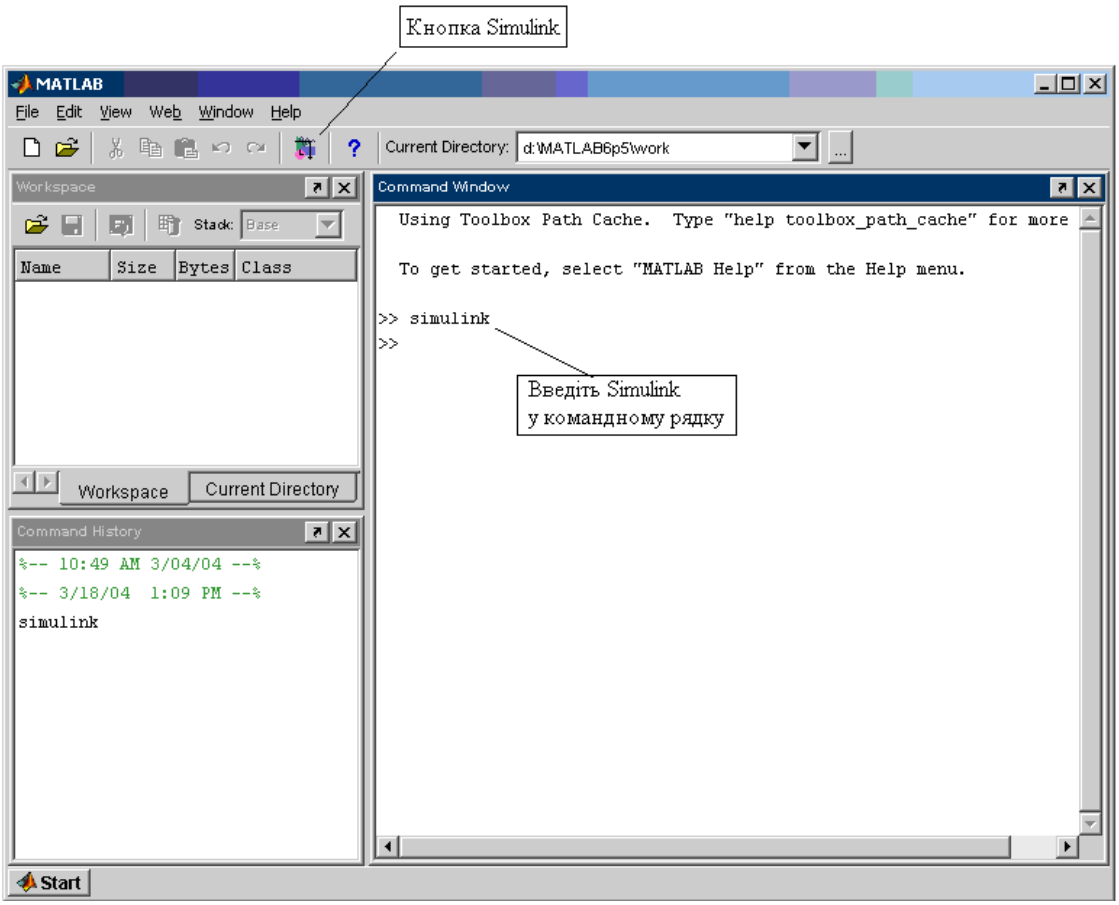


Рис. 13.1. Початок роботи в Simulink

Стандартна бібліотека блоків Simulink складається з наступних компонентів (рис. 13.2):

- елементи безперервних систем (`Continuous`);
- елементи дискретних систем (`Discrete`);
- неліній
- ності (`Discontinuities`);
- математичні операції (`Math Operations`);
- джерела сигналів (`Sources`);
- засоби реєстрації (`Sinks`) і т.ін.

У залежності від версії Simulink склад бібліотеки дещо відрізняється.

13.3 Створення найпростішої моделі

Тепер спробуємо створити свою модель. Для цього потрібно

спочатку відкрити вікно нової системи за допомогою кнопки на панелі інструментів або вибрати **File → New → Model**.

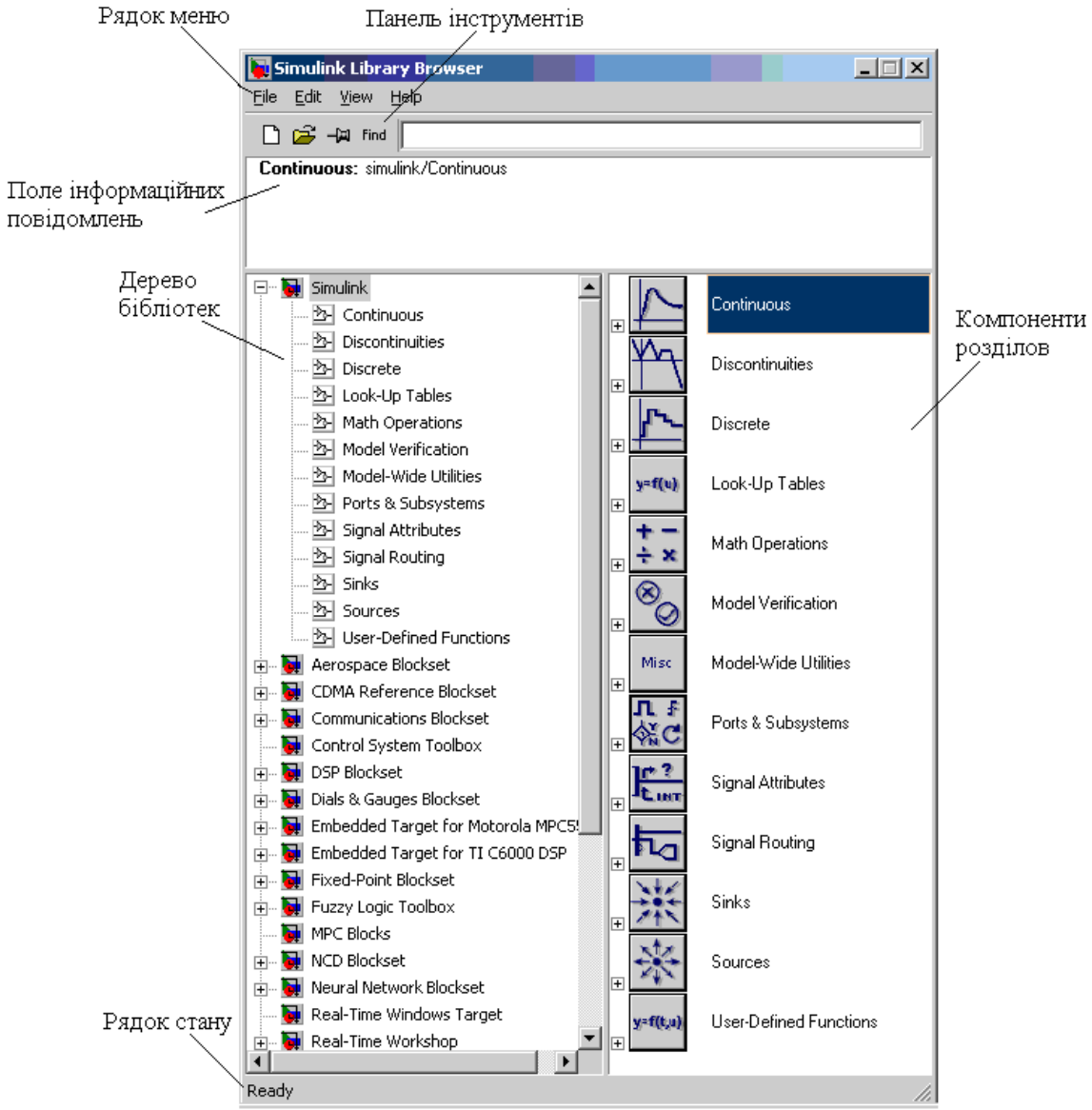


Рис. 13.2. Вікно браузера бібліотек Simulink

Вікну, що відкривається (рис. 13.3), за замовчуванням привласнюється ім'я Untitled. Щоб зберегти модель, натисніть кнопку **(File → Save)**. У вікні, що відкрилося, можна задати своє ім'я файлу, наприклад Model_1 (модель 1). Зверніть увагу, що Simulink (як і взагалі MATLAB) «не любить» імена, що починаються із цифр або містять кирилицю. Крім того, щоб уникнути непорозумінь, настійно рекомендуємо зберігати моделі

відразу ж, на самому початку її створення. Всі подальші зміни можна зберегти повторним натисканням кнопки . При збереженні автоматично створюється файл у форматі *.mdl, що містить всю інформацію, необхідну для відкриття моделі в наступних сеансах роботи Simulink. Також як й в інших додатках MATLAB, за замовчуванням файл зберігається в робочу папку `MATLAB/work`. При необхідності робочу папку можна змінити.

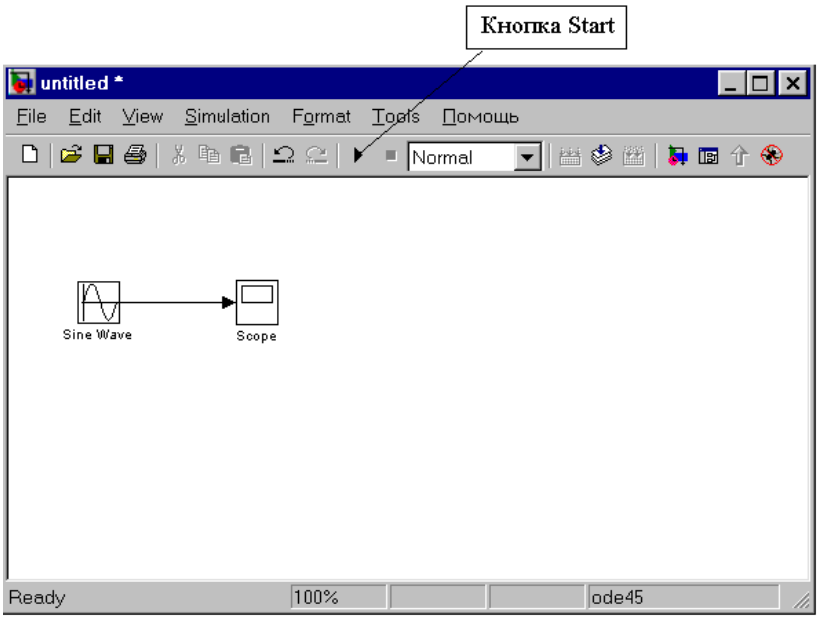


Рис. 13.3. Початок моделювання

Тепер приступимо безпосередньо до створення моделі. Виберемо з бібліотеки джерел сигналів (`Sources`) блок, що генерує синусоїдальний сигнал (`Sine Wave`). Для цього спочатку ввійдемо у бібліотеку `Sources`, після цього виберемо блок `Sine Wave` (рис. 13.4) і, утримуючи ліву кнопку миші, перетягнемо його у вікно нової моделі. Копія блоку буде встановлена в тім місці вікна нової моделі, де ми залишили його зображення. Далі виберемо у вікні бібліотеки `Sinks` осцилограф (`Scope`) і в такий же спосіб додамо його в нашу модель. Щоб з'єднати два блоки, потрібно помістити курсор на вихідний порт у правій частині блока `Sine Wave`, що позначається символом «>» (рис. 13.5).

При цьому курсор прийме форму хрестика. Утримуючи натиснутою ліву кнопку миші, перемістимо курсор до вхідного порту блоку `Scope`, що позначений символом «>» на лівій стороні

блока.

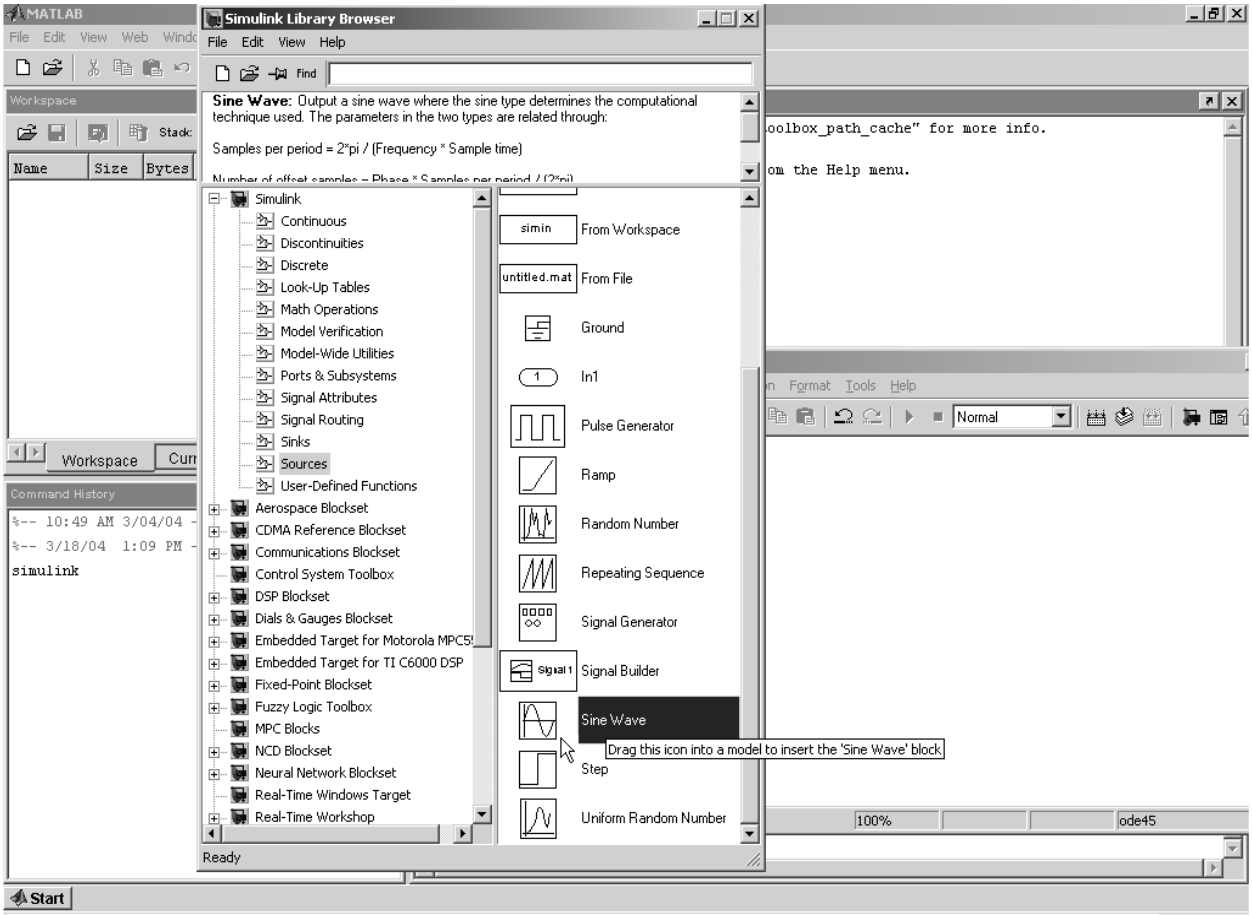


Рис. 13.4 Вибір компонента бібліотеки Simulink

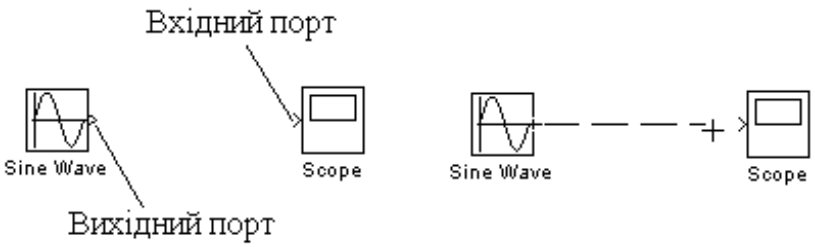


Рис. 13.5. Початок з'єднання блока джерела із блоком осцилографа

Другий спосіб - виділити блок-джерело лівою кнопкою миші і, утримуючи клавішу Ctrl, клацнути мишею на блоці-приймачі. Коли з'єднання буде встановлено, на сполучній лінії з'явиться стрілка, що вказує напрямок передачі інформації.

Для початку моделювання потрібно вибрати опцію **Simulation → Start** з меню або просто натиснути кнопку старт на панелі інструментів (рис. 13.3). Щоб спостерігати синусоїдальний сигнал, що генерується блоком `Sine Wave`,

відкрийте індикатор подвійним щигликом миші, при цьому на екрані з'явиться наведене на рис. 13.6[1] зображення.

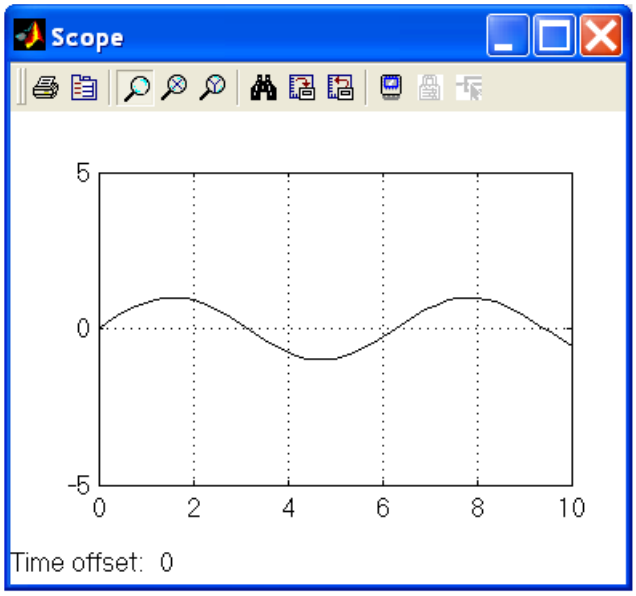


Рис. 13.6 Результат моделювання

Вікно осцилографа має свою панель інструментів (рис. 13.7). Якщо підвести покажчики миші до якої-небудь кнопки, спливає підказка про її призначенні. Деякі параметри блока `Scope` ми будемо розглядати нижче.

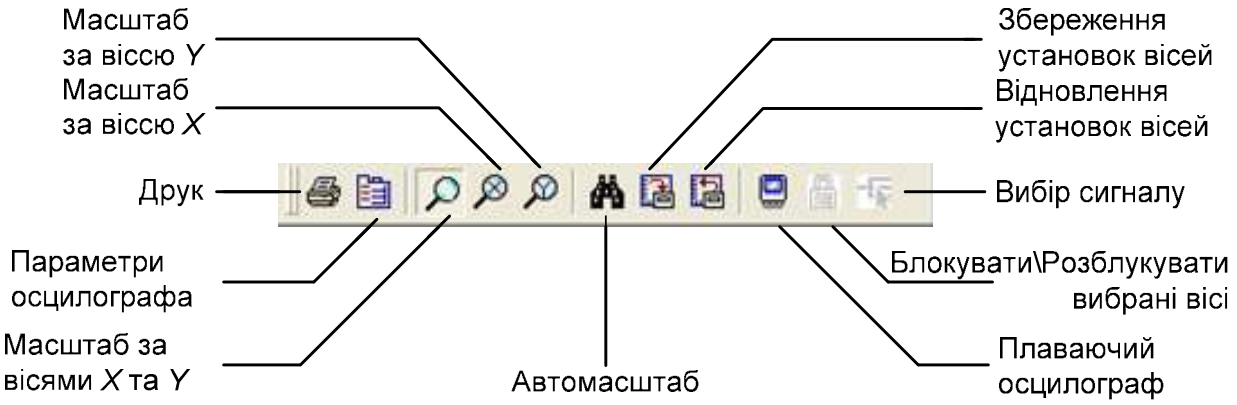


Рис. 13.7. Панель інструментів блока `Scope`

[1] Тут і далі для більшої зручності при роботі з книгою чорний колір фону блоків Scope замінений на білий, окрім рис. 20.7.

Спробуємо змінити масштаб зображення. Для цього досить клацнути правою кнопкою миші у вікні осцилограми. У контекстному меню, що з'явиться, потрібно вибрати команду `Axes Properties...`. У вікні властивостей вісей, що відкриється, треба замінити значення Y-min=-5 і Y-max = 5, наприклад, на Y-min=-2 і Y-max = 2. У цьому ж вікні можна задати ім'я отриманого сигналу. Для цього його потрібно ввести в текстовому полі заголовків **Title** (рис. 13.8).

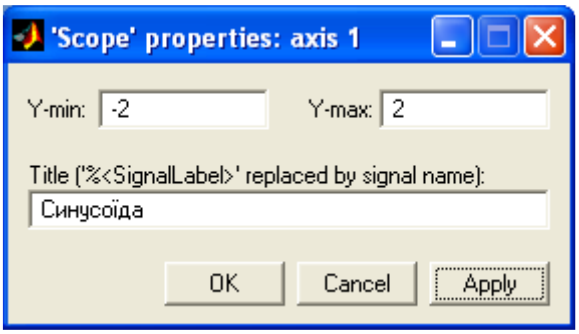


Рис. 13.8 Вікно властивостей `Axes Properties`

Тепер, натиснувши кнопку **Apply**, можна побачити осцилограму зі зміненим масштабом і назвою сигналу (рис. 13.9).

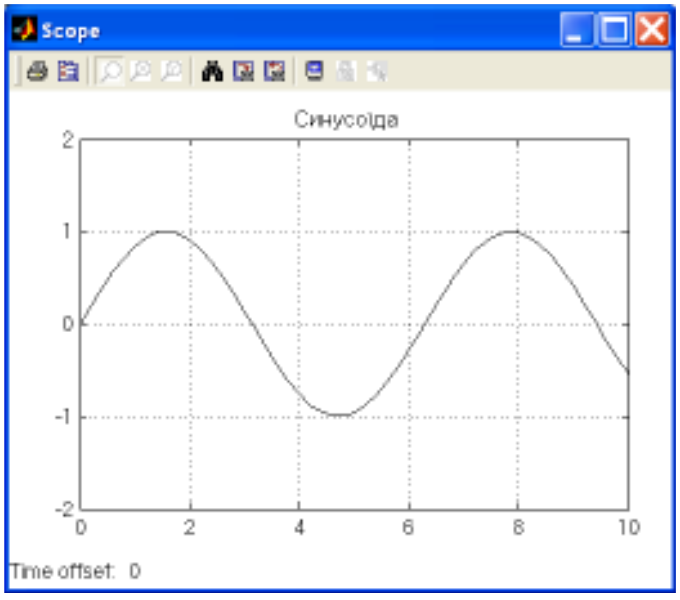


Рис. 13.9. Осцилограма в новому масштабі

Тепер осцилограма стала більш наочною. Якщо її вид вас влаштовує, то досить зафіксувати зроблені зміни масштабу, натиснувши кнопку **ОК** у вікні властивостей осей. Можна і відмовитися від зроблених змін, натиснувши кнопку **Cancel**.

У контекстному меню осцилограми є ще команда - **Autoscale** (Автомасштабування). Ця ж команда реалізується кнопкою **Autoscale** на панелі інструментів вікна осцилограми. Вона встановлює масштаб, при якому вікно осцилограми використовується повністю. У цьому випадку це означає, що синусоїда буде мати максимально можливий розмір, як показано на рис. 13.10.

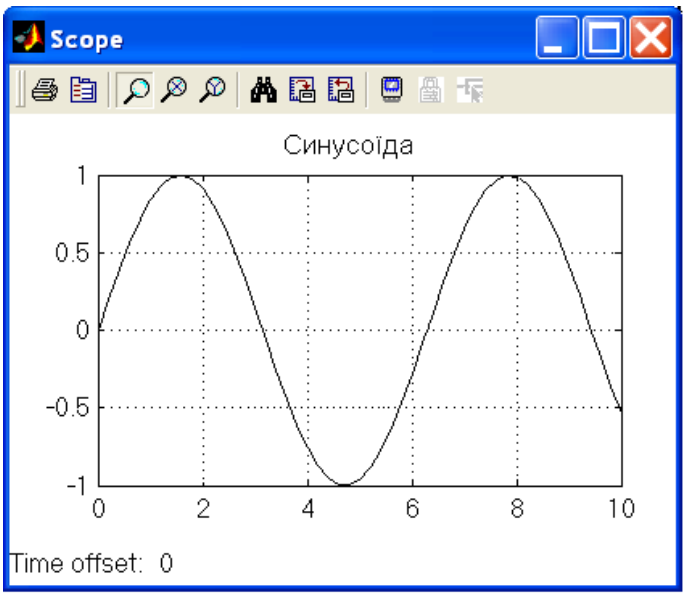


Рис. 13.10. Осцилограма після виконання команди **Autoscale**

Для більш наочного подання даних можна використати і стандартні для Windows-додатків прийоми керування вікнами. Так, наприклад, можна змінювати розміри і пропорції вікна осцилограми. Для цього покажчик миші треба підвести до краю вікна, при цьому повинна з’явитися двонаправлена стрілка. Тепер потрібно натиснути ліву кнопку миші та, не відпускаючи її, розтягувати вікно в бажаному напрямку. Щоб розгорнути вікно на весь екран, найпростіше скористатися середньою кнопкою в правому верхньому куті графіка, а переміщати вікно можна, утримуючи його за допомогою лівої кнопки миші за полі заголовків (верхня частина вікна). Втім, ці ж операції можна виконати,

клацнувши по піктограмі  в лівому верхньому куті й вибравши відповідну позицію в меню, що з'явиться.

13.4 Зміна параметрів моделювання

В Simulink можна задавати свої параметри синусоїдального сигналу, наприклад амплітуду (**Amplitude**), частоту (**Frequency**) або фазовий зсув (**Phase**). Для цього досить двічі клацнути лівою кнопкою миші на блоці `Sine Wave` у вікні моделі (рис. 13.11).

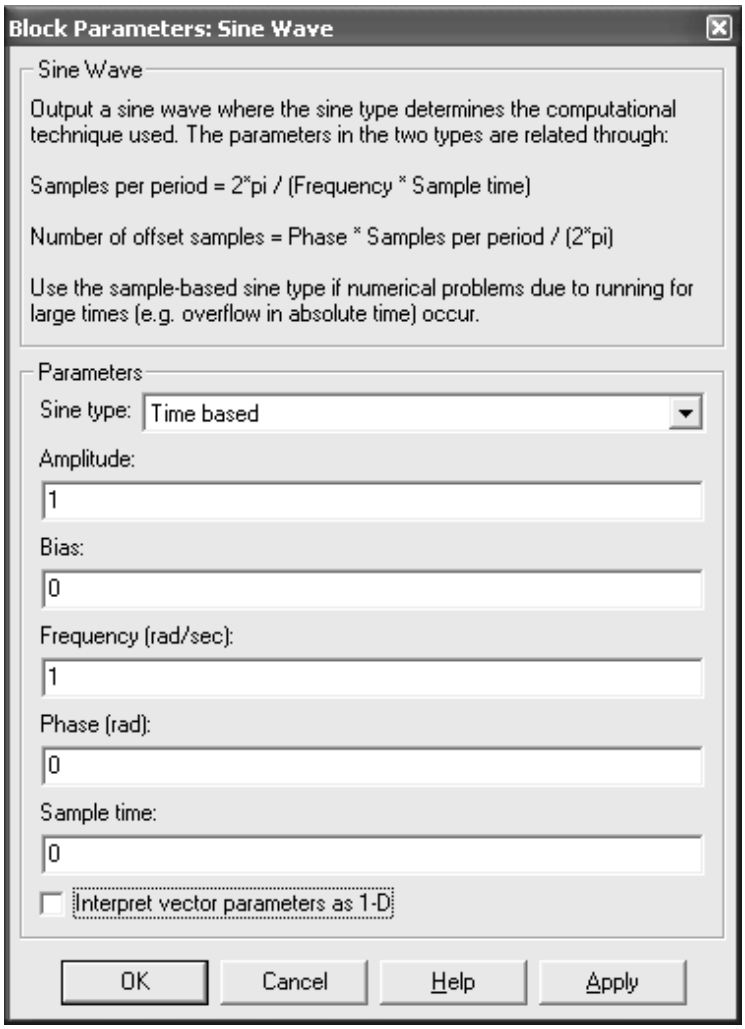


Рис. 13.11 Діалогове вікно параметрів блока `Sine Wave`

Таким же чином можна викликати вікна параметрів і для інших блоків Simulink.

Якщо вибрати з меню **Simulation** опцію **Parameters**, то можна спостерігати і редагувати параметри моделювання (рис. 13.12)**.**

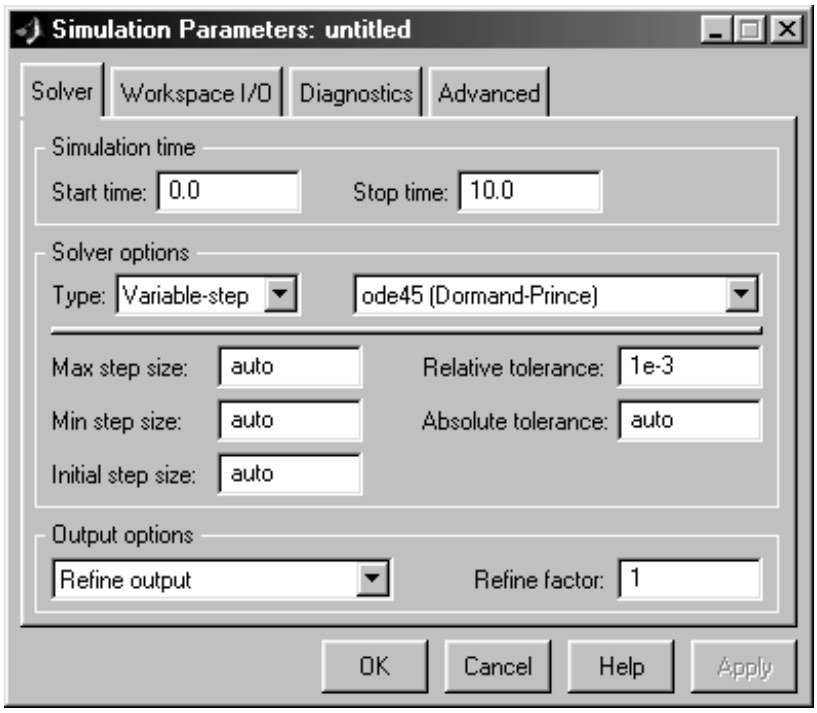


Рис. 13.12 Вікно параметрів моделювання

У вікні, що з'являється, за замовчуванням відкрита вкладка **Solver**, на якій відображені основні параметри моделювання. На вкладці **Workspace I/O** задаються параметри завантаження змінних з робочої області та розміщення даних у робочій області MATLAB. Вкладка **Diagnostics** використовується для завдання значень опцій діагностики, а вкладка **Advanced** дозволяє встановлювати значення опцій для оптимізації процесу інтегрування.

Розглянемо більш докладно вкладку **Solver** (Вирішувач). Вона включає три групи опцій. Перша група – **Simulation time** (Час моделювання). Час моделювання задається початковим часом **Start time** (за замовчуванням 0) і кінцевим часом **Stop time**. Змініть час закінчення моделювання, наприклад, на 1000 с (замість 10 с, прийнятих за замовчуванням).

Друга група опцій **Solver options** (Опції вирішувача) включає список **Type** (Тип), що розкривається**,** для завдання параметрів керування шагом інтегрування і список методу вирішення диференціальних рівнянь.

Наступні параметри цієї групи звичайно задаються автоматично, хоча при необхідності їх теж можна задавати вручну:

- **Max step size** – максимальний шаг інтегрування системи однорідних диференціальних рівнянь;

- Min step size – мінімальний шаг інтегрування;
- Initial step size – початковий шаг інтегрування;
- Relative tolerance – відносна погрішність інтегрування;
- Absolute tolerance – абсолютна погрішність інтегрування.

У групі опцій **Output options** (Опції виводу) вибирається спосіб формування вихідного масиву даних.

Залишіть ці параметри без зміни і натисніть **OK**.

Ми створили просту модель. Збережіть всі внесені зміни та закрийте вікно моделі. Щоб відкрити її для подальшої роботи можна натиснути  або вибрати **File → Open**. При цьому відкривається робоча папка.

Завдання для самостійної роботи

1. У бібліотеці джерел сигналів `Sources` є блок `Signal Generator`, що за вибором користувача може генерувати синусоїдальний, прямокутний, пилоподібний або випадковий сигнал.

а) Побудуйте модель, яка б дозволяла спостерігати у вікні блоку `Scope` за пилоподібним сигналом, що генерується блоком `Signal Generator`.

б) Встановіть масштаб, при якому вікно графіку буде використовуватися повністю.

в) У меню Simulink **Simulation Parameters** у полі **Max step size:** змініть `auto` на 0.01. Повторіть моделювання та порівняйте результати з результатами п. *а*).

г) У вікні параметрів блоку `Signal Generator` змініть значення амплітуди сигналу на -1. Повторіть моделювання та порівняйте одержані результати з попередніми. Зробить висновки.

2. Побудуйте модель, яка б дозволяла спостерігати у вікні блоку `Scope` за прямокутним сигналом, що генерується блоком `Signal Generator`.

а) Повторить дії п.п. *б*) – *г*) попереднього завдання.

б) Задайте одержаному графіку ім’я „Прямокутний сигнал”.

14. РОБОТА З МОДЕЛЯМИ SIMULINK

У попередній главі ми навчилися створювати найпростішу модель. Зараз ми розглянемо, як редагувати модель, вводити коментарі та інші текстові написи, застосовувати різні стилі оформлення, а також запускати кілька моделей одночасно.

14.1 Відкриття моделі

Відкриємо модель, створену у попередньої главі. Для цього вибираємо піктограму  **Open a model** на панелі інструментів вікна **Simulink Library Browser**. Можна також вибрати **File → Open** або застосувати комбінацію клавіш Ctrl+N. Модель також можна викликати з вікна MATLAB або вікна моделі Simulink.

14.2 Операції із блоками моделі

Перш ніж створювати більш складні моделі, навчимося робити найпростіші операції із блоками моделі: копіювати, переміщати, видаляти, обертати, змінювати розміри та ін.

Складні моделі можуть мати у своїй структурі кілька однакових блоків. Для створення таких моделей зручно використовувати операцію копіювання блоків. Копіювати можна як окремі блоки, так і всю модель цілком.

14.2.1 Копіювання одного блока

Для копіювання одного блока варто помістити покажчик миші на його зображення, натиснути праву кнопку і, не відпускаючи її, перемістити виділений блок у необхідне місце (рис. 14.1). Для зняття виділення треба клацнути лівою кнопкою миші поза зображенням блока.

Другим способом копіювання є використання буфера обміну. Щоб скопіювати виділений блок у буфер необхідно застосувати команду **Edit → Copy**, після чого копія виділеного блока надходить у буфер обміну і зберігається в ньому (до аналогічного ефекту

приводить одночасне натискання клавіш Ctrl+C або піктограми на панелі інструментів).

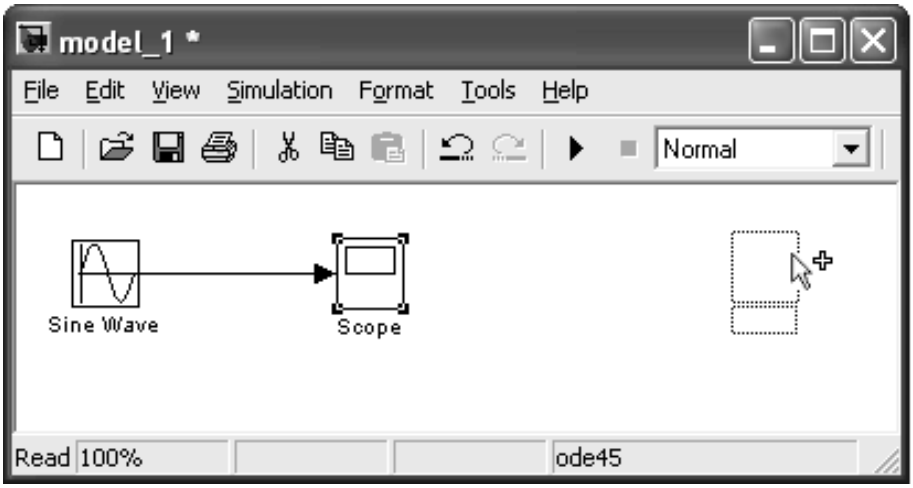


Рис. 14.1. Копіювання блока

Тепер для вставки копії блока досить помістити в потрібне місце курсор миші та виконати команду **Edit → Paste**, використати сполучення клавіш Ctrl+V або натиснути піктограму . Зверніть увагу, що кожен новий блок автоматично одержує нове найменування. Так, якщо вихідний блок називався `Scope`, то наступний стає `Scope1`, потім `Scope2` і т.д. (рис. 14.2).

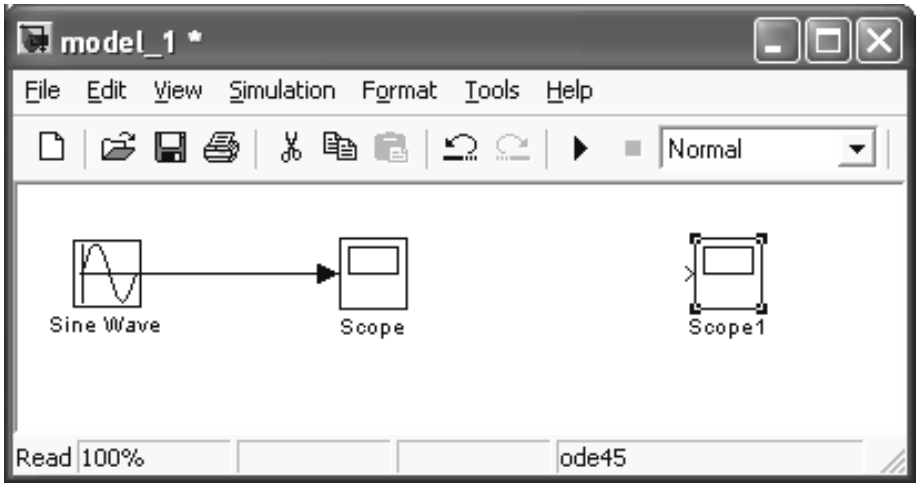


Рис. 14.2. Вставка блока

14.2.2 Переміщення і видалення блока

Щоб перемістити блок в інше місце вікна моделі або навіть в інше вікно Simulink, потрібно виділити необхідний блок лівою кнопкою миші та, не відпускаючи її, перенести блок у потрібне місце. Другим способом є використання команд **Cut** і **Paste** меню **Edit**. При виконанні команди **Cut** (можна використати комбінацію клавіш Ctrl+X або піктограму  на панелі інструментів) виділений блок зникає з вікна моделі та розміщується в буфері обміну.

Видалити виділений блок не поміщаючи його в буфер обміну можна натиснувши клавішу Delete, або командою **Edit →Clear**.

14.2.3 Виділення декількох блоків

Щоб застосувати розглянуті вище операції відразу до декількох блоків моделі або до всієї моделі, необхідні блоки потрібно виділити всі одночасно. Це можна зробити двома способами.

Перший спосіб – натиснути клавішу Shift і, не відпускаючи її, вибрати мишею кожен об’єкт групи. При другому способі потрібно встановити курсор миші поруч із блоками, яки треба виділити, і натиснути її ліву кнопку. Тепер при переміщенні миші з'явиться прямокутна рамка, що розширюється, з тонких пунктирних ліній (рис. 14.3). Як тільки ця рамка захопить який-небудь блок, він буде виділений. Охопивши рамкою кілька блоків, можна виділити їх усі. Для виділення всієї моделі можна також скористатися командою **Edit → Select all** або комбінацією клавіш Ctrl+A.

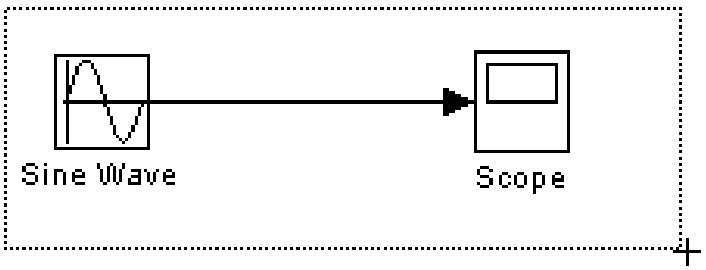


Рис. 14.3. Виділення декількох блоків

14.2.4 Зміна розмірів блоків і масштабу моделі

Раніше ми за допомогою миші змінювали розмір вікна графіка. Однак Simulink дозволяє змінювати також і розміри блоків. Це зручно, якщо вміст блока не вміщується в границях піктограми стандартного розміру. Щоб змінити розмір блока потрібно його виділити, після чого встановити курсор миші на маркери по його кутах. Як тільки курсор миші перетвориться у двонаправлену діагональну стрілку, можна збільшувати або зменшувати розмір блока, утримуючи ліву кнопку миші (рис. 14.4). Зверніть увагу на те, що розтягується тільки піктограма блока, а розміри його назви у вигляді текстового напису не змінюються.

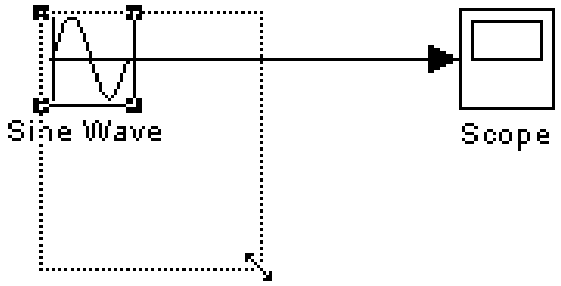


Рис. 14.4. Зміна розміру блока

Часто виникає необхідність у зміні масштабу всієї моделі. Наприклад, модель занадто велика і потрібно її всю вмістити на екран. Або навпаки, необхідно приділити увагу лише невеликому фрагменту моделі. Для цього використовуються команди зміни масштабу моделі, які розташовані в меню **View**:

- **Zoom in** (або клавіша **R**) – збільшення масштабу;
- **Zoom out** (або клавіша **V**) – зменшення масштабу;
- **Fit system to view** (або клавіша **F**) – підігнати розмір системи під розміри вікна;
- **Normal(100%)** – повернення до стандартного розміру.

При цьому масштаб зображення моделі вказується на панелі стану внизу вікна моделі (рис. 14.5).

Як і у більшості інших програм для скасування останньої операції можна скористатися командою **Edit → Undo** (як альтернатива – натискання піктограми на панелі інструментів або сполучення клавіш Ctrl+Z). Для відновлення скасованої операції

служить команда **Redo** (можна також використати піктограму або сполучення клавіш Ctrl+Y). Однак знову таки як і у більшості інших програм, при зміні масштабу ці операції не підтримуються. Крім того, «гарячі» клавіші R, F і V для зміни масштабу можуть не працювати, якщо не включена англійська розкладка клавіатури.

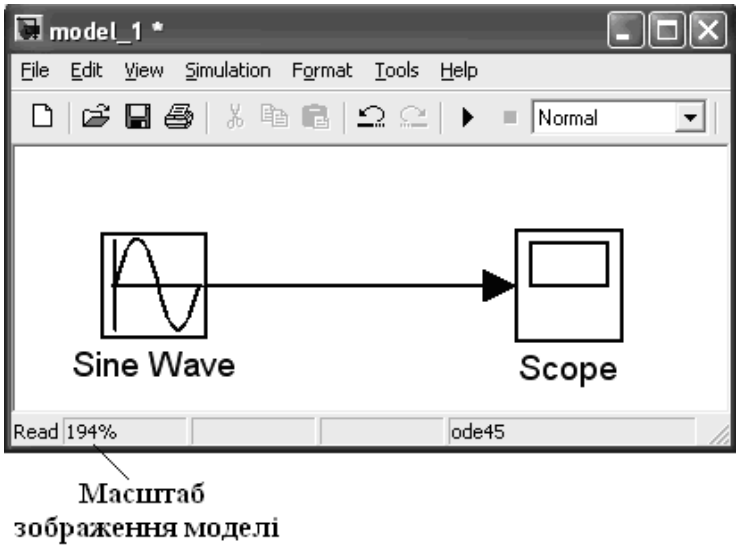


Рис. 14.5. Зміна масштабу зображення

14.2.5 Вставка блоків у з'єднання

При переміщенні блока, з'єднаного з іншими блоками моделі, лінія, що сполучує блоки, не переривається. Цю властивість можна використати для вставки нового блока в вже створену модель. Так, рис. 14.6 показує вставку блока `Integrator` (Інтегратор) з бібліотеки блоків `Continuous` між джерелом синусоїдального сигналу і осцилографом. Однак подібна проста вставка можлива для блоків, що мають тільки один вхід та один вихід.

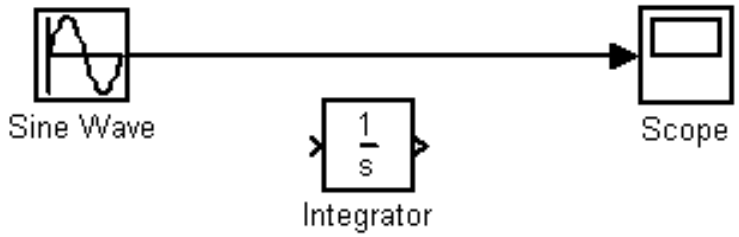


Рис. 14.6. Вставка блока

14.2.6 Форматування блоків за допомогою меню Format

Більшість операцій з форматування блоків доступна в меню **Format** у рядку меню або в контекстному меню. Нагадаємо, що контекстне меню викликається щигликом правої кнопки миші, і набір його команд залежить від елемента, на якому клацнули мишею, тобто від контексту. Отже, щоб викликати контекстне меню для роботи із блоком, потрібно клацнути правою кнопкою миші по цьому блоці (рис. 14.7).

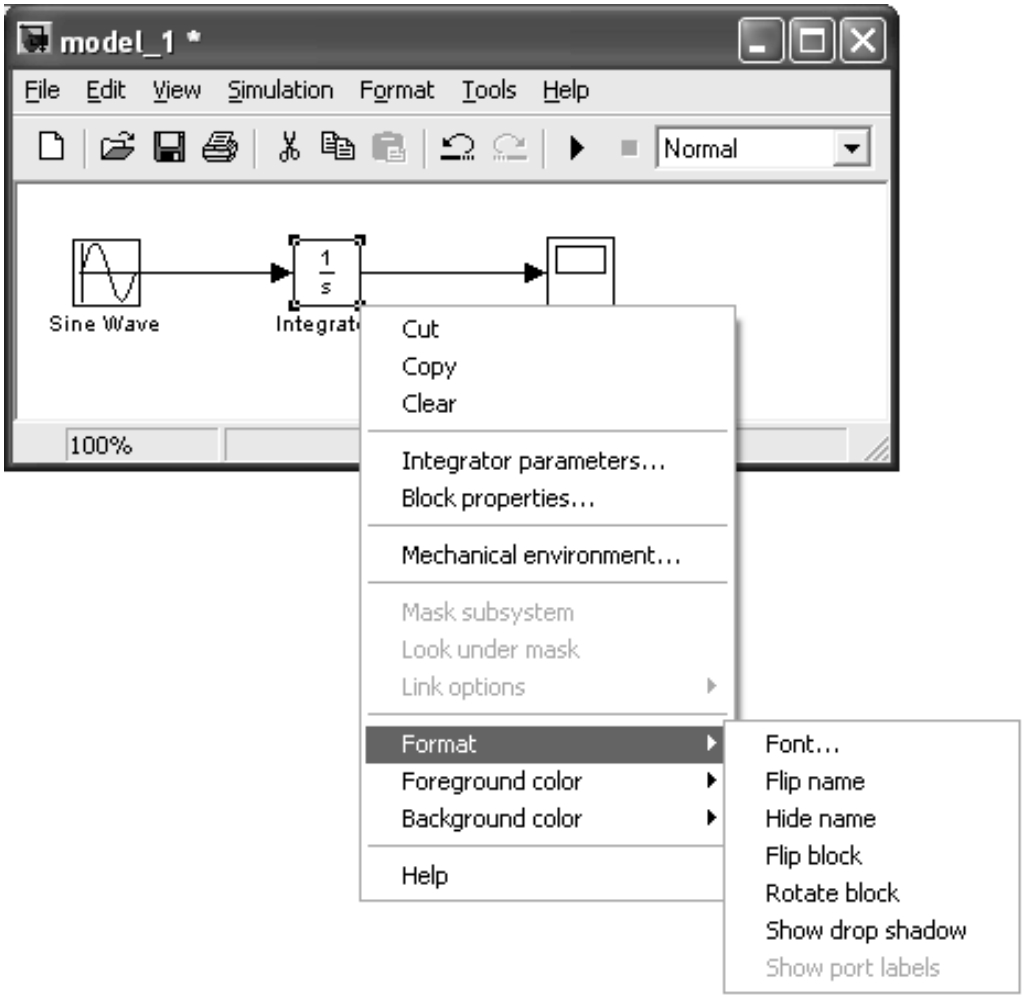


Рис. 14.7. Контекстне меню для блока `Integrator`

Нижче наведені деякі команди меню **Format** (і контекстного меню) для форматування блоків.

Font - установка шрифту для текстових написів до блока. При

виборі цієї команди з'являється вікно (рис. 14.8), у якому можна вибрати тип шрифту напису, розмір і стиль накреслення символів.

Flip name - приміщення імені блока зверху або знизу блока. Для переміщення імені блока можна також натиснути на ньому лівою кнопкою миші та, не відпускаючи її, перемістити в зазначене місце.

Show/Hide name - відображення або приховання імені виділеного блока.

Flip block – відзеркалення блока відносно вертикальної вісі. При цьому вхідні та вихідні порти блока міняють своє місце розташування.

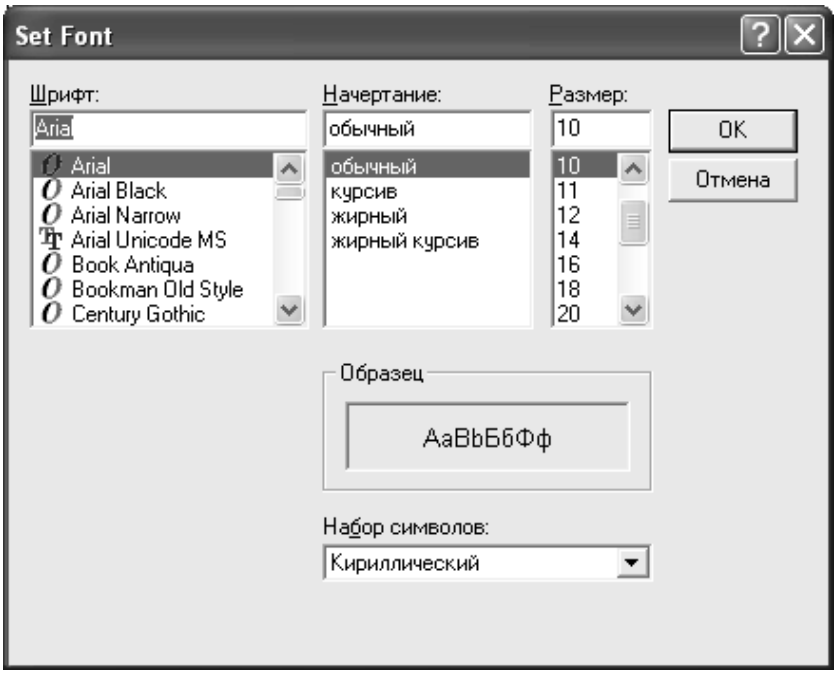


Рис. 14.8. Вікно установки шрифту

Rotate block – поворот блока на 90° за годинною стрілкою. При цьому ім'я блока розташовується збоку від його зображення. На рис. 14.9 показане розташування портів та імені блока після зміни орієнтації за допомогою команд **Flip block** і **Rotate block**.

Show/Hide drop shadow - відображення тіні від блока.

Show port labels - відображення міток портів.

Foreground color - установка кольорів зображення переднього плану виділеного блока, тобто кольори ліній і символів імені блока.

Background color - установка кольорів фону блока.

Можна також установити кольори фону для всього вікна моделі. Для цього використається команда **Screen color**.

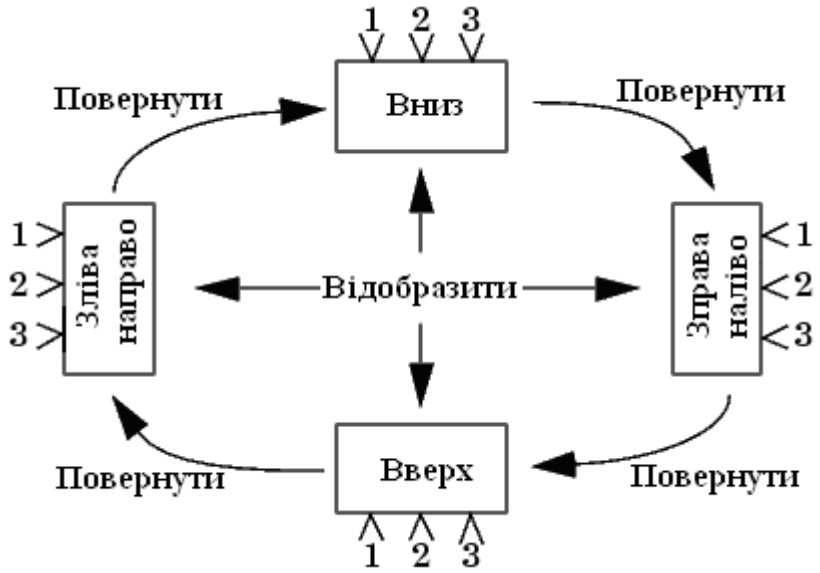


Рис. 14.9. Зміна орієнтації блока

За допомогою команд меню **Format** можна також вирівнювати текст у текстовому блоці (**Text alignment**), виділяти нескалярні лінії з'єднань (**Wide nonscalar lines**), відображати розмірності сигналів (**Signal dimensions**), виводити дані про тип портів (**Port data types**) і вводити вказівки на порядок виконання блоків у ході моделювання (**Execution order**) та ін.

14.3 Редагування ліній зв'язку

При побудові складних моделей виникає необхідність у зламі ліній зв'язку довільним образом і у створенні відводу ліній. Розглянемо яким образом здійснюється редагування ліній зв'язку.

При переміщенні блока, що входить у модель, лінії зв'язку не розриваються, а автоматично зламуються під кутом 90° (рис. 14.10).

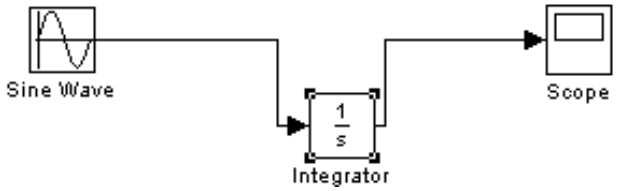


Рис. 14.10. Злам лінії при переміщенні блока

Якщо треба перемістити який-небудь відрізок лінії, його виділяють лівою кнопкою миші. При цьому покажчик миші приймає вид хрестика зі стрілками (рис. 14.11).

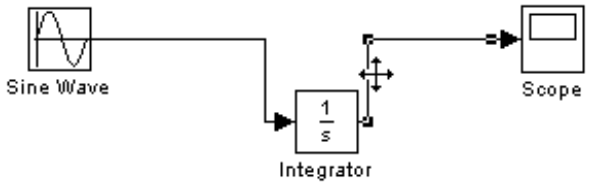


Рис. 14.11. Виділення відрізка лінії зв'язку

Далі, не відпускаючи лівої кнопки миші можна переміщати відрізок на нове місце. Кут зламу лінії, при цьому, залишається 90º (рис. 14.12).

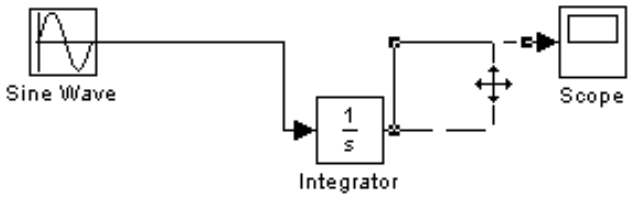


Рис. 14.12. Переміщення відрізка лінії зв'язку

Таким же чином можна переміщати і точки зламу ліній. Для цього необхідно виділити необхідну точку зламу лівою кнопкою миші, і як тільки покажчик прийме вид кружка, почати переміщення (рис. 14.13).

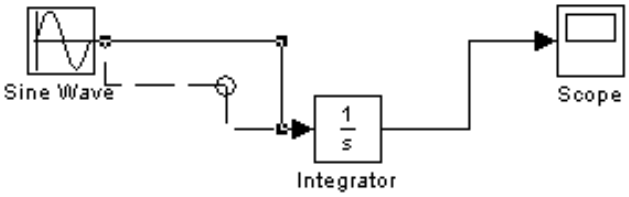


Рис. 14.13. Переміщення точки зламу лінії зв'язку

Якщо буде потреба, можна задавати новий злам лінії зв'язку. Для цього треба виділити лінію, натиснути клавішу Shift і одночасно ліву кнопку миші в потрібній точці (покажчик миші також приймає вид кружка). Тепер можна переносити нову точку зламу в довільне місце вікна. Таким чином можна задавати лінії

будь-якої форми (рис. 14.14).

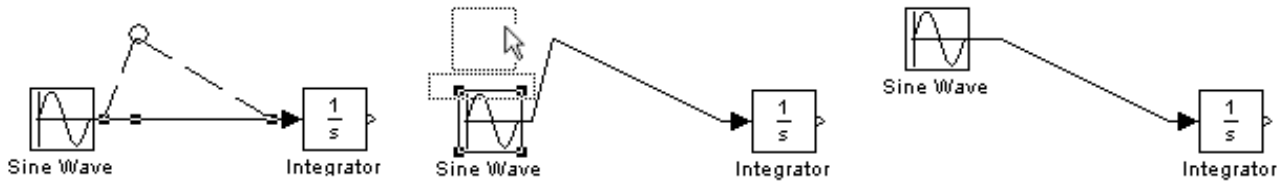


Рис. 14.14. Завдання лінії з'єднання довільної форми

У більшості випадків при побудові моделей систем керування необхідно створення відгалужень лінії зв'язку (наприклад, для створення замкнутих систем). Для цього в тім місці лінії зв'язку, де потрібно створити відгалуження, натискається права кнопка миші і утримується в процесі проведення відгалуження. При цьому покажчик миші приймає вид хреста. Рис. 14.15 відображає процес створення відгалуження лінії зв'язку.

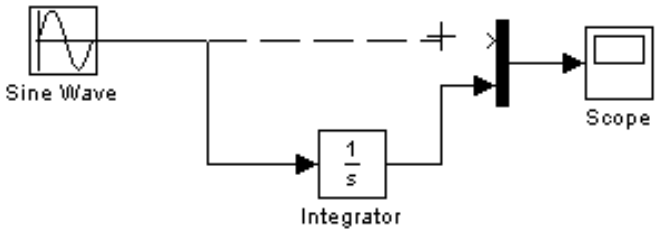


Рис. 14.15 Створення відгалуження лінії зв'язку

На схемі, зображеній на рис. 14.15, використовується новий блок мультіплексора сигналів із двома входами `Mux` з бібліотеки блоків `Signal Routing`. Мультіплексор забезпечує підключення декількох незалежних каналів до одного каналу. У цьому випадку він дозволяє одночасно спостерігати на одному осцилографі вихідні сигнали як джерела, так й інтегратора.

Вікно параметрів блока `Mux` наведене на рис. 14.16. Нагадаємо, що вікна параметрів блока викликаються подвійним щигликом лівої кнопки миші на зображенні блока (альтернативний спосіб – використання меню **Edit** або контекстного меню). Вікно параметрів блока `Mux` містить два текстові поля. У полі **Number of inputs** (Число входів) задається кількість входів мультіплексора. Тут можна також вводити імена сигналів, розділені комами.

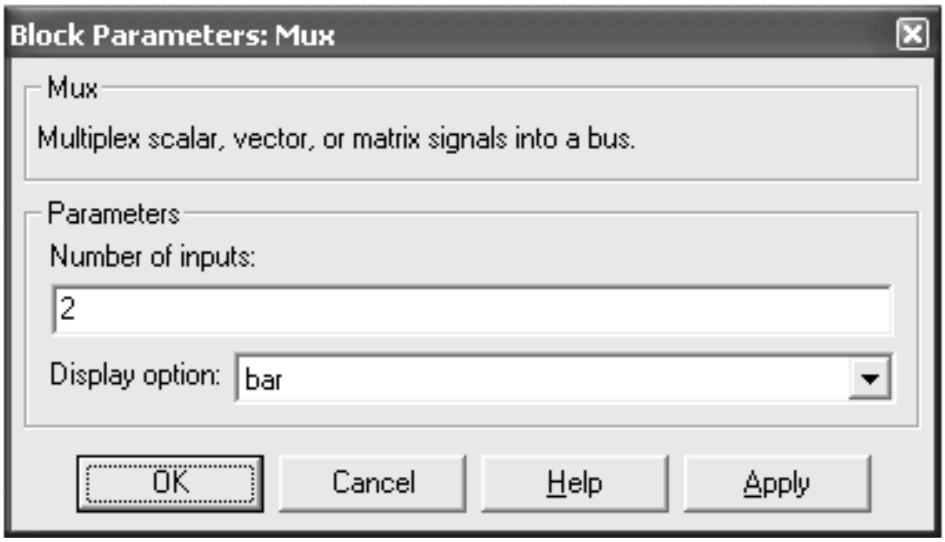


Рис. 14.16. Діалогове вікно параметрів блока `Mux`

У полі **Display option** (Опції зображення) вибирається один із трьох варіантів зображення блока:

- none – усередині блока відображається слово `Mux`;

- signals – відображає ім'я сигналу біля відповідного входу;

- bar – зображення блока заливається суцільними кольорами (за замовчуванням чорним).

Повернемося до моделі, процес побудови якої представлений на рис. 14.15. Графік, що показує результати моделювання (рис. 14.17), наочно демонструє властивість інтегруючої ланки: її вихідний сигнал відстає за фазою від вхідного на 90°.

Оскільки, як показує досвід, операція інтегрування функцій часу викликає певні утруднення у вивчаючих теорію керування, зупинимося на отриманих результатах інтегрування більш докладно.

Якщо сигнал на вході інтегратора позитивний і зростає із часом, то вихідний сигнал збільшується із все зростаючою швидкістю. Швидкість зміни вихідного сигналу максимальна в момент часу, коли вхідний сигнал приймає максимальне значення.

Якщо вхідний сигнал інтегратора позитивний, але убуває із часом, то вихідний сигнал на цьому інтервалі часу збільшується, але із все убуваючою швидкістю.

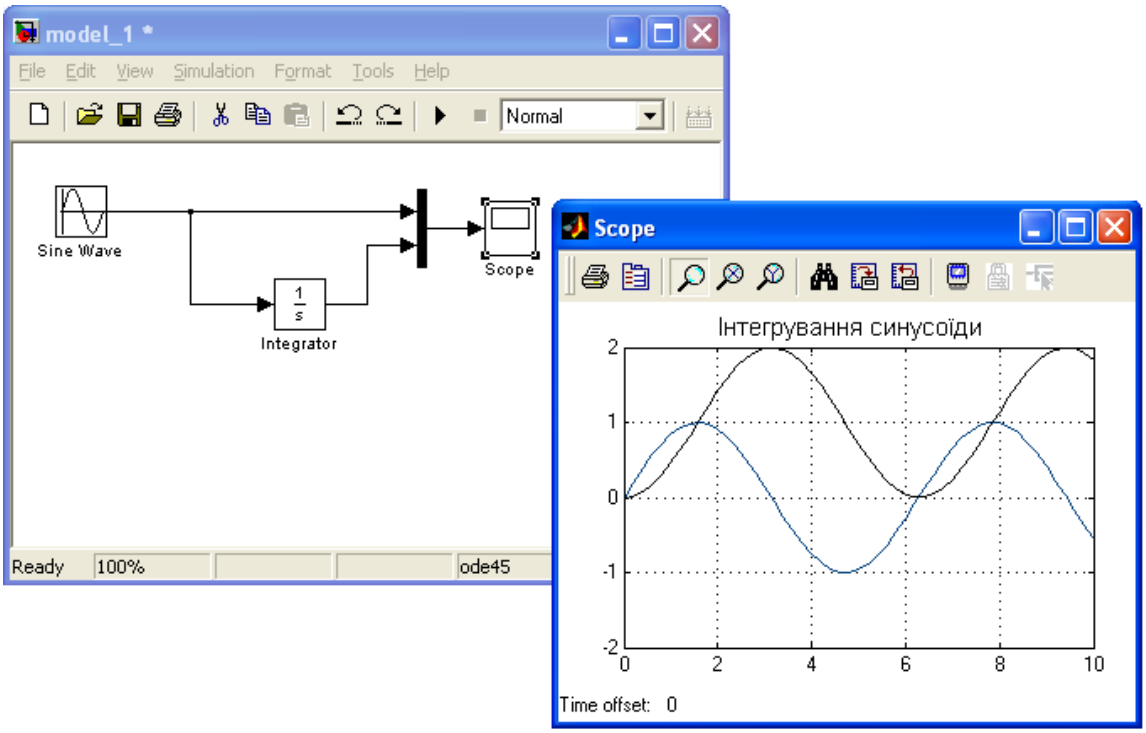


Рис. 14.17. Інтегрування синусоїди

Якщо вхідний сигнал має негативне значення, то вихідний сигнал убуває, причому з максимальною швидкістю вихідна величина убуває в той момент, коли вхідна величина має найменше значення (найбільше за абсолютною величиною, але негативне значення).

У моменти часу, коли вхідний сигнал дорівнює нулю, швидкість зміни вихідної величини також дорівнює нулю. Звичайна помилка полягає в тім, що при нульовому вхідному сигналі дорівнює нулю і вихідний сигнал інтегратора, а не швидкість його зміни. Це невірно. Значення вихідного сигналу в будь-який момент часу визначаються всією передісторією величини вхідного сигналу, а не його миттєвим значенням. Коли вхідна величина стає рівною 0, то «інтегрувати нема чого» і вихідна величина не міняється в часі, але не стає рівною нулю. У цьому легко можна переконатися, якщо у якості джерела сигналу використати джерело прямокутних імпульсів `Pulse Generator`. Сигнал на виході інтегратора буде являти собою східчасто наростаючу лінію (рис. 14.18). Коли рівень сигналу на виході генератора дорівнює нулю, сигнал на виході інтегратора залишається незмінним.

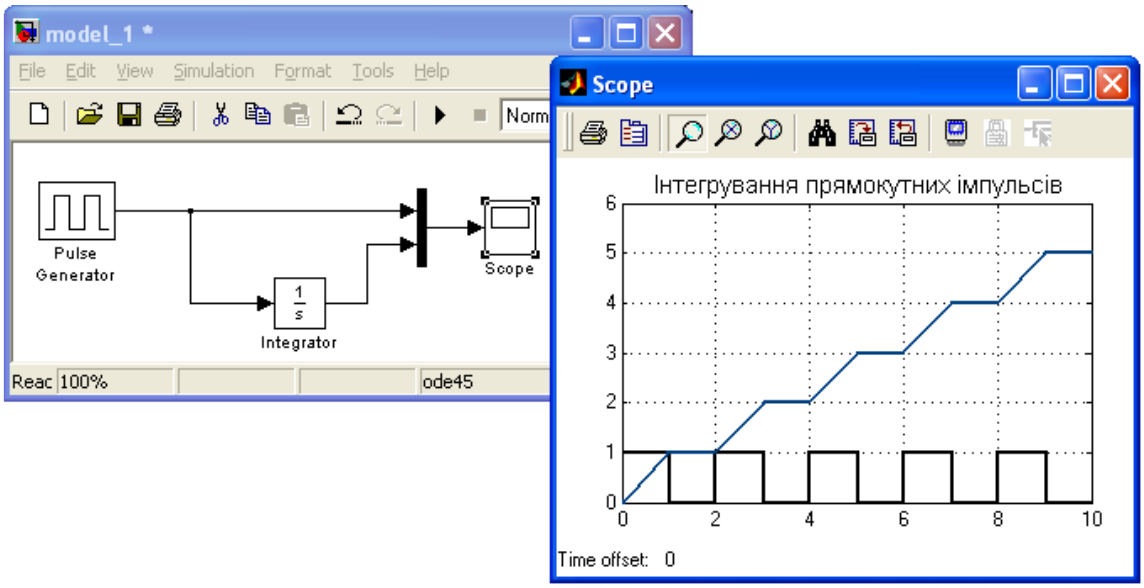


Рис. 14.18. Інтегрування прямокутних імпульсів

Зазначені вище властивості інтегруючих ланок дозволяють використовувати їх у регуляторах для поліпшення якості роботи систем керування в сталих режимах, про що мова йтиме далі.

14.4 Додавання і форматування текстових написів

В Simulink є можливість введення і форматування тексту. Це необхідно для постачання моделі коментарями, які можуть містити інформацію про призначення моделі, принципах її роботи та ін. Коментарі можна розташовувати в будь-якому місці моделі.

Для введення тексту достатньо вказати мишею місце напису і двічі клацнути лівою кнопкою миші. При цьому з'явиться рамка, що визначає область текстового блока, і курсор введення (рис.14.19). Тепер можна ввести потрібний напис. Для переходу на новий рядок потрібно натиснути **Enter**. Щиглик мишею поза текстовою областю дозволяє вийти з режиму введення тексту.

Як показує досвід, Simulink, як і взагалі MATLAB, не завжди коректно працює з кирилицею. Тому без вагомих підстав користуватися символами кирилиці не рекомендується.

Для форматування тексту необхідно відкрити текстовий блок щигликом лівої кнопки миші та викликати підменю **Text alignment** (Вирівнювання тексту) меню **Format**. Тут доступні три команди: **Left** (По лівому краю), **Center** (По центру) і **Right** (По правому

краю). Ці ж команди доступні з контекстного меню. Для редагування параметрів шрифту необхідно викликати вікно **Set Font** (рис. 14.8).

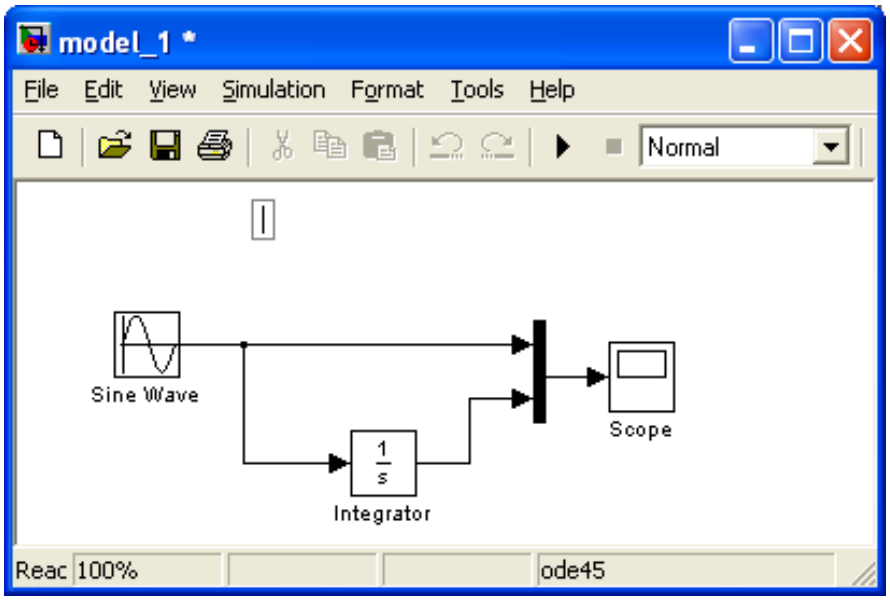


Рис. 14.19. Початок введення текстового напису

У такий же спосіб, як і при введенні коментарів, можна змінювати й імена блоків. Для цього досить клацнути лівою кнопкою миші на імені блока. На рис. 14.20 показана зміна імені блока `Scope`.

Рис. 14.20. Зміна імені блока

Завдання для самостійної роботи

1. Побудуйте модель, яка б дозволяла в одному вікні спостерігати синусоїдальний сигнал, інтеграл від синусоїдального сигналу та його похідну (блок `Derivative` з бібліотеки `Continuous`).

2. Дайте моделі ім’я "Операції з синусоїдою".

3. Змініть розмір блоку `Scope` та змініть його ім’я на "Осцилограф".

15. МОДЕЛЮВАННЯ ЛІНІЙНИХ СИСТЕМ АВТОМАТИЧНОГО КЕРУВАННЯ

Дана глава присвячена питанню використання Simulink для моделювання лінійних безперервних систем автоматичного керування. Тут ми більш докладно розглянемо вже відомий нам блок `Integrator`, а також познайомимося з новими блоками, необхідними при розгляді систем керування – блоком підсилювача (`Gain`), блоком передаточної функції (`Transfer Fcn`), блоком диференціювання (`Derivative`), а також суматором (`Sum`).

15.1. Структурні схеми безперервних систем

Як відомо, безперервні (аналогові) системи керування моделюються системами звичайних диференціальних рівнянь, які в загальному випадку є нелінійними. Однак більшість реальних процесів при малих відхиленнях змінних від робочої точки можна розглядати як лінійні. Лінійні системи описуються лінійними диференціальними рівняннями (з постійними коефіцієнтами). Такий підхід значно спрощує аналіз систем, але робить отримані результати наближеними. Моделювання на ЕОМ, і зокрема в Simulink, може істотно допомогти адекватно використати лінійну стаціонарну модель реальної системи.

В Simulink моделі стаціонарних безперервних систем формуються у вигляді звичних структурних схем. Основними елементами структурних схем є блоки передаточних функцій (`Transfer Fcn`). Крім того, структурні схеми можуть включати блоки: `Gain` (Підсилювач), `Sum` (Суматор), `Derivative` (Диференціювання) і `Integrator` (Інтегратор). Розглянемо ці блоки докладніше.

15.2 Блок Gain

Найпростішим елементом структурної схеми є підсилювач. Вихідна величина цієї ланки прямо пропорційна вхідній величині, тому його рівняння має такий вигляд:

$$y(t) = k \cdot x(t). \qquad (15.1)$$

Коефіцієнт k зветься коефіцієнтом підсилення ланки. Передаточна функція підсилювальної ланки збігається з коефіцієнтом підсилення ланки:

$$W(s) = k. \qquad (15.2)$$

В Simulink підсилювач представляється блоком `Gain` з бібліотеки `Math Operations`. На рис. 15.1 наведене графічне зображення блока Gain.

Рис. 15.1. Блок `Gain`

Прикладом підсилювальної ланки є важільна система, показана на рис. 15.2.

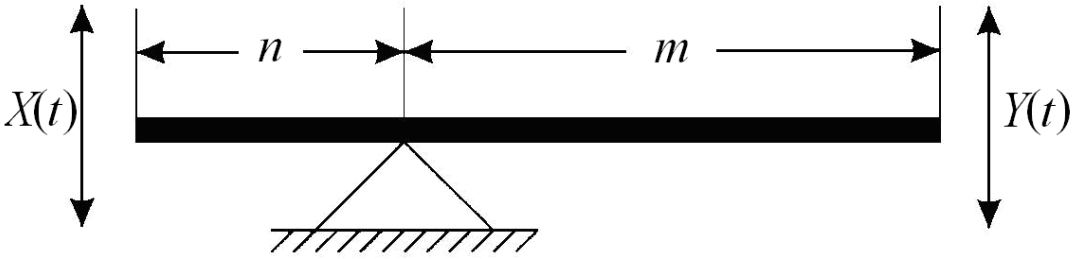


Рис. 15.2. Важільна система.

Величини $x(t)$ і $y(t)$ зв'язані співвідношенням:

$$y(t) = \frac{m}{n} \cdot x(t) = k \cdot x(t). \qquad (15.2)$$

Іншим прикладом може служити чотирьохполюсник, показаний на рис. 15.3. Напруги $U_{вх}$ і $U_{вих}$ зв'язані співвідношенням:

$$y(t) = \frac{R_2}{R_1 + R_2} x(t) = k \cdot x(t). \qquad (15.3)$$

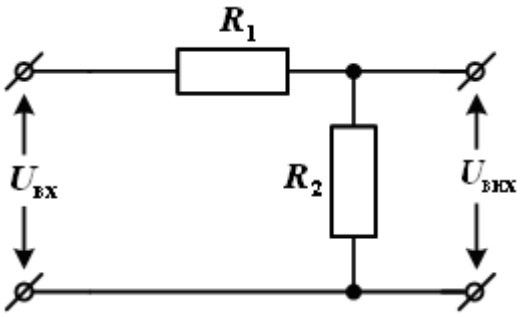


Рис. 15.3. Чотирьохполюсник

Рівнянням типу (15.1) описується і закон Гука:

$$F = kx, \qquad (15.4)$$

де F - прикладена сила, x - переміщення кінця пружини, k - жорсткість пружини. Таким чином, пружину також можна описати моделлю у вигляді блока `Gain`.

Розглянуті приклади ще раз ілюструють той факт, що процеси зовсім різної фізичної природи можуть моделюватися однаковими за структурою рівняннями, тобто ілюструють універсальність методів математичного моделювання.

Вікно параметрів блока `Gain` наведене на рис. 15.4.

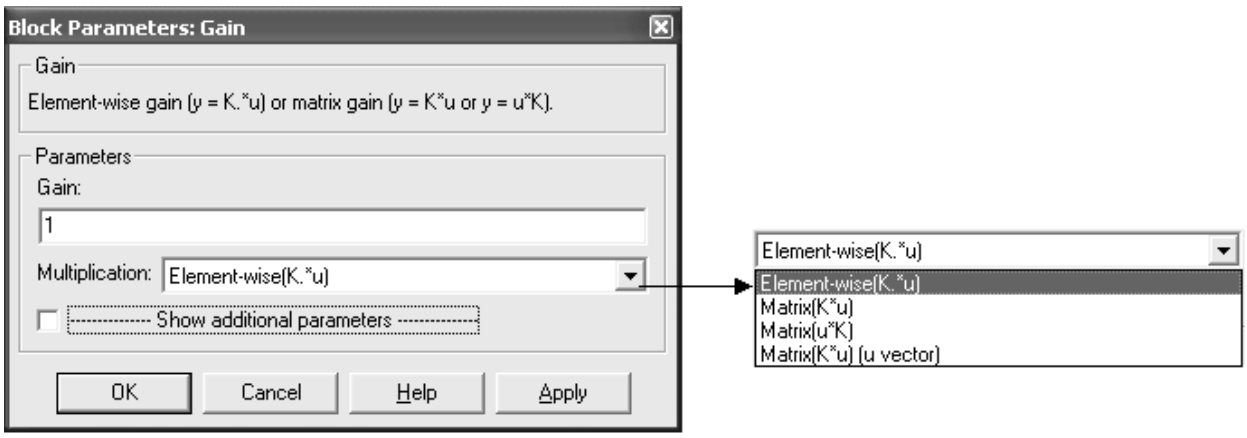


Рис. 15.4. Вікно параметрів блоку `Gain`

У цьому вікні можна встановити наступні параметри блока.

Gain – введення значення коефіцієнта підсилення, що може бути скаляром (як дійсним, так і комплексним), вектором або матрицею.

Multiplication – у цьому полі вибирається зі списку спосіб виконання операції підсилення:

- **Element-wise K*u** (поелементний) – кожен елемент входу помножується на кожен елемент, заданий у полі **Gain**.

- **Matrix K*u** (матричний) – здійснюється матричне множення матриці коефіцієнтів підсилення на матрицю входу (матриця входу є другим співмножником).

- **Matrix u*K** (матричний) - множення матриці входу на матрицю підсилення (матриця підсилення є другим співмножником).

- **Matrix(K*u)(u vector)** – кожен елемент вектора підсилення помножується на відповідний елемент входу **u**. У цьому випадку вихідна величина також є вектором, довжина якого повинна збігатися з довжиною вектора входу.

- **Show additional parameters** (Показати додаткові параметри) - при встановленому прапорці можна вибрати тип даних, з яким буде працювати підсилювач або зробити масштабування підсилювача.

У процесі моделювання іноді буває зручніше користуватися іншим блоком, робота якого також описується рівнянням (15.1). Мова йде про блок `Slider Gain`, що також знаходиться в бібліотеці `Math Operations`. Зовнішній вигляд блока і діалогове вікно цього блока представлені на рис. 15.5.

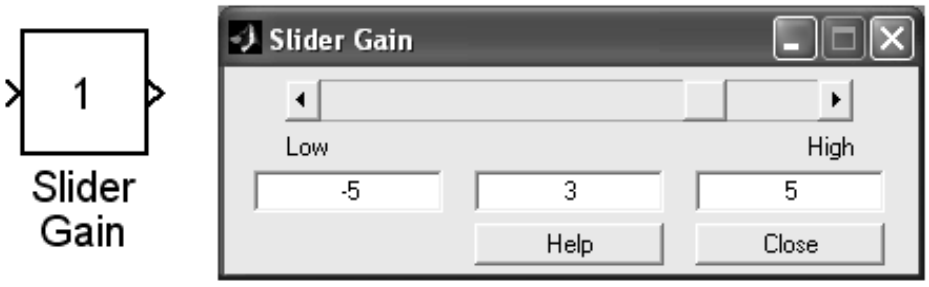


Рис. 15.5. Зовнішній вигляд і діалогове вікно блока `Slider Gain`

Блок `Slider Gain` дозволяє змінювати коефіцієнт підсилення в процесі моделювання за допомогою повзунка (slider). Для зміни коефіцієнта підсилення блока `Slider Gain` необхідно пересунути повзунок у відповідну сторону. Переміщення повзунка вправо приведе до збільшення коефіцієнта підсилення, переміщення вліво – до його зменшення. Зміна коефіцієнта підсилення буде виконуватися в межах діапазону заданого параметрами **Low** і **High.**

Якщо клацнути за допомогою миші на одній зі стрілок по краях повзунка, то коефіцієнт підсилення зміниться на 1% від установленого діапазону. Якщо клацнути за допомогою миші на самій шкалі регулятора ліворуч або праворуч від повзунка, то коефіцієнт підсилення зміниться на 10% від установленого діапазону. Можна також просто задати необхідне значення коефіцієнта у середньому вікні блока.

15.3 Блок `Sum`

Як правило, сучасна система керування має складну структуру. При цьому часто необхідно мати можливість складати кілька вхідних сигналів. Це може знадобитися, наприклад, для враховування впливу на систему зовнішніх збурень, для синтезу паралельних коригувальних пристроїв, та й будь-яка система немислима без зворотного зв'язку. Організовувати подібні структури допомагає блок `Sum` з бібліотеки `Math Operations`. Блок `Sum` служить для знаходження алгебраїчної суми двох або більше вхідних змінних, кожній з яких привласнюється знак операції додавання «+» або віднімання «-». Так, для блока `Sum`, показаного на рис. 15.6, справедливе вираження

$$x = u - y. \qquad (15.5)$$

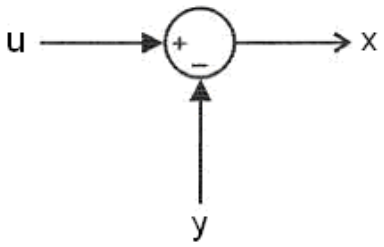


Рис. 15.6. Блок `Sum`

Блок Sum повинен мати, принаймні, один вхідний та один вихідний порти. Вікно його властивостей наведене на рис. 15.7.

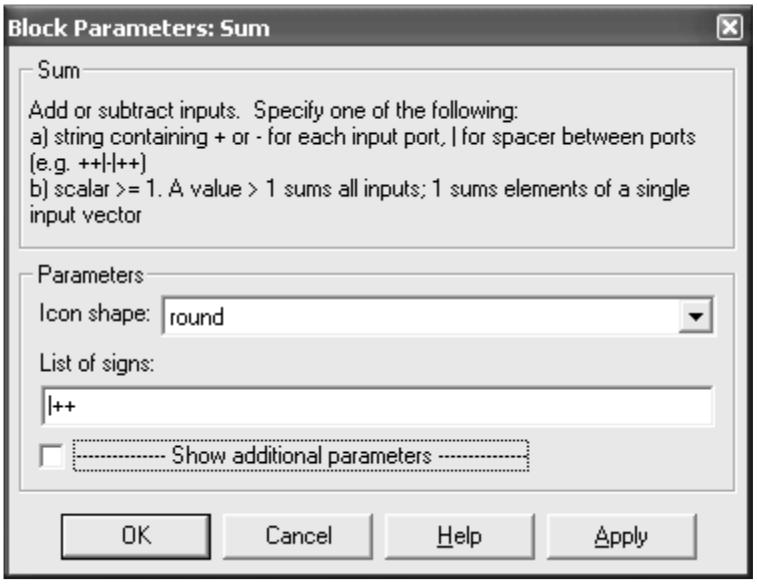


Рис. 15.7. Вікно параметрів блока Sum

У цьому вікні можна вказати наступні параметри блока.

Icon shape - цей список, що розкривається, дозволяє вибрати форму блока:

- round – окружність,

- rectangular – прямокутник.

List of sign – у цьому полі задається список знаків, які визначають кількість входів і задають арифметичні дії над відповідними вхідними сигналами блока. У списку можна використати наступні знаки: + (плюс), - (мінус) і | (роздільник знаків, його положення в списку визначає, який вхідний порт блока буде закритий).

Кількість входів можна також задати константою. У цьому випадку всі входи будуть підсумовуючими. Якщо у якості списку знаків указати цифру 1 (один вхід), то блок можна використати для визначення суми елементів вектора. При цьому усередині піктограми блока з'явиться знак Σ (рис. 15.8).

Show additional parameters (Показати додаткові параметри) - при встановленому прапорці можна вибрати опцію спостереження за тим, щоб вхідні змінні блока мали один тип (**Require all inputs to have same data type**), встановлювати тип вихідних даних (**Output data type mode**, **Output data type**) та ін.

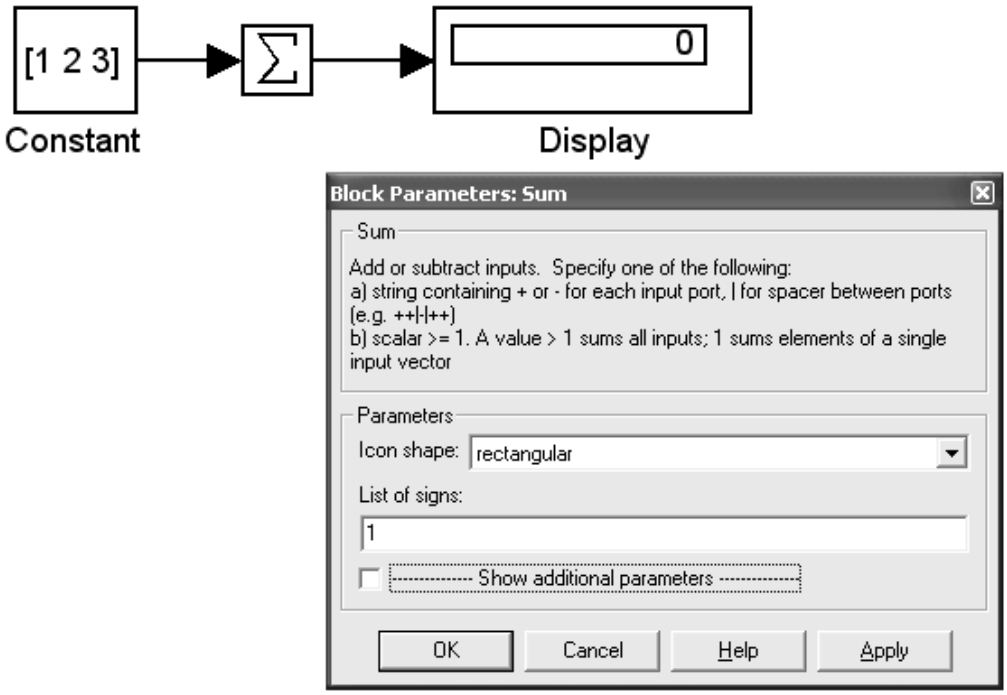


Рис. 15.8. Використання блока `Sum`
для підсумовування елементів вектора

15.4 Блок `Derivative`

Блок `Derivative` (рис. 15.9) робить обчислення похідної вхідного сигналу за часом відповідно до рівняння

$$y(t) = \frac{dx(t)}{dt}. \qquad (15.6)$$

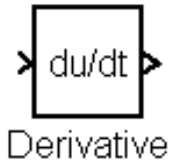


Рис. 15.9. Блок `Derivative`

Таким чином, блок `Derivative` моделює ідеальну ланку, що диференціює, з передаточною функцією $W(s) = s$. Прикладом такої ланки є тахогенератор, якщо його вхідним сигналом вважати кут φ

повороту вала, вихідним – напруга U, а інерційністю тахогенератора можна зневажити:

$$U(t) = k \cdot \frac{d\varphi(t)}{dt}, \tag{15.7}$$

де k - коефіцієнт передачі тахогенератора.

У якості ланки, що диференціює, можна розглядати механічну систему, представлену на рис. 15.10. Її математична модель описується рівнянням (15.8)

$$F = c \cdot \frac{dx(t)}{dt}, \tag{15.8}$$

де F - діюча сила, x - переміщення, c - коефіцієнт тертя.

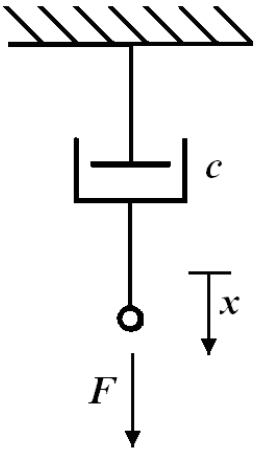


Рис. 15.10. Механічна система, що диференціює

15.5 Блок `Integrator`

Із блоком `Integrator` (Інтегратор) ми вже зустрічалися в попередній главі. Рівняння, що зв'язує вихідний і вхідний сигнали блока має вигляд:

$$y(t) = \int_{t_0}^{t} x(t)dt + y_0, \tag{15.9}$$

де $x(t)$ – вхідний сигнал, $y(t)$ – вихідний сигнал, y_0 – початкові умови.

Усередині піктограми блока зображена передаточна функція інтегруючої ланки 1/*s* (рис. 15.11, а), що моделюється рівнянням (15.9) при нульових початкових умовах, тобто при $y_0 = 0$.

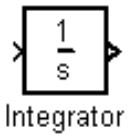


Рис. 15.11. Піктограма блока `Integrator`

Прикладами реальних елементів, еквівалентні схеми яких зводяться до інтегруючої ланки, є електронний інтегратор, безінерційний двигун постійного струму, якщо вхідна величина – напруга на якорі, а вихідна – кут повороту вала, а також конденсатор (рис. 15.12), вхідною змінною якого є струм *i* через ємність, а вихідна змінна – напруга $U_к$ між обкладинками конденсатора.

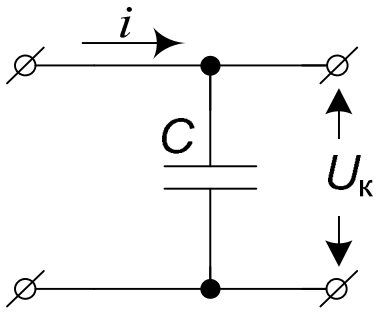


Рис. 15.12. Інтегруюче електричне коло

Розглянемо параметри блока `Integrator`. Діалогове вікно блока наведене на рис. 15.13. У цьому вікні можна встановити наступні параметри.

External reset (Зовнішнє скидання) – у цьому меню задається тип блока: звичайний інтегратор або інтегратор зі скиданням. Інтегратор зі скиданням у момент перемикання встановлює (скидає) величину вихідного сигналу на початкове значення. Список, що розкривається, містить п'ять пунктів:

- **none** (немає) - скидання не виконується (встановлено за замовчуванням);

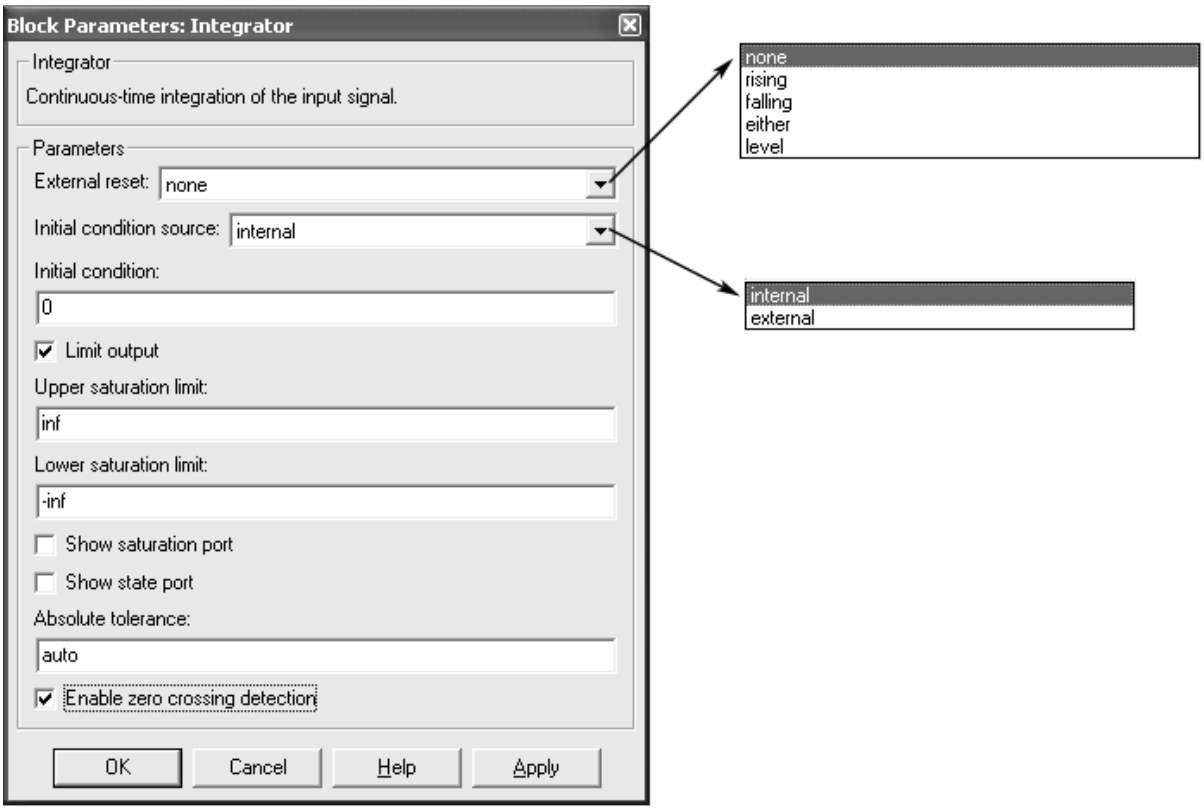


Рис. 15.13. Діалогове вікно параметрів блока `Integrator`

- **rising** (зростання) – вихідний сигнал інтегратора скидається в початкове значення, коли при збільшенні керуючого сигналу (сигналу скидання) відбувається перетинання нульового значення;

- **falling** (убування) - вихідний сигнал інтегратора скидається в початкове значення, коли при зменшенні керуючого сигналу відбувається перетинання нульового значення;

- **either** (будь-який) – сполучення останніх двох пунктів: вихід інтегратора скидається до початкового значення, якщо керуючий сигнал перетинає нуль як зверху, так і знизу;

- **level** (рівень) - скидання виконується якщо сигнал на керуючому вході стає не рівним нулю.

У тому випадку, якщо обрано інтегратор зі скиданням (будь-який пункт меню **External reset** крім **none**), то на зображенні блока з'являється додатковий керуючий вхід. Поруч із додатковим входом буде показана умовна позначка керуючого сигналу: **rising** , **falling** , **either** , **level** .

Initial condition source (Джерело початкових умов) – список, що розкривається, містить два пункти:

- internal (внутрішній) – при виборі цього пункту початкові умови визначаються значенням, уведеним у текстовому полі **Initial condition** (Початкові умови), що може бути як константою, так і змінною, та за замовчуванням дорівнює 0.

- external (зовнішній) – при виборі цього пункту на зображенні блока з'являється додатковий вхід, позначений як x_0. Величина сигналу на цьому вході буде задавати початкове значення вихідної змінної блока.

При встановленому прапорці напроти позиції **Limit output** (Обмеження по виходу) відбувається обмеження рівня вихідного сигналу інтегратора. Значення вихідного сигналу змінної не буде перевищувати значення, задане в текстовому полі **Upper saturation limit** (Верхня межа насичення), і буде менше (за модулем) або дорівнювати значенню, введеному в текстовому полі **Lower saturation limit** (Нижня межа насичення). За замовчуванням значення верхньої та нижньої меж насичення дорівнюють відповідно `inf` та `-inf` (нагадаємо, що `inf` – вбудована змінна MATLAB, що відповідає +∞).

Для виробітку сигналу, який показує, що інтегратор перебуває в стані насичення, необхідно поставити прапорець напроти позиції **Show saturation port** (Показати порт насичення). При цьому в блоці з'являється ще один вихідний порт, вихідний сигнал якого може приймати три значення:

а) **+1**, якщо вихідний сигнал інтегратора досяг верхньої межі насичення;

б) **0**, якщо інтегратор не перебуває в стані насичення;

в) **-1**, якщо вихідний сигнал інтегратора досяг нижньої межі насичення.

При виборі опції **Show state port** (Показати порт стану) зверху блока `Integrator` з'являється додатковий вихідний порт – порт стану. Даний порт необхідно використовувати в тому випадку, якщо вихідний сигнал інтегратора потрібно подати на порт скидання або порт початкових умов цього ж інтегратора. Вихідний сигнал із цього порту може використовуватися також для організації взаємодії з керованою підсистемою.

На рис. 15.14. зображений блок `Integrator` з усіма активізованими портами.

Введення числового значення в текстовому полі **Absolute tolerance** (Абсолютна погрішність) дозволяє задати для даного блока інші значення абсолютної погрішності, чим значення погрішності для вихідних змінних інших блоків моделі, встановлені в діалоговому вікні **Simulation Parameters**.

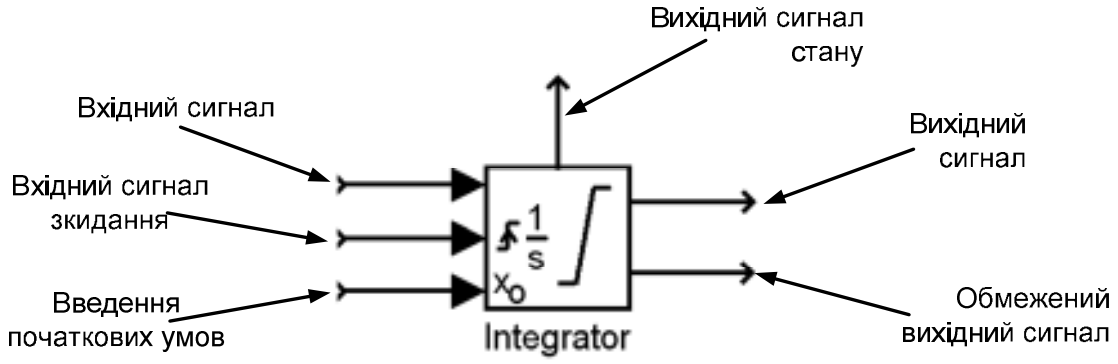


Рис. 15.14. Блок `Integrator` з усіма активізованими портами

Якщо встановлено прапорець напроти позиції **Enable zero crossing detection** (Дозволити визначення перетинання нуля), а також обрана опція **Limit output,** блок `Integrator` використовує функцію перетинання нуля для визначення моменту часу, у який відбуваються наступні події: скидання, вхід або вихід зі стану верхнього насичення, вхід або вихід зі стану нижнього насичення.

15.6 Блок `Transfer Fcn`

При побудові структури лінійної системи керування однією з найпоширеніших форм опису ланок блоків є передаточна функція. В Simulink передаточна функція ланки моделюється блоком `Transfer Fcn` з бібліотеки `Continuous` (рис. 15.15).

Блок `Transfer Fcn` задає передаточну функцію у вигляді відношення поліномів:

$$W(s)=\frac{y(s)}{x(s)}=\frac{\mathbf{B}(s)}{\mathbf{A}(s)}=\frac{b_m s^m+b_{m-1}s^{m-1}+\ldots+b_1 s+b_0}{a_n s^n+a_{n-1}s^{n-1}+\ldots+a_1 s+a_0}, \qquad (15.10)$$

де m і n – порядок чисельника і знаменника передаточної функції, причому $n \geq m$; $\mathbf{B}(s)$ – вектор або матриця коефіцієнтів чисельника; $\mathbf{A}(s)$ - вектор коефіцієнтів знаменника.

Блок `Transfer Fcn` має наступні параметри.

Numerator (Чисельник) – у цьому полі задається вектор-рядок або матриця коефіцієнтів полінома чисельника передаточної функції в порядку убування ступенів багаточлена, тобто [b_m b_{m-1} b_{m-2} … b_0].

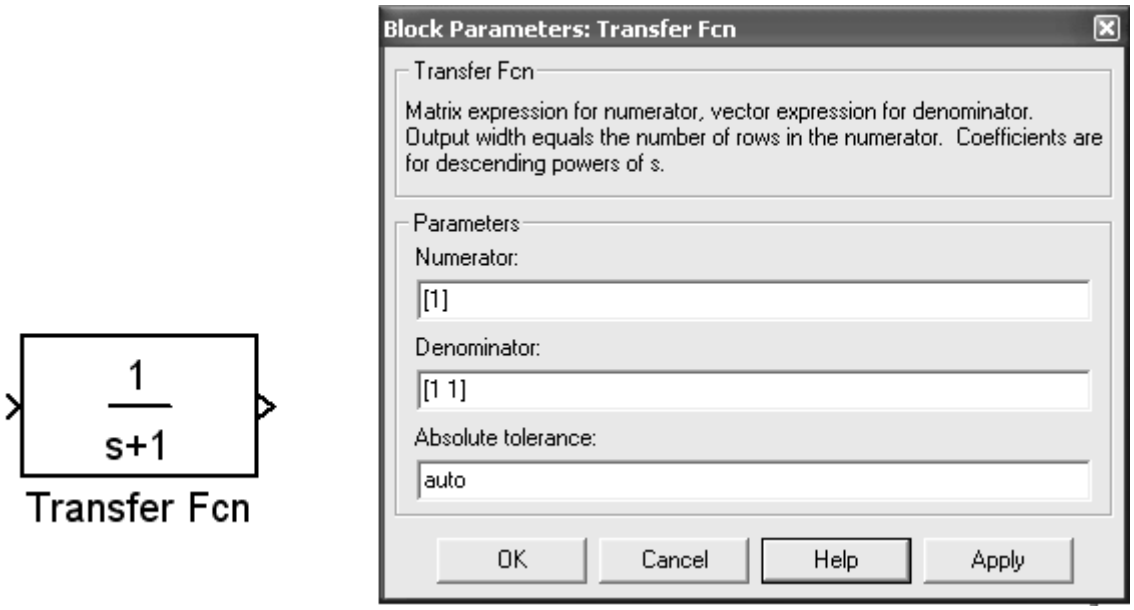


Рис. 15.15. Зовнішній вигляд і вікно параметрів блока `Transfer Fcn`

Denominator (Знаменник) – тут задається вектор коефіцієнтів полінома знаменника: [a_n a_{n-1} a_{n-2} … a_0]...

Якщо який-небудь коефіцієнт багаточлена відсутній, то він вважається рівним нулю і також вказується при завданні багаточлена. Наприклад, на рис. 15.16 показано, як задавати наступну передаточну функцію:

$$W(s) = \frac{s}{3s^3 + 2s^2 + s}.$$

Якщо коефіцієнти чисельника задані матрицею, то блок `Transfer Fcn` моделює векторну передаточну функцію, яку можна інтерпретувати як кілька передаточних функцій, що мають однаковий знаменник, але різні поліноми чисельника. При цьому вихідний сигнал блока є векторним і кількість рядків матриці чисельника задає розмірність вихідного сигналу.

Поліноми **A**(s) і **B**(s) можуть бути задані безпосередньо в MATLAB. Такий випадок розглядається на рис. 15.17.

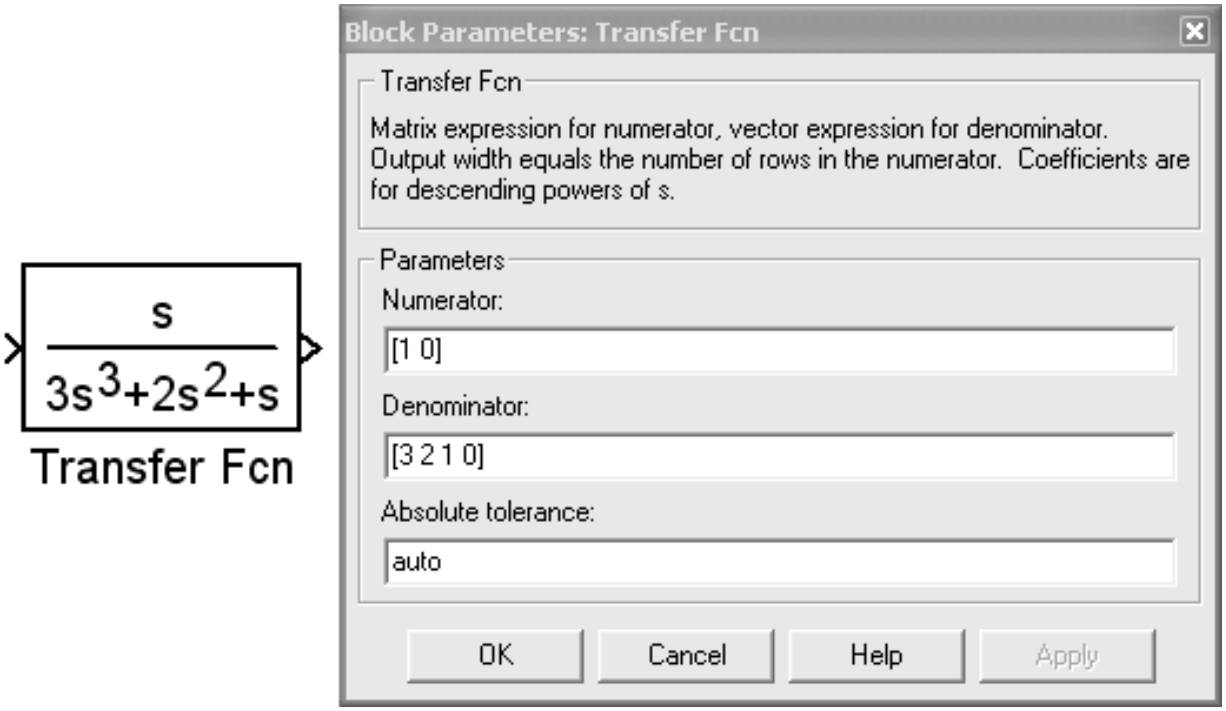


Рис. 15.16. Приклад введення передаточної функції

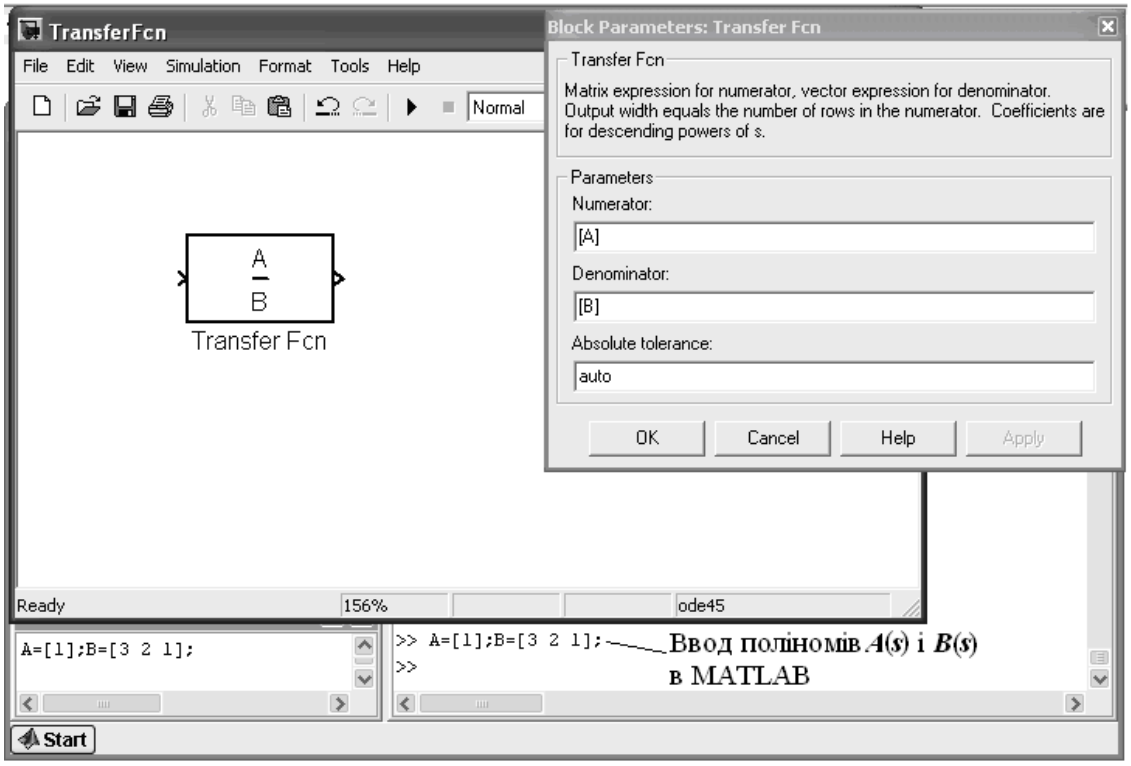


Рис. 15.17. Задання коефіцієнтів передаточної функції в MATLAB

Призначення опції **Absolute tolerance** (Абсолютна погрішність) у вікні параметрів блока `Transfer Fcn` аналогічне призначенню такої ж опції в блоці `Integrator`.

В Simulink блок передаточної функції `Transfer Fcn` використовується лише при нульових початкових умовах. Якщо ж потрібно розглянути систему при ненульових початкових умовах, то необхідно використати блоки `Transfer Fcn (with initial outputs)`, `Transfer Fcn (with initial states)` бібліотеки `Simulink Extras` → `Additional Linear`, а також блок `State-Space`, що буде розглянутий нижче.

15.7 Блок `Zero-Pole`

Розглянутий вище блок `Transfer Fcn` дозволяє будувати моделі систем керування у так званої TF формі (TF – скорочення від «Transfer Function» – передаточна функція). В Simulink, як і в Control Toolbox, є можливість будувати модель динамічних об'єктів за допомогою полюсів, нулів і коефіцієнта підсилення (ZPG-форма, скорочено від zero-pole-gain). Для цього служить блок `Zero-Pole` бібліотеки `Continuous`. Блок `Zero-Pole` дозволяє задавати передаточну функцію у вигляді:

$$W(s) = K\frac{\mathbf{Z}(s)}{\mathbf{P}(s)} = K\frac{(s-z_1)(s-z_2)...(s-z_m)}{(s-p_1)(s-p_2)...(s-p_n)}, \qquad (15.11)$$

де $\mathbf{Z}(s)$ – вектор або матриця нулів передаточної функції (корені полінома чисельника), $\mathbf{P}(s)$ – вектор полюсів передаточної функції (корені полінома знаменника), K – коефіцієнт передаточної функції, або вектор коефіцієнтів, якщо нулі передаточної функції задані матрицею. При цьому розмірність вектора K визначається числом рядків матриці нулів, а кількість нулів не повинна перевищувати число полюсів передаточної функції.

Піктограма блока `Zero-Pole` і вікно його параметрів представлене на рис. 15.18. Вікно параметрів блока містить чотири текстові поля.

Zeros (Нулі) – вектор або матриця нулів.

Poles (Полюси) – вектор полюсів.

Gain (Підсилення) – скалярний або векторний коефіцієнт підсилення.

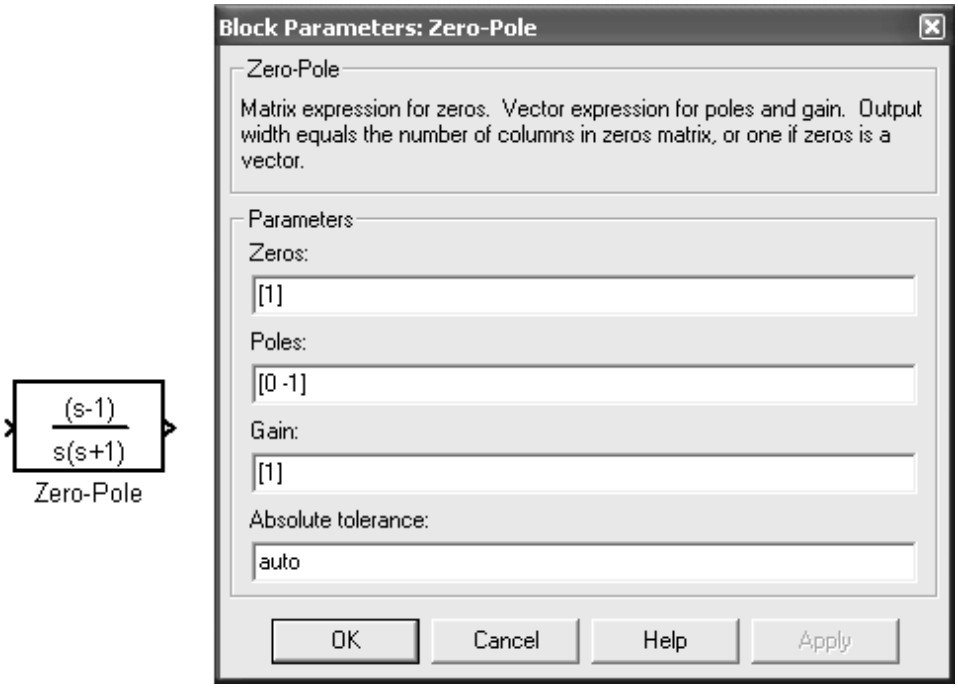


Рис. 15.18. Піктограма і діалогове вікно параметрів блока `Zero-Pole`

Absolute tolerance (Абсолютна погрішність) – призначення цього параметра таке ж, як і у попередніх блоках: введене значення скасовує для даного блока встановлену у вікні **Simulation Parameters** точність обчислень.

Початкові умови при використанні блока `Zero-Pole` покладаються нульовими.

Як й у блоці `Transfer Fcn` можливе завдання значень нулів, полюсів і коефіцієнта підсилення передаточної функції безпосередньо в MATLAB.

15.8 Блок `Transport Delay`

Цей блок забезпечує затримку вихідного сигналу відносно вхідного сигналу на заданий час. Таким чином, блок `Transport Delay` моделює ланку запізнювання. Нагадаємо, що ця ланка описується наступним рівнянням руху

$$y(t) = x(t - \tau), \qquad (15.12)$$

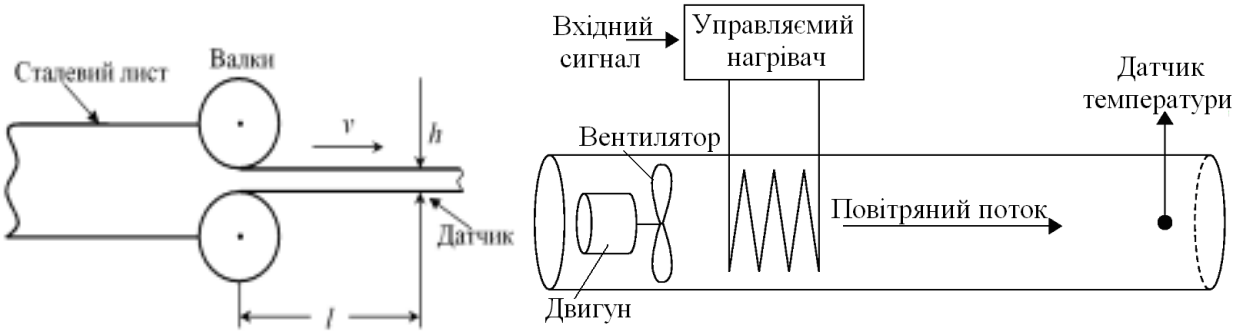
і передаточною функцією

$$W(s) = e^{-s\tau}, \qquad (15.13)$$

де τ - час запізнювання вихідного сигналу відносно вхідного.

Прикладами ланок запізнювання є конвеєрні лінії та транспортери, система регулювання товщини сталевого листа при прокаті, акустичні та інші лінії зв'язку, якщо зневажати втратою сигналу, лінії затримки на *LC* – колах тощо.

Так, при виготовленні сталевого листа необхідно точно витримувати його товщину *h*, що дорівнює відстані між зовнішніми поверхнями валків (рис. 15.19, а). Ця відстань звичайно встановлюється за допомогою гідравлічного виконавчого механізму і не піддається точному виміру, тому товщина сталевого листа вимірюється на деякому видаленні *l* від валків і датчик видає інформацію не про поточну товщину листа, а про значення, що було на $\tau = l/v$ секунд раніше. У системі нагрівання (рис. 15.19, б) тепле повітря досягне місця призначення також лише через якийсь час τ.



а б

Рис. 15.19. Приклади ланок запізнювання
а - система керування прокатним станом, б – система передачі тепла

Піктограма і вікно параметрів блоку `Transport Delay` представлені на рис. 15.20. Блок має наступні параметри:

- **Time Delay** - час затримки сигналу (не негативне значення);
- **Initial input** - початкове значення вихідного сигналу;
- **Initial Buffer size** - розмір пам'яті, що виділяється для зберігання затриманого сигналу; задається в байтах числом, кратним 8 (за замовчуванням 1024);

- **Pade order (for linearization)** - порядок ряду Паде, який використовується при лінеаризації. За замовчуванням дорівнює 0, але для підвищення точності лінеаризації може задаватися цілим позитивним числом.

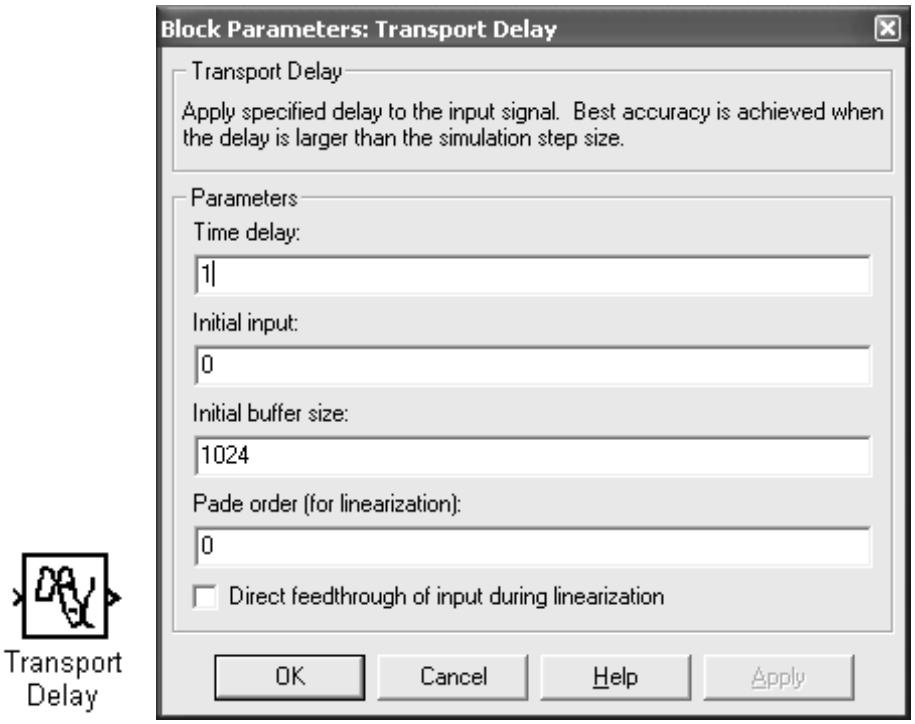


Рис. 15.20. Зображення та вікно параметрів блоку `Transport Delay`

При виконанні моделювання значення сигналу і відповідний йому модельний час зберігаються у внутрішньому буфері блока `Transport Delay`. Після закінчення часу затримки значення сигналу витягується з буфера та передається на вихід блока. У тому випадку, якщо шаги модельного часу не збігаються зі значеннями моментів часу для записаного в буфер сигналу, блок `Transport Delay` виконує апроксимацію вихідного сигналу.

Якщо початкового значення обсягу пам'яті буфера не вистачить для зберігання затриманого сигналу, Simulink автоматично виділить додаткову пам'ять. Після завершення моделювання в командному вікні MATLAB з'явиться повідомлення із вказівкою потрібного розміру буфера.

На рис. 15.21 показаний приклад використання блока `Transport Delay` для затримки східчастого сигналу на 1 с. Відповідні параметри блоку `Step` представлені на рис. 15.22, а. Для

підвищення точності моделювання максимальний шаг моделювання встановлений 0,01 с (рис. 15.22, б).

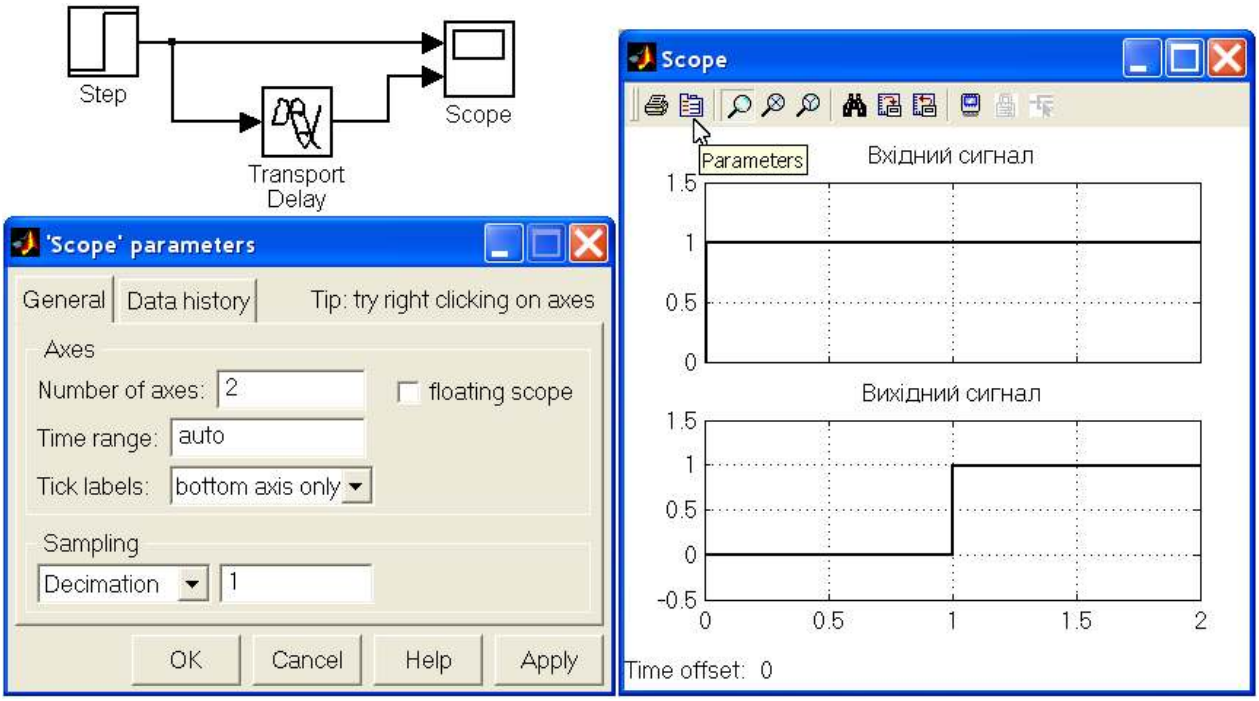
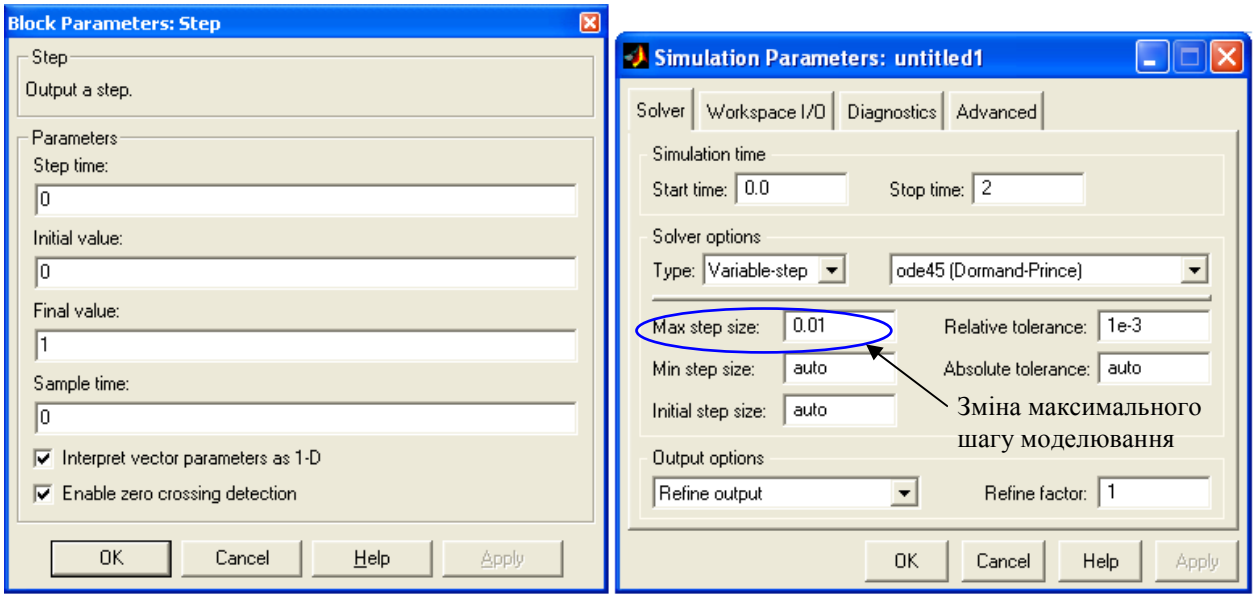


Рис. 15.21. Приклад використання блоку `Transport Delay`



а *б*

Рис. 15.22. Параметри блоків `Step` (*а*) та `Transport Delay` (*б*) для наведеного прикладу

У цьому прикладі для побудови декількох графіків у різних вікнах використовується лише один осцилограф. Для цього в

текстовому полі **Number of axes** (Число вісей) на вкладці **General** вікна параметрів осцилографа вказується необхідна кількість каналів (у нашому випадку 2). При цьому у осцилографа з'являється відповідне число входів. Для кожного входу будується окремий графік. Приклад завдання числа вісей осцилографа наведений на рис. 15.21.

15.9 Блок `Variable Transport Delay`

На відміну від попереднього блока, блок змінної затримки сигналу `Variable Transport Delay` (рис. 15.23) здійснює затримку вхідного сигналу на величину, задану сигналом керування. Параметри блока `Variable Transport Delay` аналогічні параметрам попереднього блока (рис. 15.20) за винятком першого - **Maximum delay** - максимального значення часу затримки сигналу. У тому випадку, якщо значення керуючого сигналу, що задає величину затримки, перевищує значення, задане параметром **Maximum delay,** то затримка виконується на величину **Maximum delay**.

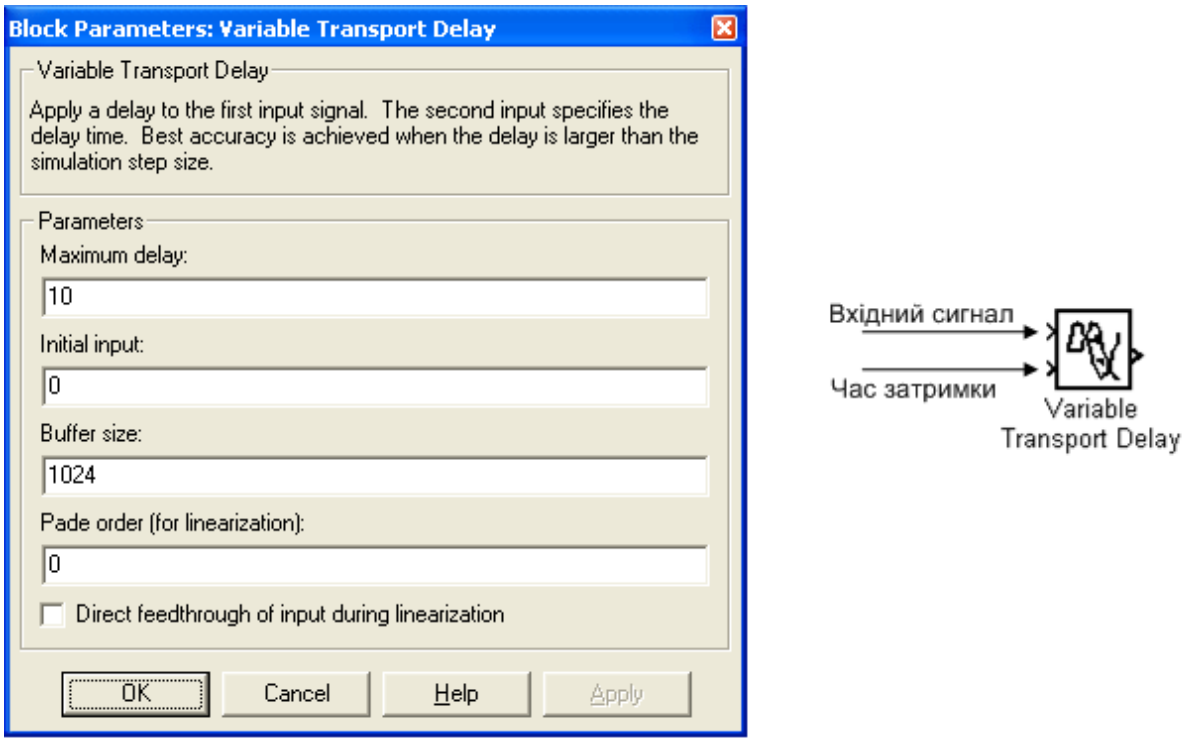


Рис. 15.23. Зображення і вікно параметрів блока `Variable Transport Delay`

Такий блок зручно використовувати, наприклад, при моделюванні систем із трубопроводом, коли швидкість рідини у трубопроводі змінюється з часом.

На рис. 15.24 показаний приклад використання блока `Variable Transport Delay`. Параметри блоків `Step` і `Variable Transport Delay` наведені на рис. 15.25. Спочатку моделювання вихід блоку визначається значенням, вказаним у полі **Initial input:** вікна параметрів блоку `Variable Transport Delay` (в нашому випатку 0,5). Цей сигнал підтримується доти, поки час моделювання не перевищить встановлений керуючим сигналом час затримки. У момент часу, що дорівнює 4 с, величина часу затримки прямокутного сигналу змінюється від 0,25 с до 1 с. Таксамо, як і в прикладі для блоку `Transport Delay`, тут максимальний шаг моделювання встановлений 0,01 с.

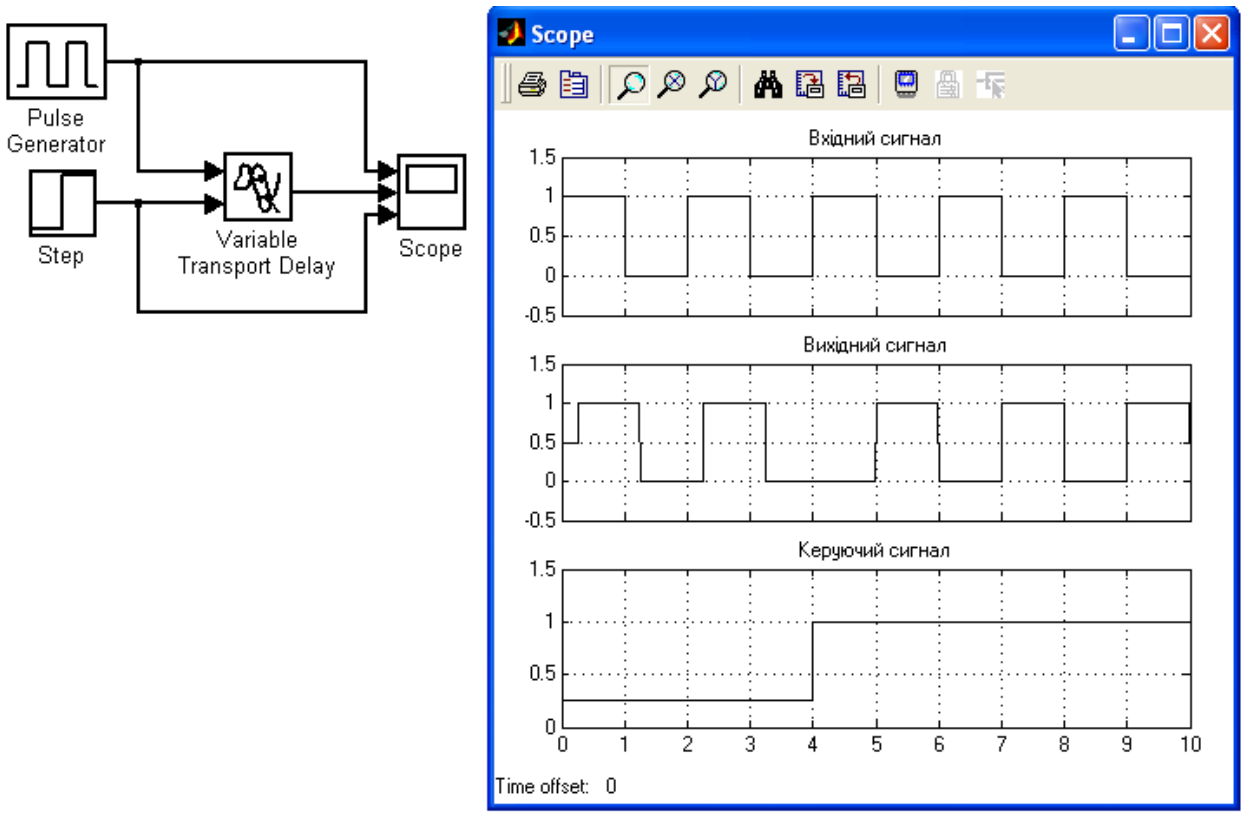
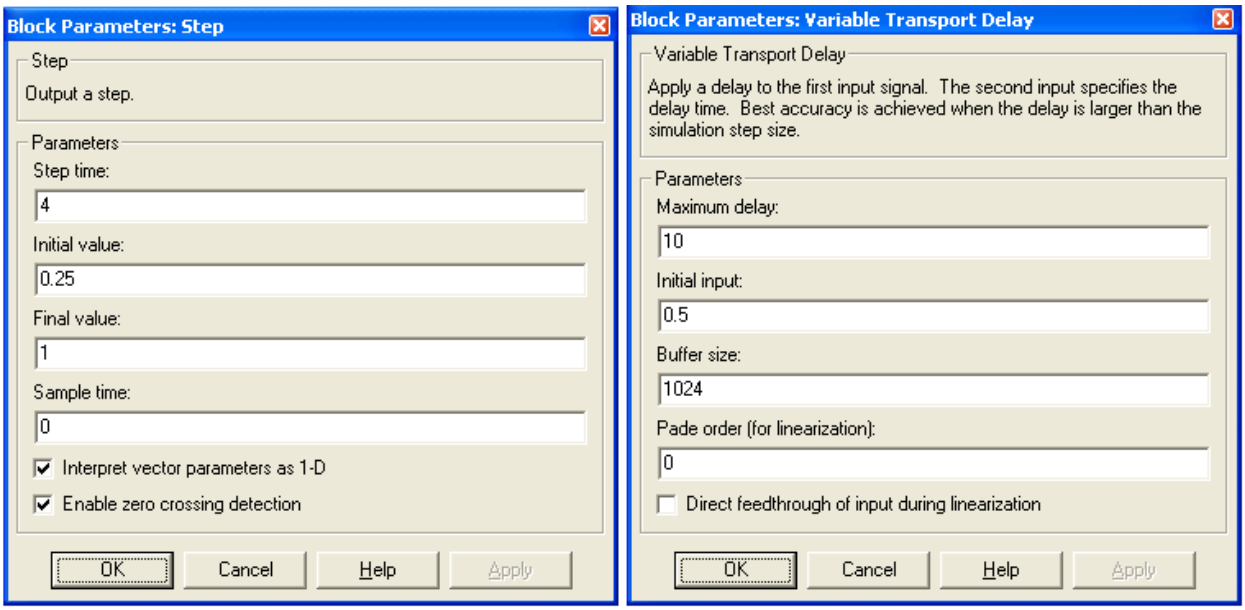


Рис. 15.24. Приклад використання блоку `Variable Transport Delay`



а *б*

Рис. 15.25. Параметри блоків `Step` (*а*) та `Variable Transport Delay` (*б*) для наведеного прикладу

15.10 Приклад побудови моделі лінійної безперервної системи

Розглянемо модель замкнутої лінійної системи керування, що складається з об'єкта, виконавчого механізму і регулятора (рис. 15.26).

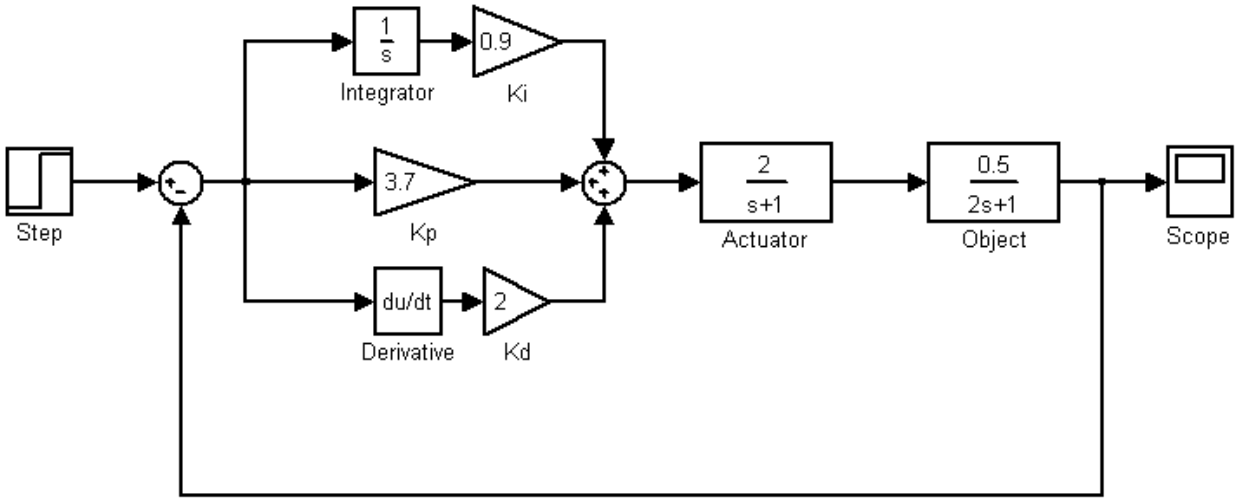


Рис. 15.26. Simulink-модель системи керування з ПІД-регулятором

Датчик вихідної величини будемо вважати звичайним підсилювачем з коефіцієнтом підсилення $K = 1$, тому вказувати його на схемі немає необхідності. У якості регулятору будемо використовувати ПІД – регулятор, що є найпоширенішим у промисловій практиці (до 90% від всіх регуляторів). Передаточна функція ПІД-регулятора:

$$W_{\text{ПІД}}(s) = K_p + \frac{K_i}{s} + K_d s, \qquad (15.14)$$

де коефіцієнти K_p, K_i та K_d підлягають визначенню при синтезі системи керування. Процес визначення значень коефіцієнтів ПІД-регулятора - це окрема тема та у даному посібнику не розглядається.

При побудові моделі ПІД-регулятора на рис. 15.26 використовувалися блоки `Integrator` (Інтегратор) і `Derivative` (Диференціювання) бібліотеки `Continuous`, а також блок `Gain` (Підсилювач) з бібліотеки `Math Operations`, за допомогою якого задаються коефіцієнти підсилення K_p, K_i та K_d пропорційного, інтегруючого і диференціюючого каналів відповідно.

На вхід системи подається одинична східчаста функція, що в Simulink моделюється блоком `Step` з бібліотеки `Sources`. Такий сигнал у теорії керування є типовим, і до нього часто прибігають при аналізі систем. До того ж східчастий вхідний сигнал часто зустрічається в житті, наприклад, при подачі напруги на двигун постійного струму. Як правило, моменту появи сигналу привласнюється значення $t = 0$, тому в даній моделі в текстовому полі **Step Time** (Час подачі східчастого сигналу) вікна параметрів блока `Step` установлене значення 0.

Для моделювання ПІД-регулятора Simulink має й окремий блок – `PID Controller` (рис. 15.27), що знаходитсяє в бібліотеці `Simulink Extras` → `Additional Linear`. Однак ми, для кращого розуміння особливостей роботи моделі, зупинимося на схемі, представленій на рис. 15.26.

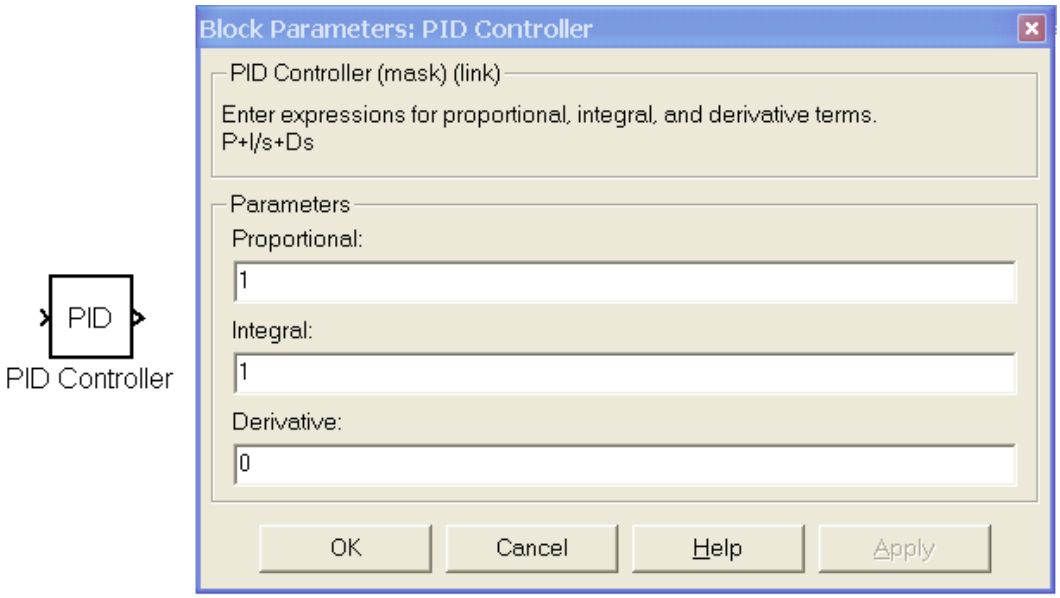


Рис. 15.27. Зображення і вікно параметрів блока `PID Controller`

На рис. 15.28 показана перехідна функція системи при значеннях коефіцієнтів регулятора $K_p = 3{,}7$, $K_i = 0{,}9$ і $K_d = 2$.

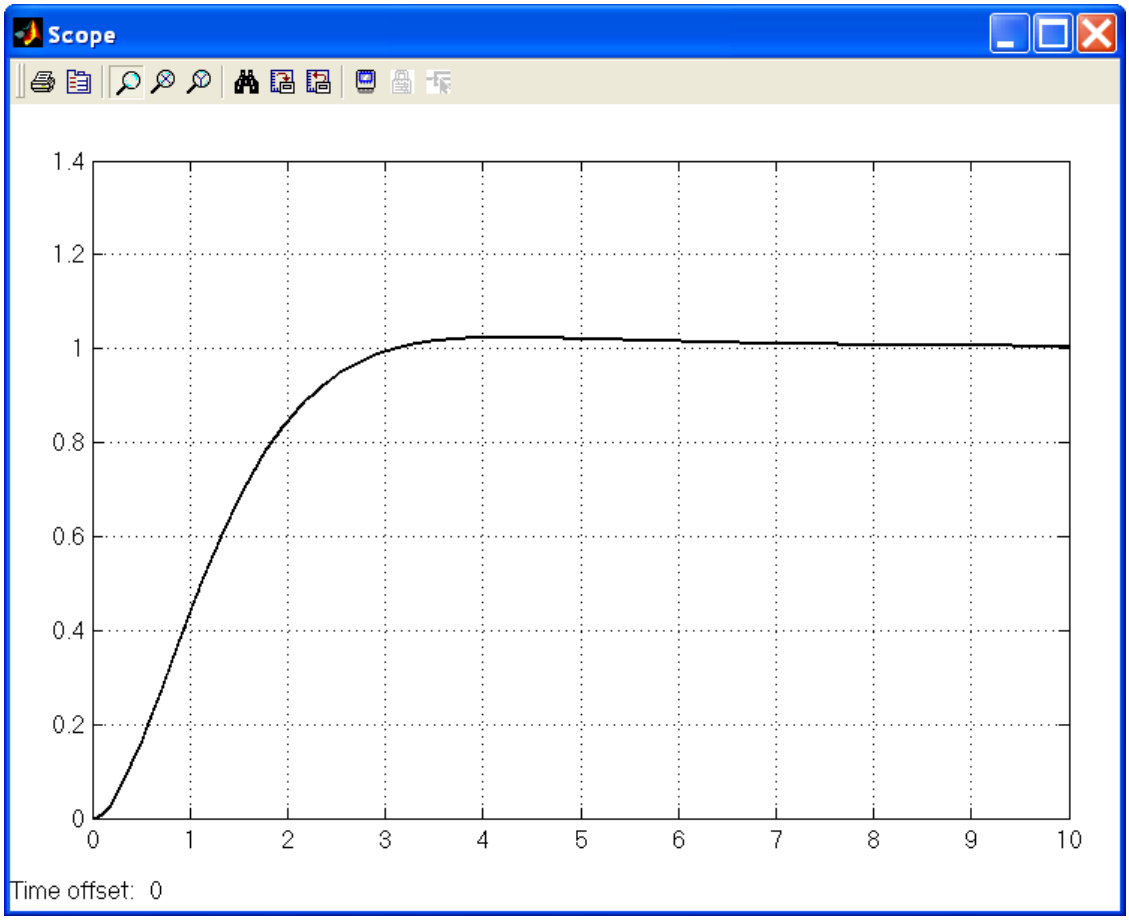
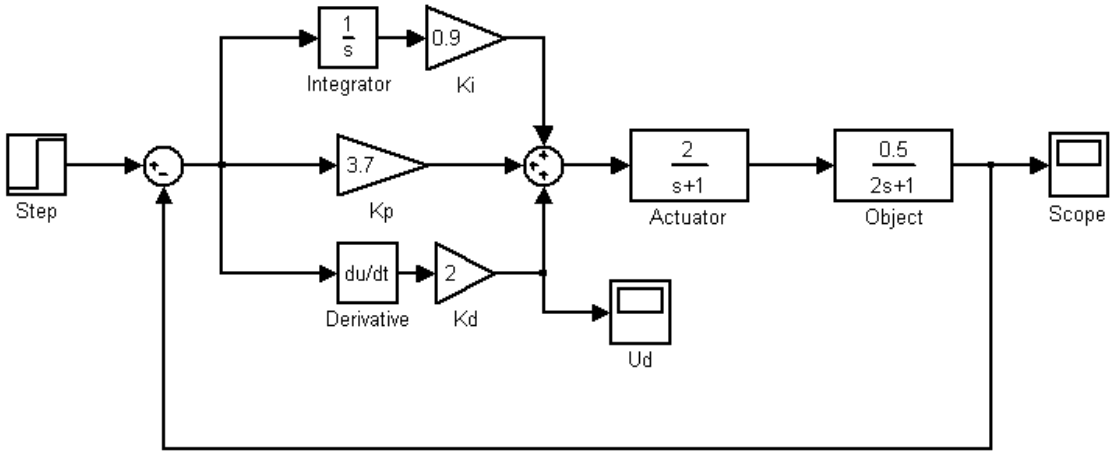
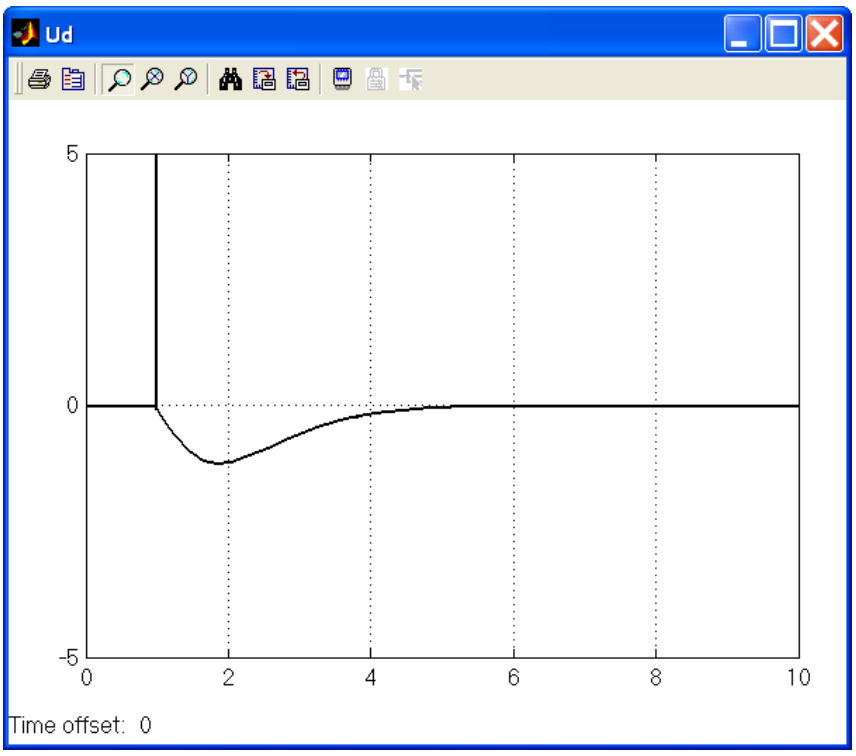


Рис. 15.28. Перехідна функція розглянутої системи керування

Зробимо одне істотне зауваження. Модель ПІД-регулятора, представлена на рис. 15.26 є не зовсім вдалою. Це пояснюється тим, що при східчастій зміні задаючого впливу, керування $U_д(0)$ на виході диференціюючого каналу, буде мати дуже велике значення, оскільки похідна східчастої функції дорівнює нескінченності. Цей факт ілюструє рис. 15.29.



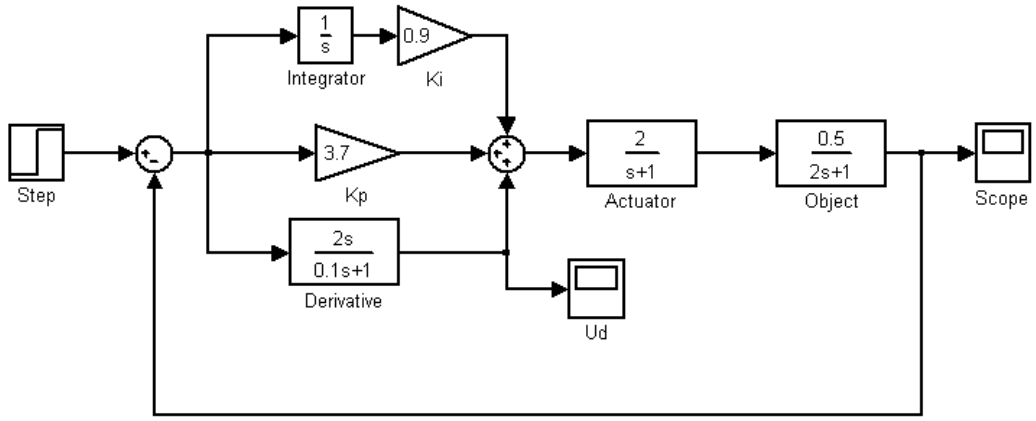
а



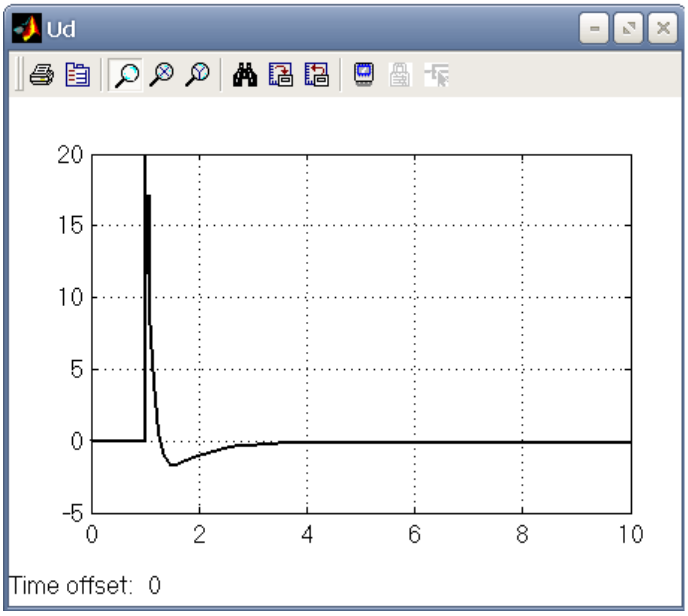
б

Рис. 15.29. Спостереження сигналу *Ud* диференціюючого каналу ПІД-регулятора: *а* – структурна схема; *б* – вид сигналу *Ud*. Добре видний стрибок вихідного сигналу в момент подачі вхідного (при t = 1 с)

Зрозумило, що у реальній системі через її інерційні властивості керування $U_д(0)$ хоча й може бути досить великим, але все ж таки буде обмежено за величиною. При моделюванні «важких», істотно інерційних систем така розбіжність між модельним і реальним сигналом $U_д(0)$ не помітна, однак у випадку «легкої» системи нескінченне значення $U_д(0)$ може стати причиною значної погрішності. У зв'язку із цим у диференціюючий канал моделі ПІД-регулятора рекомендується додавати аперіодичну ланку першого порядку з коефіцієнтом підсилення 1 і малою постійною часу (рис. 15.30).



а



б

Рис. 15.30. ПІД-регулятор з реальною диференціюючою ланкою:
а – структурна схема; б – вид сигналу *Ud*

Тоді на вхід виконавчого механізму буде надходити не такий великий вплив, як у системі на рис. 15.26, і він не буде підданий «удару» східчастою зміною на вході системи. Крім того, операція диференціювання дуже підсилює високочастотну похибку вимірювання вихідної величини, що може викликати «тряску» системи. Аперіодична ланка першого порядку є фільтром низьких частот, що обрізає цей високочастотний шум.

На рис. 15.30 у регуляторі використовується реальна диференціюючи ланка, що утворюється в результаті послідовного з'єднання ідеальної диференціюючої та аперіодичної ланок. Можна також використати блок `PID Controller (with Approximate Derivative)`, що розташований в бібліотеці `Simulink Extras → Additional Linear`. Зображення і вікно параметрів цього блока представлені на рис. 15.31. У полях **Proportional**, **Integral** і **Derivative** задаються коефіцієнти Kp, Ki та Kd, а в полі **Derivative divisor (N)** вказується постійна часу.

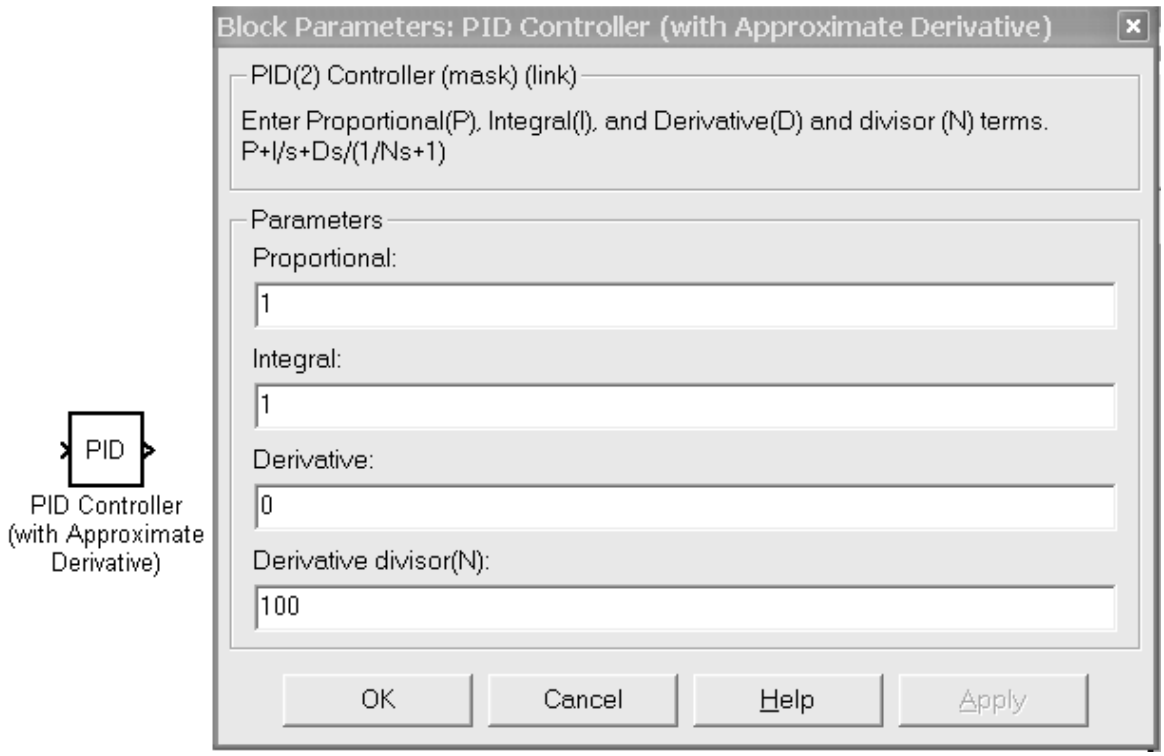


Рис. 15.31. Зображення і вікно параметрів блоку
`PID Controller (with Approximate Derivative)`

Завдання для самостійної роботи

1. Побудуйте модель, яка б дозволяла в одному вікні спостерігати перехідні функції трьох аперіодичних ланок першого порядку. Прийміть $k = 1$, $T_1 = 0{,}5$; $T_2 = 1$; $T_3 = 4$. Зробить висновок щодо впливу постійної часу T на тривалість перехідного процесу.

2. Динаміка системи описується коливальною ланкою з передаточної функцією

$$W(s) = \frac{1}{4s^2 + 4\xi s + 1}.$$

Побудуйте Simulink-модель, яка б дозволяла в одному вікні спостерігати вплив коефіцієнту демпфування ξ на від перехідної функції. Прийміть $\xi = 0{,}1$; $0{,}25$; $0{,}5$; 1. Для зручності прийміть час моделювання 20 с.

3. Система керування являє собою охоплену негативним зворотним зв'язком аперіодичну ланку першого порядку з постійною часу $T = 2$. Побудуйте модель, яка б дозволяла в одному графічному вікні спостерігати реакції системи на ступінчатий вплив при зміні коефіцієнту підсилення $k = 1$; 3; 4; 9. Для зміни значення k використайте блок `Slider Gain`. Час моделювання прийміть 0,5с.

4. Система керування має структуру, що являє собою інтегруючу ланку, охоплену негативним зворотним зв'язком.

а) Побудуйте реакцію системи на вплив $x(t) = t$ при $k = 1$.

б) В одному графічному вікні побудуйте графіки похибки $\varepsilon(t) = (x(t)\text{-}y(t))$ при $k = 0{,}5$; 1; 2; 5. Проаналізуйте вплив коефіцієнту підсилення на сталу похибку.

5. Система має структуру, що зображена на рис. 15.32. За допомогою Simulink дослідить вплив часу запізнювання τ на стійкість системи.

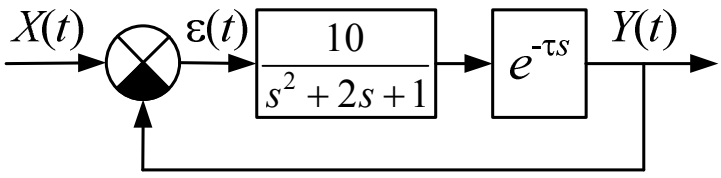


Рис. 15.32. Структура системи керування

16. АНАЛІЗ БАГАТОМІРНИХ СИСТЕМ

У даній главі ми розглянемо останній блок з бібліотеки `Continuous` пакета Simulink - блок `State-Space`, а також можливості Simulink для побудови, дослідження і синтезу багатомірних динамічних систем.

16.1 Блок `State-Space`

Блок `State-Space` являє собою модель динамічного об'єкта у векторно-матричній формі простору станів, які описуються рівняннями (11.3) – (11.4). Запишемо їх ще раз:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}; \\ \mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}, \end{cases}' \qquad (16.1)$$

де **x** – вектор стану, **u** – вектор вхідних впливів, **y** – вектор вихідних сигналів, **A** – матриця коефіцієнтів системи; **B** - матриця керування; **C** – матриця спостереження виходу; **D** – матриця зв'язку.

Вікно параметрів блока `State-Spase` має шість текстових полів (рис. 16.1). У перші чотири поля вводяться матриці **A**, **B**, **C** і **D**, вид яких визначає властивості об'єкта. Матриця **D** звичайно дорівнює нулю, оскільки в реальних системах у каналах між входами та виходами присутні динамічні ланки. Якщо матриця **D** відмінна від нульової матриці, це вказує на те, що, принаймні, один прямий шлях від входів до виходів не має динамічних ланок. Формат ввода матриць відповідає прийнятому в системі MATLAB: елементи кожного рядка відокремлюються один від одного комою або пробілом, а самі рядки матриці відокремлюються крапкою з комою. Так, наприклад, матрицю **A** розміром 2×2

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -5 & -2 \end{bmatrix}$$

в Simulink можна ввести в такий спосіб:

```
A=[0 1;-5 -2]
```

або

`A=[0,1;-5,-2].`

Нагадаємо, що в MATLAB існує й інший спосіб завдання матриць, однак для Simulink він не підходить.

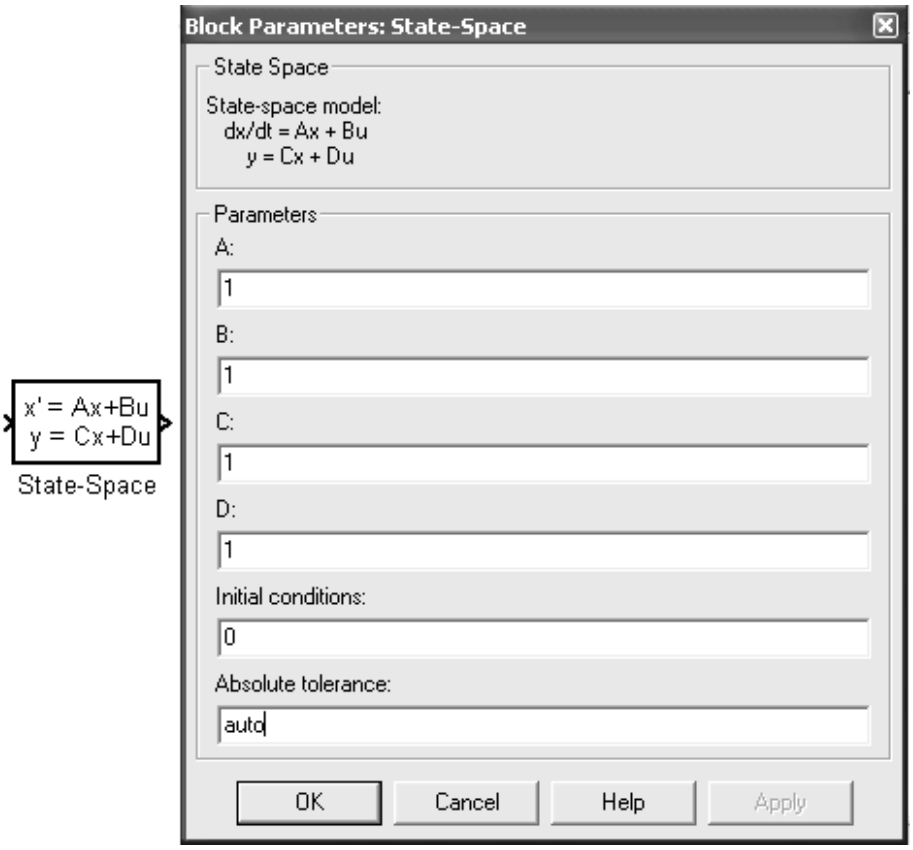


Рис. 16.1. Зображення і вікно параметрів блока `State-Space`

У полі **Initial condition** задається вектор початкових умов моделювання.

Параметр **Absolute tolerance** (Абсолютна погрішність) виконує ті ж функції, що й у розглянутих раніше блоках.

На рис. 16.2 показаний приклад моделювання динамічного об'єкта за допомогою блока `State-Space`. Матриці блока мають наступні значення:

$$\mathbf{A}=\begin{bmatrix}0 & 1\\ -5 & -2\end{bmatrix};\ \mathbf{B}=\begin{bmatrix}0\\ 3\end{bmatrix};\ \mathbf{C}=\begin{bmatrix}0 & 1\end{bmatrix};\ \mathbf{D}=\begin{bmatrix}0\end{bmatrix}.$$

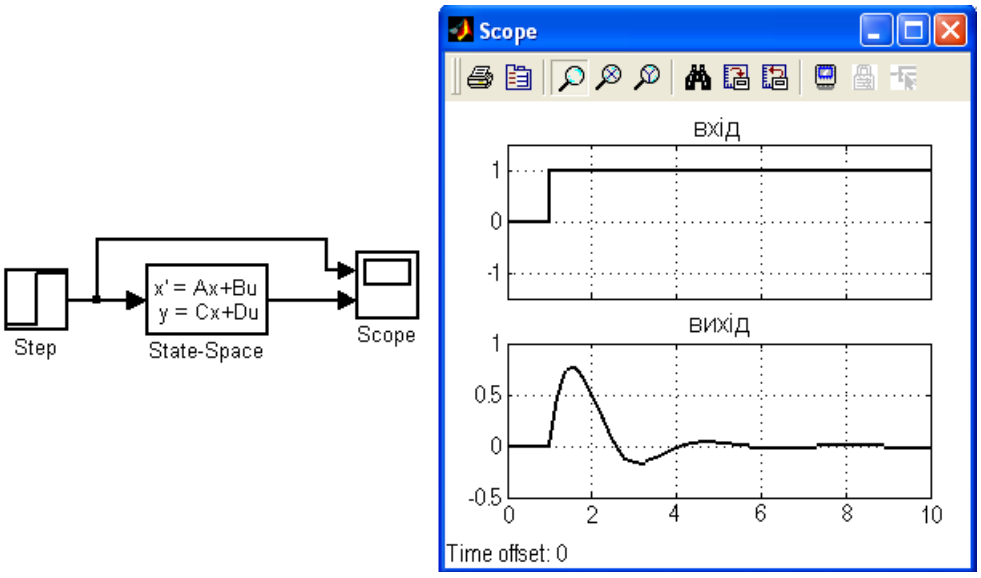


Рис. 16.2. Приклад використання блоку `State-Space`

Багато моделей фізичних систем мають декілька вхідних і декілька вихідних змінних. Такі системи називають багатомірними. Для моделювання багатомірних систем доцільно використовувати блок `State-Space`.

16.2 Приклад моделювання багатомірної системи блоком `State-Space`

Розглянемо модель із двома ступенями свободи, у якій дві маси зв'язані пружними та дисипативними зв'язками (рис. 16.3). Така модель спрощено описує вертикальні коливання підвіски автомобіля.

Рух розглянутої системи при наявності збурень з боку дороги описується наступною системою диференціальних рівнянь:

$$\begin{cases} m_1 \dfrac{d^2 z_1}{dt^2} + b_2\left(\dfrac{dz_1}{dt} - \dfrac{dz_2}{dt}\right) + c_1 z_1 + c_2\left(z_1 - z_2\right) = c_1 \eta; \\ m_2 \dfrac{d^2 z_2}{dt^2} + b_2\left(\dfrac{dz_2}{dt} - \dfrac{dz_1}{dt}\right) + c_2\left(z_2 - z_1\right) = 0. \end{cases} \qquad (16.2)$$

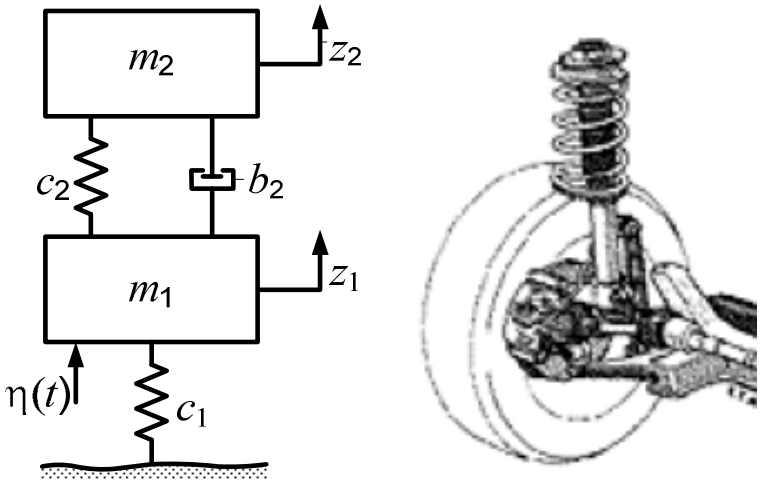


Рис. 16.3. Двомасова модель підвіски автомобіля
m_1 – маса колеса; m_2 – маса кузова, приведена до одного колеса;
c_1 – коефіцієнт пружності шини колеса; c_2, b_2 – коефіцієнти пружності та демпфірування підвіски; $\eta(t)$ - збурення з боку дороги;
z_1 – вертикальні коливання колеса; z_2 - вертикальні коливання кузова

Виберемо наступні координати стану: $x_1 = z_1$; $x_2 = z_2$; $x_3 = \dot{z}_1$; $x_4 = \dot{z}_2$. Вихідними змінними є x_1 і x_2. Тоді система рівнянь, що описує поведінку даної системи в просторі станів, має такий вигляд:

$$\begin{cases} \dot{x}_1 = x_3; \\ \dot{x}_2 = x_4; \\ \dot{x}_3 = -\dfrac{c_1 + c_2}{m_1} x_1 + \dfrac{c_2}{m_1} x_2 - \dfrac{b_2}{m_1} x_3 + \dfrac{b_2}{m_1} x_4 + \dfrac{c_1}{m_1} \eta; \\ \dot{x}_4 = \dfrac{c_2}{m_2} x_1 - \dfrac{c_2}{m_2} x_2 + \dfrac{b_2}{m_2} x_3 - \dfrac{b_2}{m_2} x_4. \end{cases} \qquad (16.3)$$

Звідси

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\dfrac{c_1 + c_2}{m_1} & \dfrac{c_2}{m_1} & -\dfrac{b_2}{m_1} & \dfrac{b_2}{m_1} \\ \dfrac{c_2}{m_2} & -\dfrac{c_2}{m_2} & \dfrac{b_2}{m_2} & -\dfrac{b_2}{m_2} \end{bmatrix}; \ \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ \dfrac{c_1}{m_1} \\ 0 \end{bmatrix};$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}; \ \mathbf{D} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Нерівність дороги добре описується наступною формулою

$$\eta(t) = \left| A_1 \sin \omega t + A_2 \sin 3\omega t \right|, \qquad (16.4)$$

де A_1 й A_2 - амплітуди нерівностей, що вибираються залежно від типу і стану дороги;

ω - частота збурення, $\omega = \dfrac{\pi}{L} V$;

L - довжина ділянки дороги, що розглядається;

V - швидкість руху автомобіля.

Побудуємо описану вище модель у середовищі Simulink (рис. 16.4).

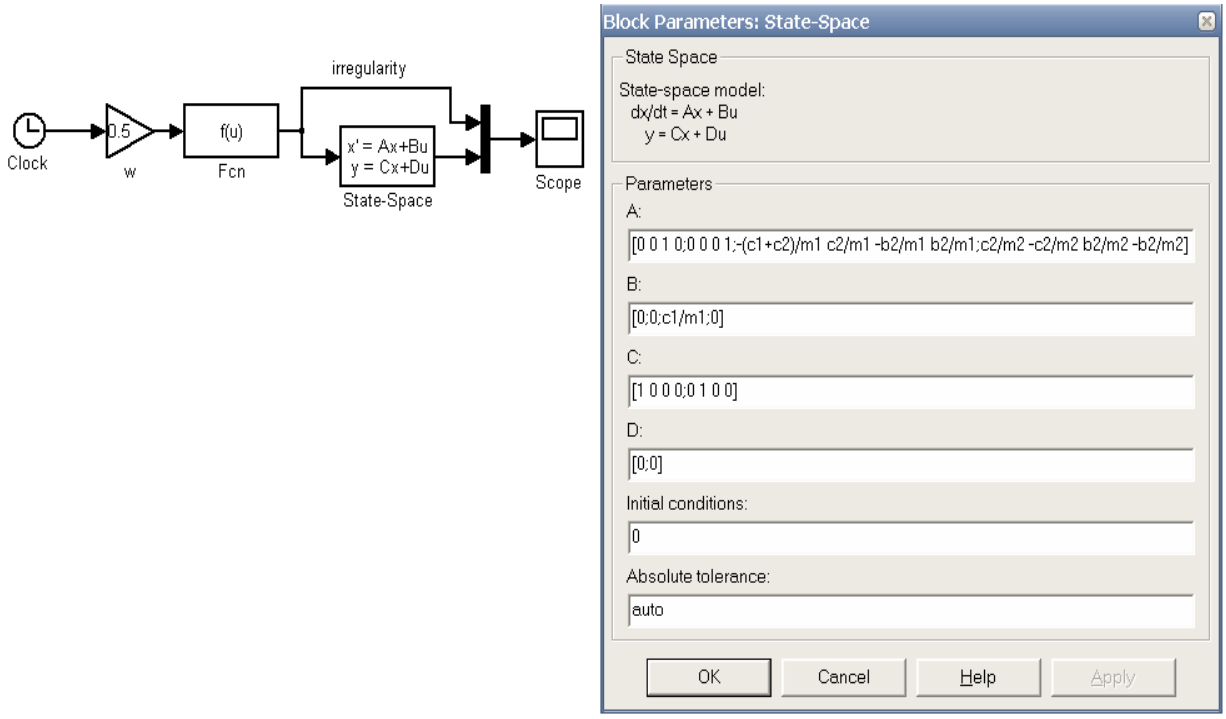


Рис. 16.4. Simulink-модель динаміки підвіски
і параметри блоку `State-Space`

Для моделювання будемо використовувати вхідні дані, наведені в табл. 16.1.

Таблиця 16.1

Вхідні дані для моделювання

Амплітуда першої гармоніки нерівності дороги	$A_1 = 0{,}5$	м
Амплітуда другої гармоніки нерівності дороги	$A_2 = 0{,}2$	м
Довжина ділянки дороги	$L = 25$	м
Маса колеса	$m_1 = 65$	кг
Приведена маса кузова	$m_2 = 1630$	кг
Коефіцієнт пружності шини колеса	$c_1 = 880$	кг/см
Коефіцієнт пружності підвіски	$c_2 = 131$	кг/см
Коефіцієнт демпфірування підвіски	$b_2 = 14{,}5$	кг с/см

Результати моделювання наведені на рис. 16.5.

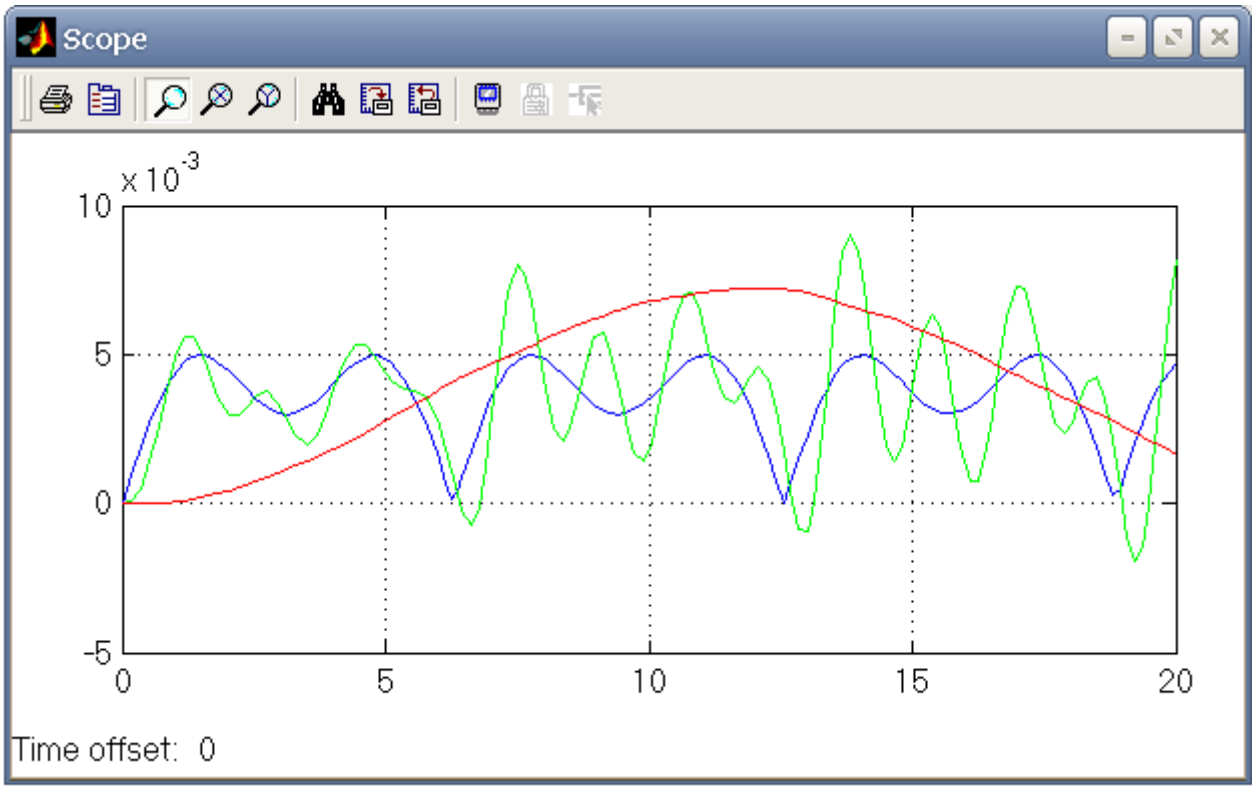


Рис. 16.5. Нерівність дороги та коливання підвіски автомобіля

16.3 Блок Demux

При розгляді багатомірних систем часто використовуються ще два блоки Mux і Demux бібліотеки Signal Routing. Із блоком Mux ми познайомилися у главі 14. Розглянемо тепер блок Demux.

Блок Demux (Демультіплексор) розділяє вектори вхідної змінної на скалярні вихідні змінні. Кожна з вихідних змінних відповідає компоненту вектора вхідної змінної.

Блок має наступні параметри (рис. 16.6). У текстовому полі **Number of outputs** (Число виходів) задається кількість виходів демультіплексора. Як і для блока Mux, тут можна також вводити імена сигналів, розділені комами.

У списку **Display option** (Опції зображення) пропонуються варіанти зображення блока:

none – усередині блока відображається слово Demux;

bar – зображення блока заливається суцільними кольорами (за замовчуванням чорним).

Встановлення прапорця напроти позиції **Bus selection mode** (Режим вибору шини) активізує режим розділення векторних сигналів.

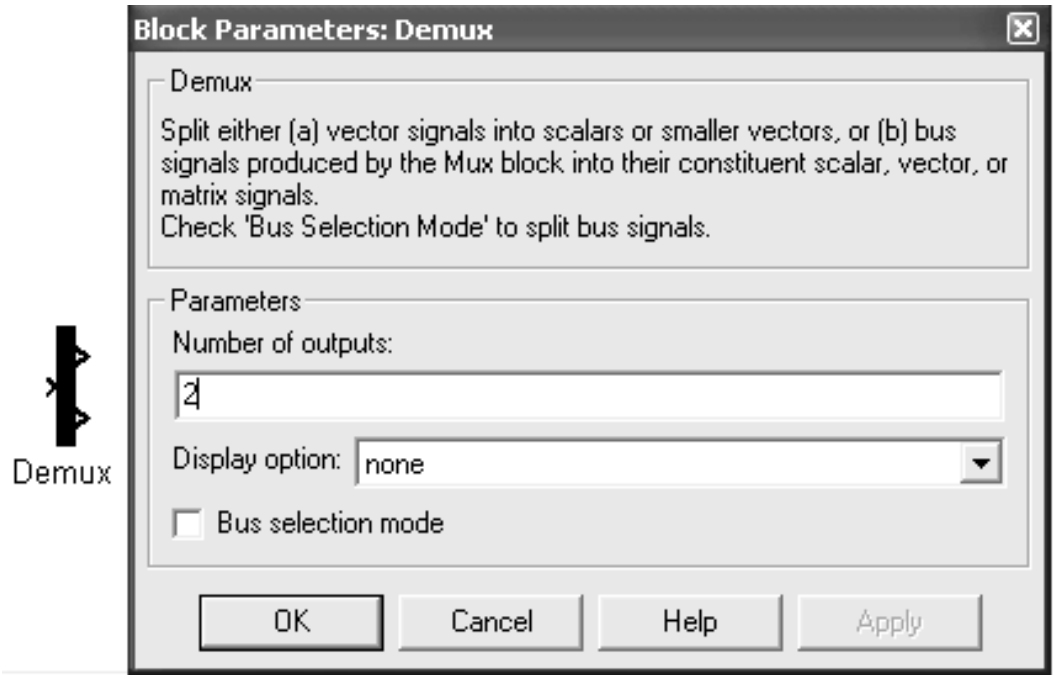
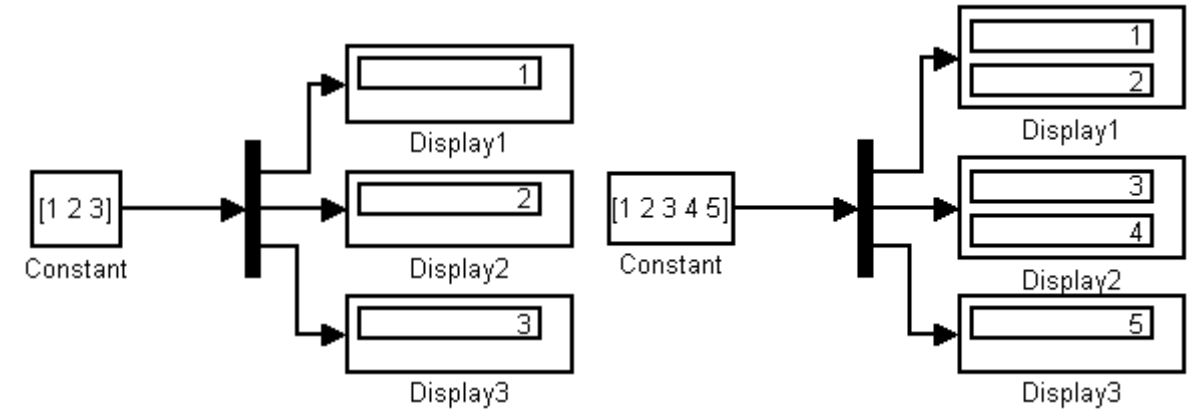


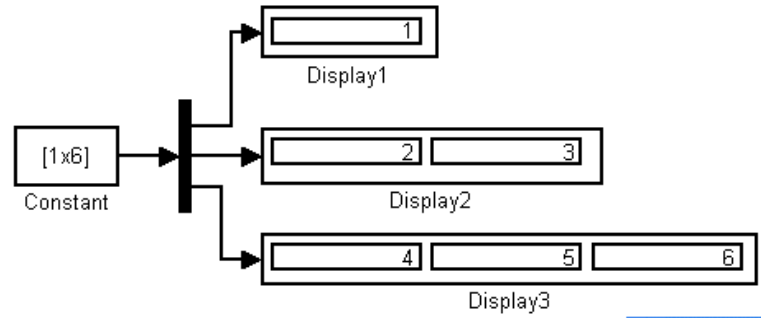
Рис. 16.6. Блок Demux і вікно його властивостей

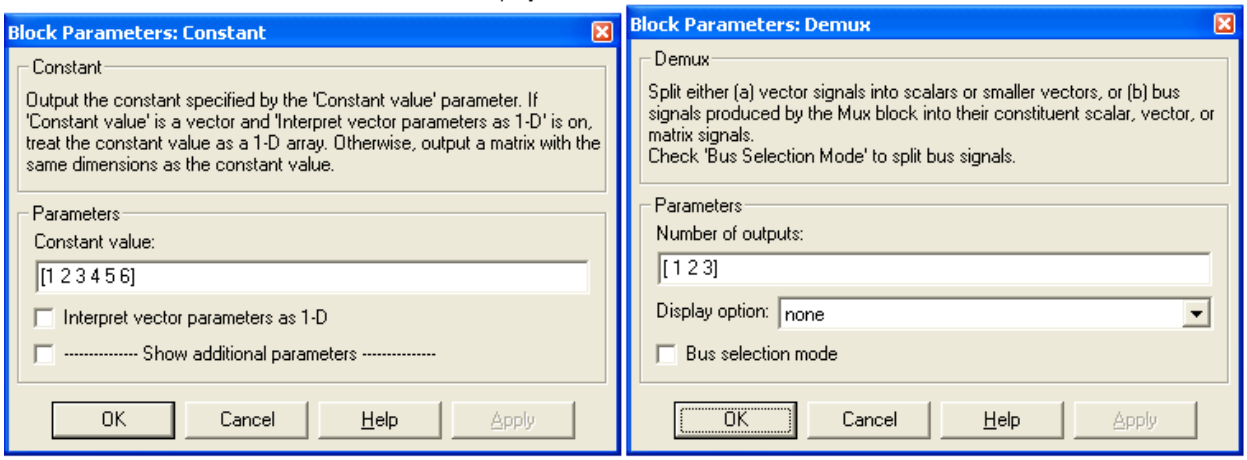
На вхід блоку `Demux` подається вектор, а вихідними сигналами можуть бути як скаляри, так і вектори, кількість та розмірність яких визначається параметром **Number of Outputs** та розмірністю вхідного вектора.

Якщо кількість виходів *p* дорівнює розмірності *n* вхідного вектора, то блок `Demux` здійснює розділення вхідного вектора на окремі елементи (рис. 16.7, а).



а *б*





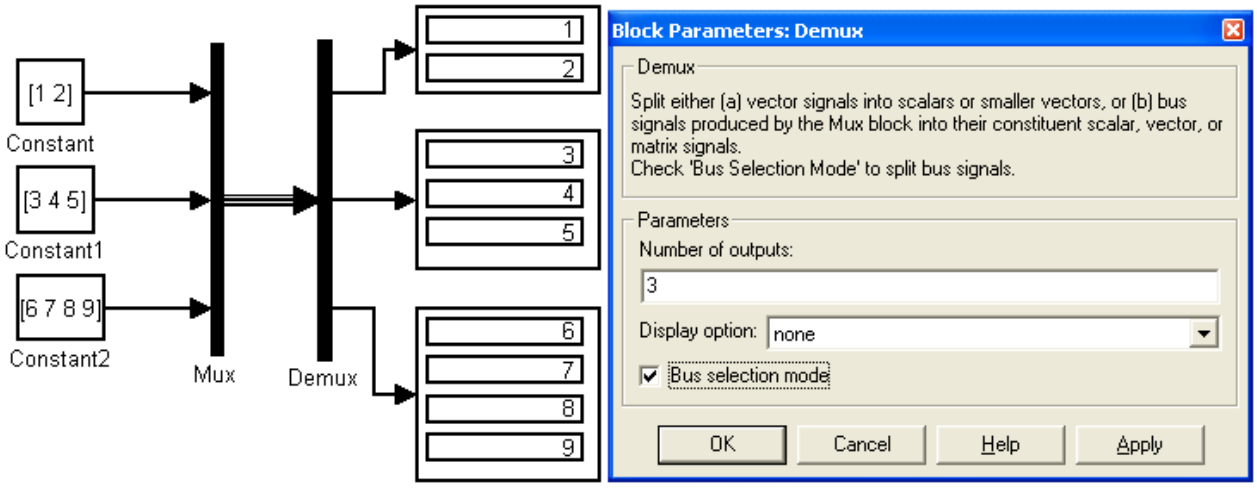
в

Рис. 16.7. Приклади використання блоку `Demux`

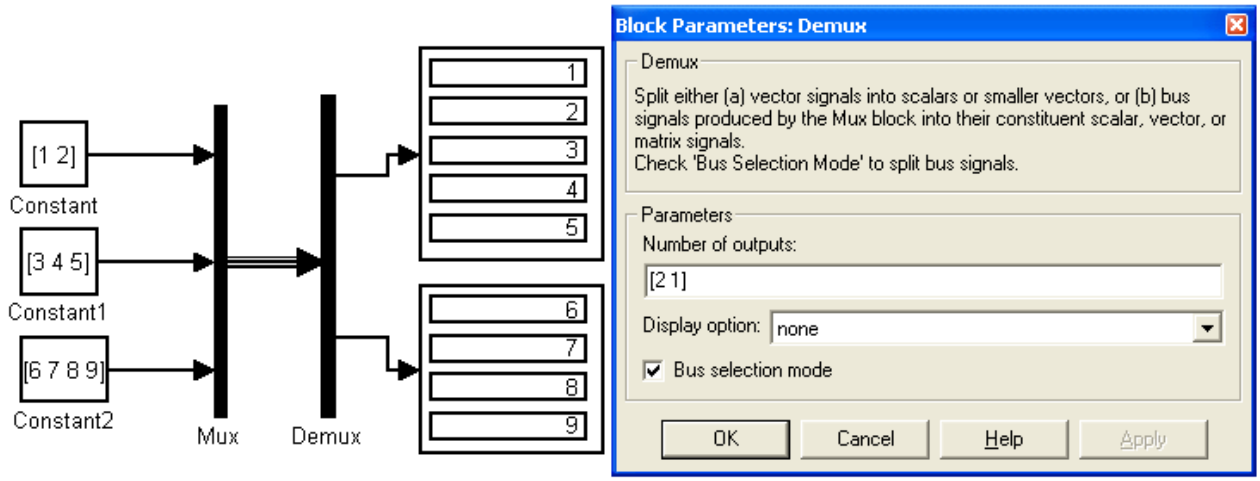
Якщо кількість виходів *p* менш, ніж розмірність *n* вхідного сигналу, то розмірність перших *p* - 1 вихідних сигналів дорівнює відношенню *n*/*p*, округленому до найближчого більшого числа, а розмірність останнього вихідного сигналу дорівнюватиме різниці між розмірністю вхідного сигналу та сумою розмірностей попередніх *p* - 1 виходів. Наприклад, якщо розмірність вхідного сигналу дорівнює 5, а кількість виходів дорівнює 3, то перші два вихідних вектора будуть мати розмірність ceil(5/3) = 2, а останній вихідний вектор буде мати розмірність 5 - (2+2) = 1 (рис. 16.7, *б*). Тут ceil – команда округлення до найближчого більшого цілого.

Параметр **Number of Outputs** можна також задати у вигляді вектора, що визначатиме розмірність кожного з вихідних сигналів. Наприклад, якщо у цьому полі ввести `[1 2 3]`, то блок `Demux` буде мати три виходи: перший - скаляр, другий - вектор з двох елементів, а третій - вектор з трьох елементів (рис. 16.7, *в*).

В режимі **Bus Selection Mode** блок `Demux` працює не з окремими елементами векторів, а з векторними сигналами. Вхідний сигнал в цьому режимі повинен бути сформований блоком `Mux` або іншим блоком `Demux`. Параметр **Number of Outputs** в цьому випадку задається або в вигляді скаляра, що визначає кількість вихідних сигналів (рис. 16.8, *а*), або в вигляді вектора, кожний елемент якого визначає кількість векторних сигналів в даному вихідному сигналі. Наприклад, якщо вхідним сигнал складається з трьох векторів, а в полі параметру **Number of Outputs** заданий вектор `[2 1]`, то на виході блоку `Demux` з’явиться два виходи, перший з яких буде містити два векторних сигнали, а другий – один (рис. 16.8, *б*). Зверніть увагу, що в режимі **Bus Selection Mode** зображення лінії зв’язку між блоками `Mux` і `Demux` змінюється на зображення шині.



а



б

Рис. 16.8. Приклади використання блоку `Demux` в режимі **Bus Selection Mode**

16.4 Зображення ліній зв'язку для векторних і скалярних змінних

Для підвищення наочності та спрощення налагодження моделі Simulink має можливість відображати імена сигналів, що проходять по лініях зв’язку.

Розглянемо приклад моделі, що має дві скалярні вхідні змінні - числові константи *a* і *b*, які в той же час є компонентами вектора вихідної змінної блока `Mux`. Блок `Demux` розгалужує сигнал і формує дві скалярні вихідні змінні (рис. 16.9).

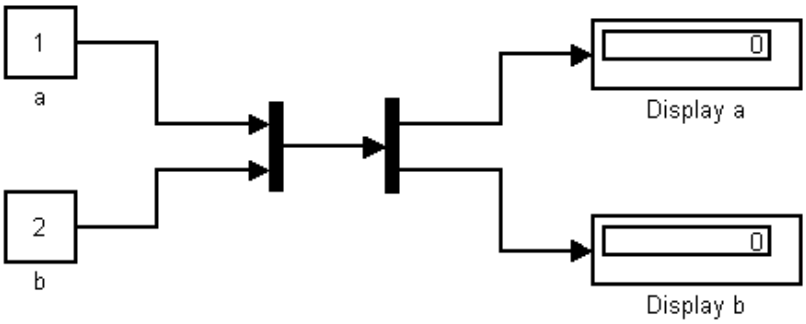


Рис. 16.9. Формування зі скалярних сигналів вектора
і розгалуження вектора

Спочатку на структурній схемі наносяться позначення ліній зв'язку, що з'єднують блоки `Constant` (з бібліотеки `Sources`) і `Mux` у вигляді символів *a* і *b*. Для цього на потрібній лінії клацаємо правою кнопкою миші та з контекстного меню вибираємо меню **Signal properties** (Властивості сигналу). У вікні, що з'являється, у **полі Signal name** (Ім'я сигналу) задаємо ім'я сигналу: *a* – для верхньої лінії (рис. 16.10), та *b* – для нижньої. Після підтвердження імені натискання кнопки OK або Apply, необхідно запустити модель, й тоді біля відповідних ліній будуть відображатися їхні імена.

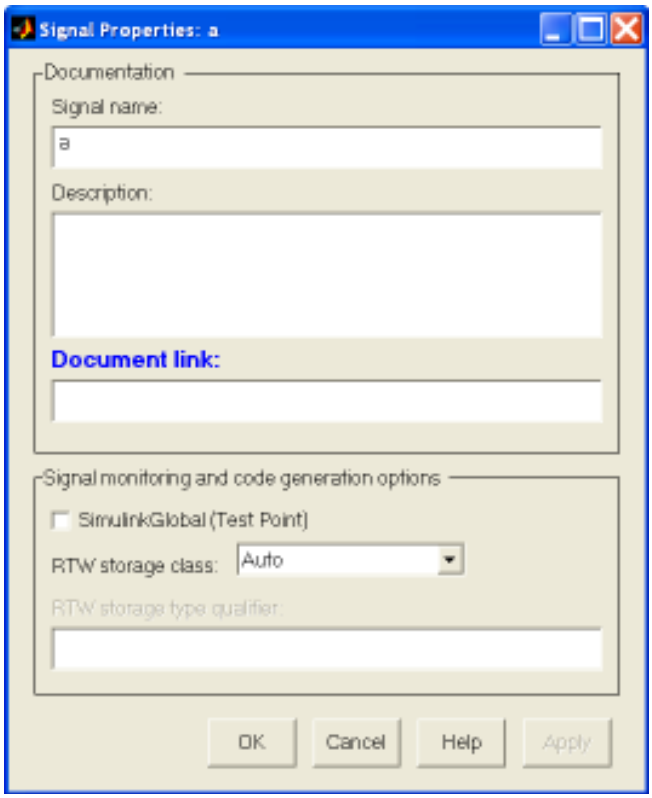


Рис. 16.10. Вікно властивостей вхідного скалярного сигналу

З вихідного порту блока `Mux` передається векторна змінна, що включає два скалярних елементи *a* і *b*. Щоб відобразити сигнали, що проходять по тій лінії зв’язку Щоб відобразити сигнали, що проходять по цій лінії зв'язку, викличемо вікно її властивостей та зі списку **Show propagated signals** (Показати сигнали, що передаються) виберемо функцію **on** (рис. 16.11). При цьому елементи сигналу будуть відображатися в дужках < > і відокремлюватися комами.

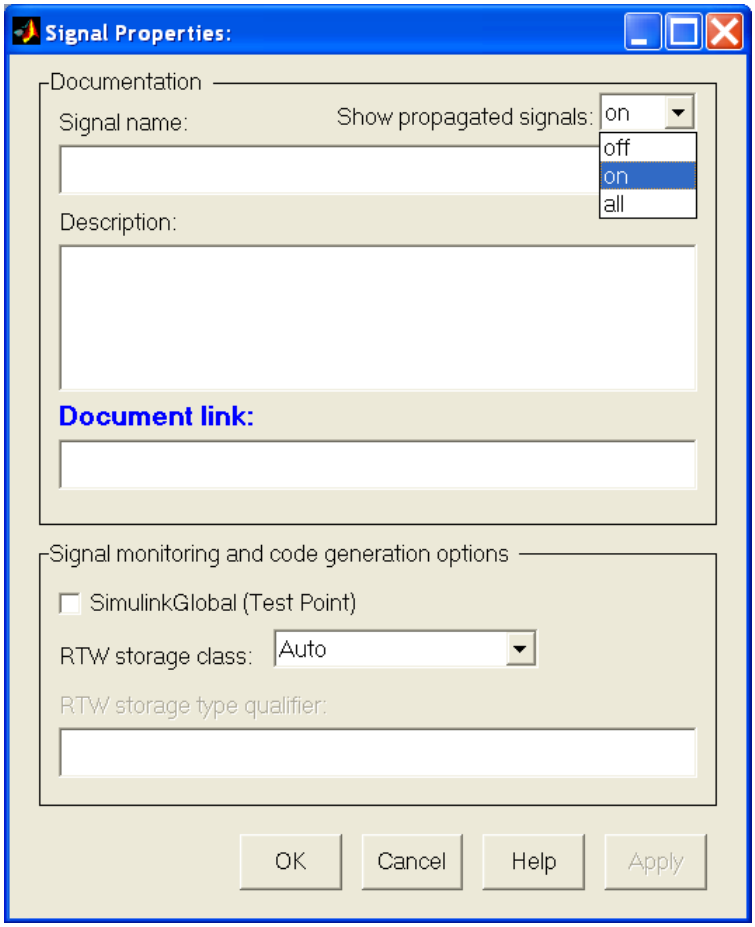


Рис. 16.11. Вікно властивостей вихідного сигналу блока `Mux`

Таким же чином позначаються лінії зв'язку, що йдуть від вихідних портів блока `Demux`. Схема моделі з усіма позначеними лініями зв'язку зображена на рис. 16.12.

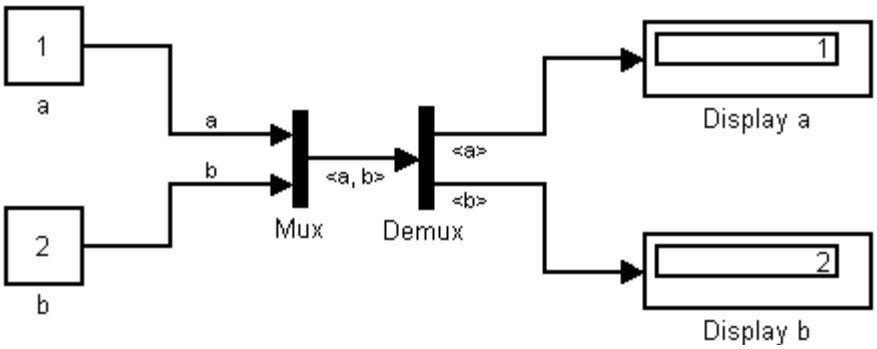


Рис. 16.12. Модель із усіма позначеними лініями зв'язку

У разі, коли сигнали, що передаються, є векторами, порядок дій для маркування ліній зв’язку такий самий. Необхідне, лише, не забути встановити прапорець напроти позиції **Bus Selection Mode** у вікні властивостей блоку `Demux`. На рис. 16.13 наведена модель з позначеними лініями зв’язку, по яким передаються векторні сигнали **A** =[1 2 3 4] та **B** =[5 6 7 8 9].

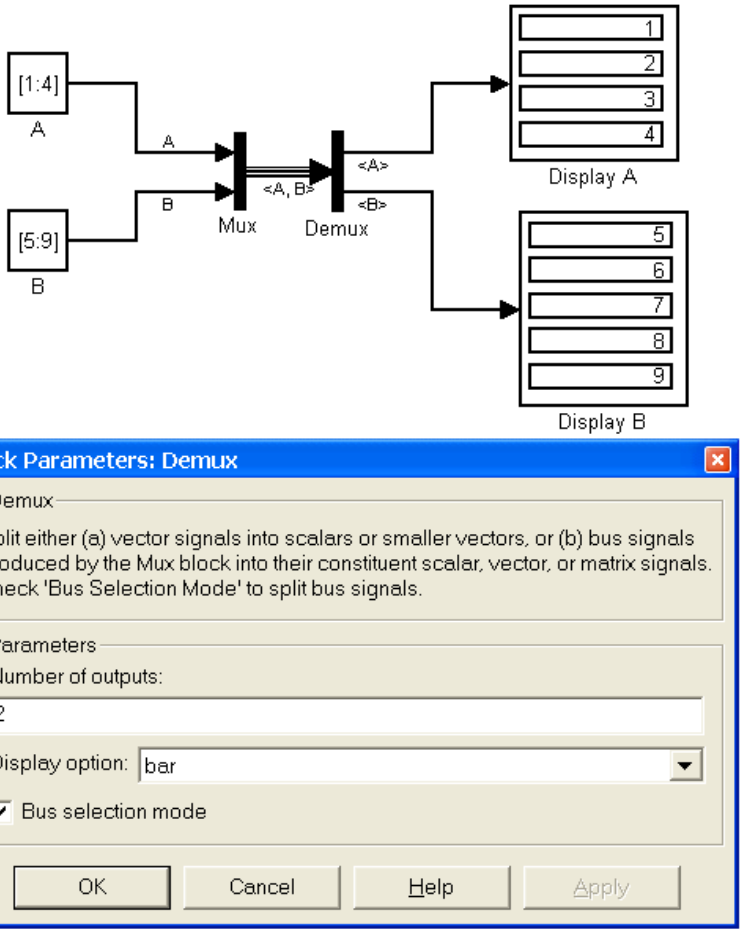


Рис. 16.13. Модель із векторними сигналами та усіма позначеними лініями зв'язку

Завдання для самостійної роботи

1. Побудуйте модель, яка б дозволяла в одному вікні спостерігати синусоїдальний сигнал, інтеграл від синусоїдального сигналу та його похідну (блок `Derivative` з бібліотеки `Continuous`).

2. На рис. 16.14 надана механічна система з трьома пружно пов'язаними візками. На перший і третій візки діють зовнішні сили F_1 та F_2.

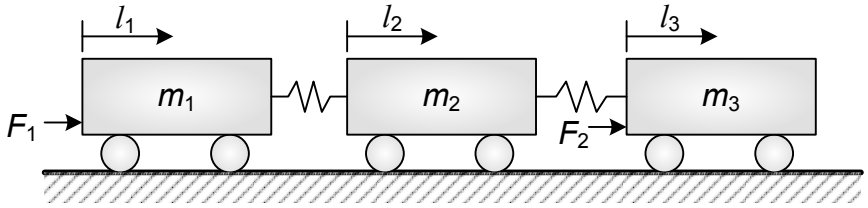


Рис. 16.14. Механічна система з трьома пружно пов'язаними візками

а) Запишіть диференціальні рівняння, що описують динаміку даної системи, якщо вихідними змінними є лінійні переміщення l_1, l_2 та l_3 кожного візка.

б) Побудуйте модель системи в просторі станів. У якості змінних стану прийміть переміщення та швидкості візків.

в) За допомогою Simulink дослідіть реакцію системи на вхідні сигнали $F_1 = 0$ та $F_2 = 1(t)$ при нульових початкових умовах.

г) Дослідіть реакцію системи на вхідні сигнали $F_1 = 0$ та $F_2 = \delta(t)$ при нульових початкових умовах.

Вказівка. Одиничний імпульс $\delta(t)$ можна апроксимувати вираженням $\delta(t) \approx 100(t) - 100(t - 0{,}01)$. Сформувати такий сигнал можна за допомогою двох блоків східчастих сигналів і суматора (рис. 16.15). Параметри **Step Time**, **Initial value** та **Final value** блоку `Step Start pulse` мають значення 0, 0 та 100 відповідно, а ті ж самі параметри блоку `Step End pulse` 0, 0,01 та 100.

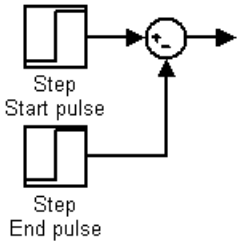


Рис. 16.15. Формування одиничного імпульсу

3. У моделі, що була побудована при виконанні завдання 2, *г* виконайте наступне.

а) Побудуйте графіки переміщень l_1, l_2 та l_3 кожного візка в окремих графічних вікнах.

б) Дайте позначення лініям зв'язку в моделі.

17. МОДЕЛЮВАННЯ НЕЛІНІЙНИХ СИСТЕМ

Більшість реальних систем автоматичного керування описуються нелінійними рівняннями. Їхня лінеаризація і розгляд як лінійних спрощує розрахунки, але робить їх наближеними. Іноді для досягнення необхідної точності не можна зневажати нелінійними властивостями системи. У цих випадках у структурні схеми поряд з лінійними моделями включають моделі різних нелінійних елементів, що дозволяє підвищити точність моделювання реальних систем.

Пакет Simulink має ряд блоків, що моделюють типові нелінійності. Ці блоки розташовані в розділі `Discontinuities` бібліотеки Simulink. Як прийнято в теорії керування, більшість нелінійних блоків розглядаються як ідеальні, без урахування інерційних властивостей. Для урахування інерційності необхідно використати блоки з розділу `Continuous`.

17.1 Нелінійні блоки бібліотеки `Discontinuities`

Розглянемо блоки бібліотеки `Discontinuities` (рис. 17.1), що моделюють основні типові нелінійні ланки.

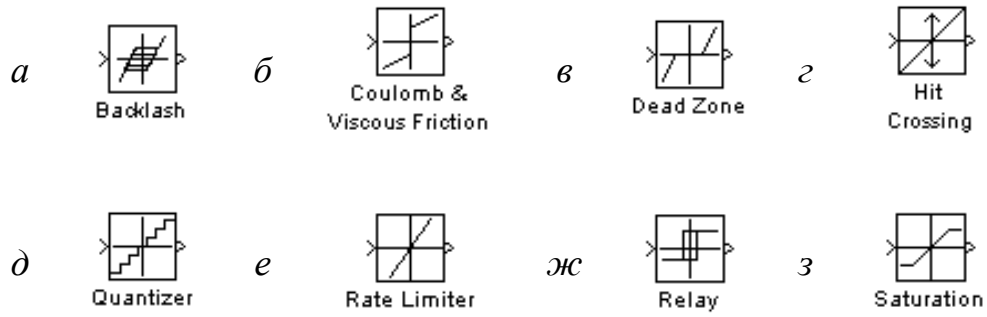


Рис. 17.1. Нелінійні елементи бібліотеки `Discontinuities`

Властивості нелінійностей найкраще досліджувати, подаючи на їхній вхід синусоїдальний сигнал. Для цього будемо збирати моделі відповідно до схеми, представленої на рис. 17.2.

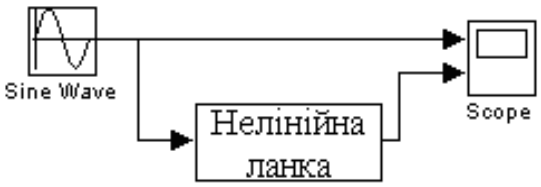
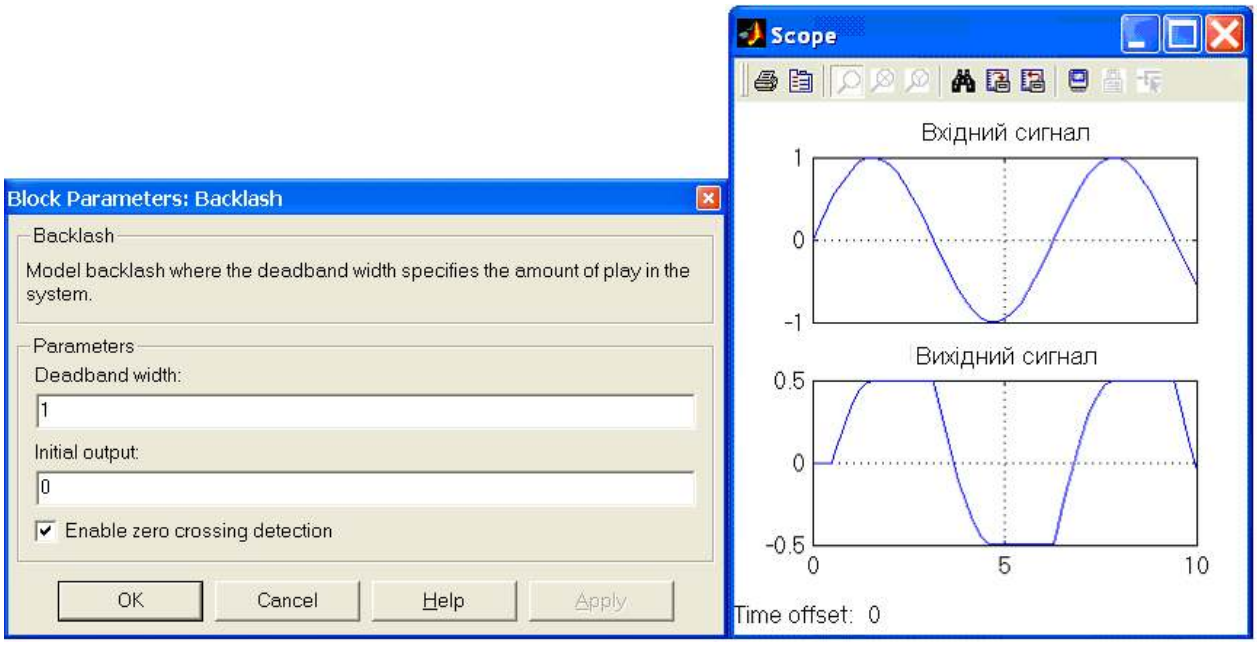


Рис. 17.2. Схема для дослідження властивостей нелінійної ланки

17.2 Блок `Backlash`

`Backlash` або люфт (рис. 17.1, *а*) - одна з нелінійностей, що найбільш часто зустрічається в системах і пов'язана з наявністю зазорів між різними механічними компонентами. На рис. 17.3, представлені діалогове вікно параметрів нелінійного блока `Backlash` (люфт) і його реакція на синусоїдальний сигнал.



а *б*

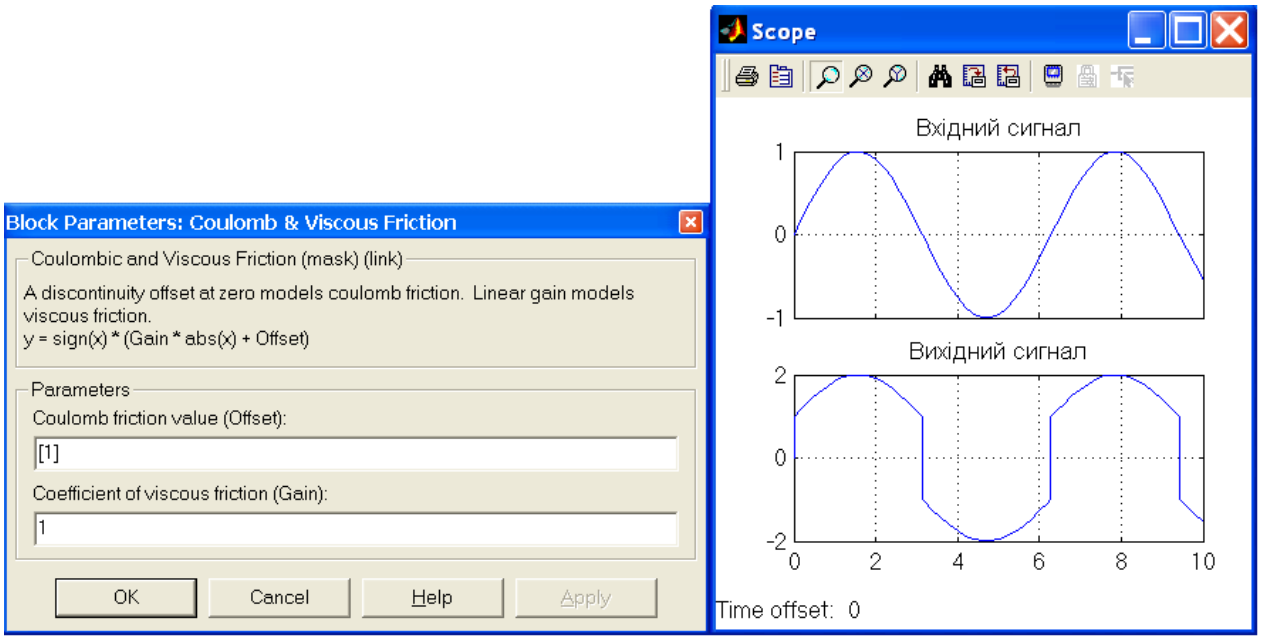
Рис. 17.3. Діалогове вікно параметрів блока `Backlash` (*а*) і його реакція на синусоїдальний сигнал (*б*)

Діалогове вікно містить два поля введення: **Deadband width** (Ширина смуги люфту) і **Initial output** (Початкове значення вихідного сигналу). Сигнал на виході блока буде дорівнювати значенню, заданому в поле **Initial output**, поки вхідний сигнал при

зростанні не досягне половини значення, зазначеного в полі **Deaband width**, після чого вихідний сигнал буде дорівнювати *u*-*b*/2, де *u* – вхідний сигнал, *b* - ширина смуги люфту. Після того як відбудеться зміна напрямку зміни вхідного сигналу, вихідний сигнал буде залишатися незмінним доти, поки вхідний сигнал не зміниться на величину *b*/2, після чого вихідний сигнал буде дорівнювати *u*+*b*/2.

17.3 Блок `Coulomb&Viscous Friction`

Блок `Columbic&Viscous Friction` (кулоновске і в'язке тертя) служить для моделювання фрикційних ефектів, тобто враховує нелінійні ефекти, які виникають при терті між різними механічними елементами (рис. 17.1, *б*). На рис. 17.4, *а* представлене вікно параметрів цього блока.



а *б*

Рис. 17.4. Вікно параметрів (*а*) і реакція на вхідний сигнал (*б*) блока `Coulomb&Viscous Friction`

У вікні параметрів зазначене рівняння, відповідно до якого блок `Coulomb&Viscous Friction` формує вихідний сигнал:

$$y = \text{sign}(x) \cdot \left(\text{Gain} \cdot \text{abs}(x) + \text{Offset}\right), \qquad (17.1)$$

де *x* - вхідний сигнал,

y - вихідний сигнал,

Gain - коефіцієнт в'язкого тертя,

Offset - величина сухого тертя.

Крім того, вікно параметрів має два текстові поля:

- **Coulomb friction value (Offset)** – Величина сухого тертя;

- **Coefficient of viscous friction (Gain)** – Коефіцієнт в'язкого тертя.

На рис. 17.4, *б* показаний приклад використання блока `Coulomb&Viscous Friction`. Обидва параметри блока задані рівними 1.

17.4 Блок `Dead Zone`

`Dead Zone` – мертва зона (або зона нечутливості) - одна з нелінійностей, що являє собою лінійну залежність вихідного сигналу від вхідного скрізь, за винятком області мертвої зони (рис. 17.1, *в*). Подібною характеристикою володіють багато реальних елементів в області малих вхідних сигналів. Вікно параметрів блока `Dead Zone` показане на рис. 17.5, *а*.

Вікно параметрів містить два поля введення:

- **Start of dead zone** - Початок мертвої зони;

- **End of dead zone** - Кінець мертвої зони;

і три параметри - прапорця:

- **Saturate on integer overflow** - Придушувати переповнення цілого. При встановленому прапорці обмеження сигналів цілого типу виконується коректно;

- **Treat as gain when linearizing** - Трактувати як підсилювач із коефіцієнтом передачі рівним 1 при лінеаризації;

- **Enable zero crossing detection** - Дозволити визначення перетинання нуля.

Вихідний сигнал блока (рис. 17.5, *б)* визначається наступним рівнянням:

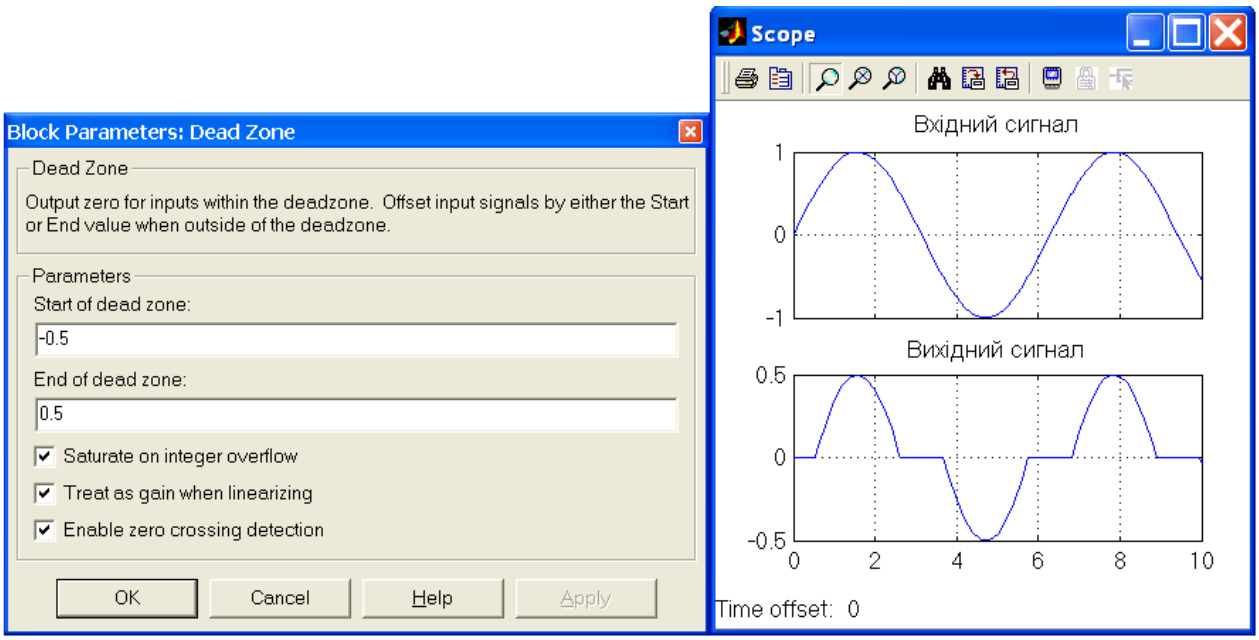
$$y = \begin{cases} 0, & \text{якщо } b_1 < u < b_2; \\ u - b_1, & \text{якщо } u \le b_1; \\ u - b_2, & \text{якщо } u \ge b_2, \end{cases} \qquad (17.2)$$

де u - вхідний сигнал блока;

y - вихідний сигнал блока;

b_1 - початок мертвої зони;

b_2 – кінець мертвої зони.



а *б*

Рис. 17.5. Діалогове вікно параметрів елемента `Dead Zone` і його вихідний сигнал

17.5 Блок `Hit Crossing`

Цей блок здійснює порівняння вхідного сигналу із заданим граничним значенням (за замовчуванням 0) і видає в цей момент одиничний імпульс.

Діалогове вікно параметрів блока (рис. 17.6) містить текстове поле **Hit crossing offset**; у ньому встановлюється поріг, перетинання якого вхідним сигналом потрібно визначити.

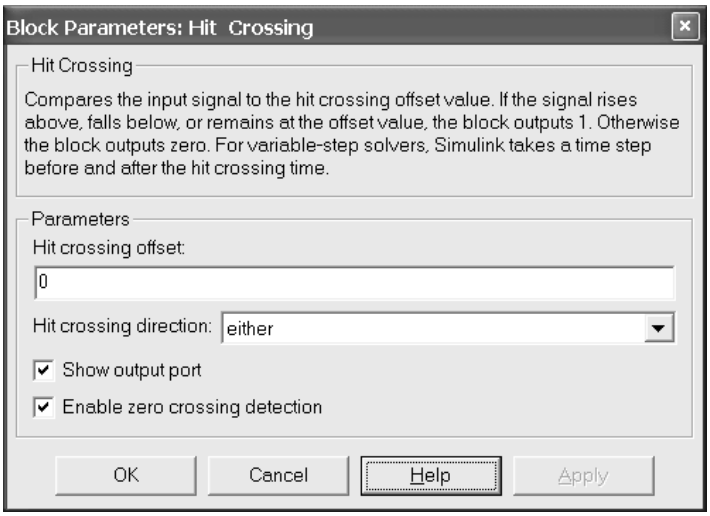


Рис. 17.6. Діалогове вікно параметрів блока `Hit Crossing`

Наступним параметром є список, що розкривається **Hit crossing direction** (Напрямок перетинання).

Тут вказується, у якому випадку фіксувати перетинання порога, при цьому піктограма блока змінює свій вид:

- **rising** – при збільшенні вхідного сигналу (рис. 17.7);
- **falling** - при зменшенні вхідного сигналу (рис. 17.8);
- **either** – якщо керуючий сигнал перетинає заданий поріг як зверху, так і знизу (рис. 17.9).

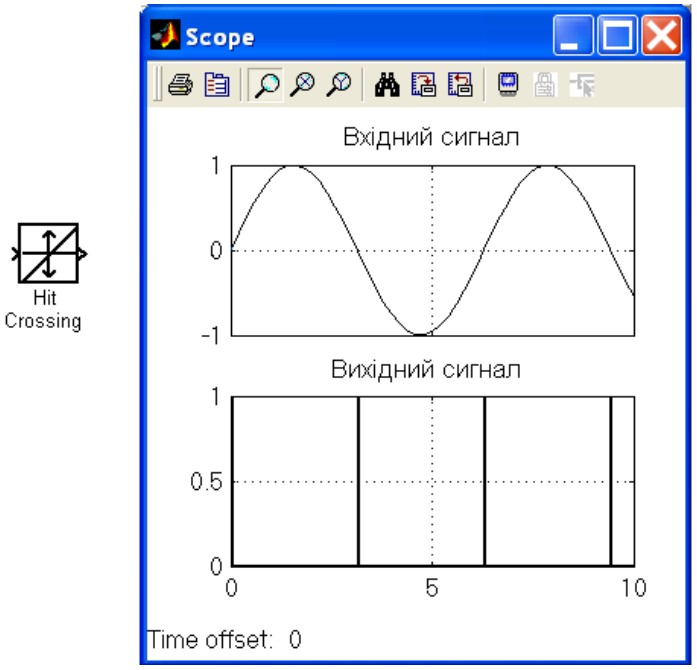


Рис. 17.7. Зовнішній вигляд і вихідний сигнал блока `Hit Crossing` при виборі значення **either** параметра **Hit crossing offset**

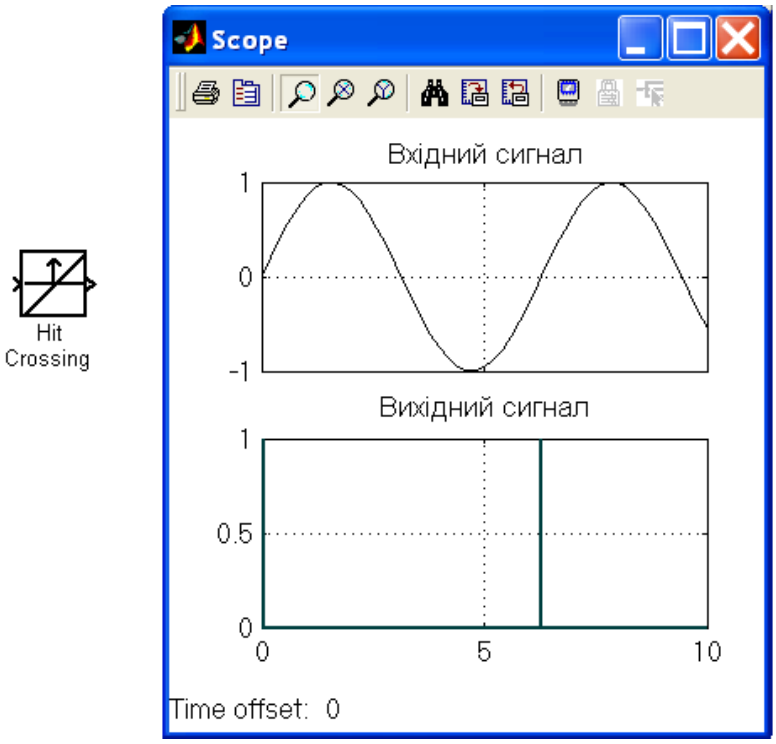


Рис. 17.8. Зовнішній вигляд і вихідний сигнал блока `Hit Crossing` при виборі значення **falling** параметра **Hit crossing offset**

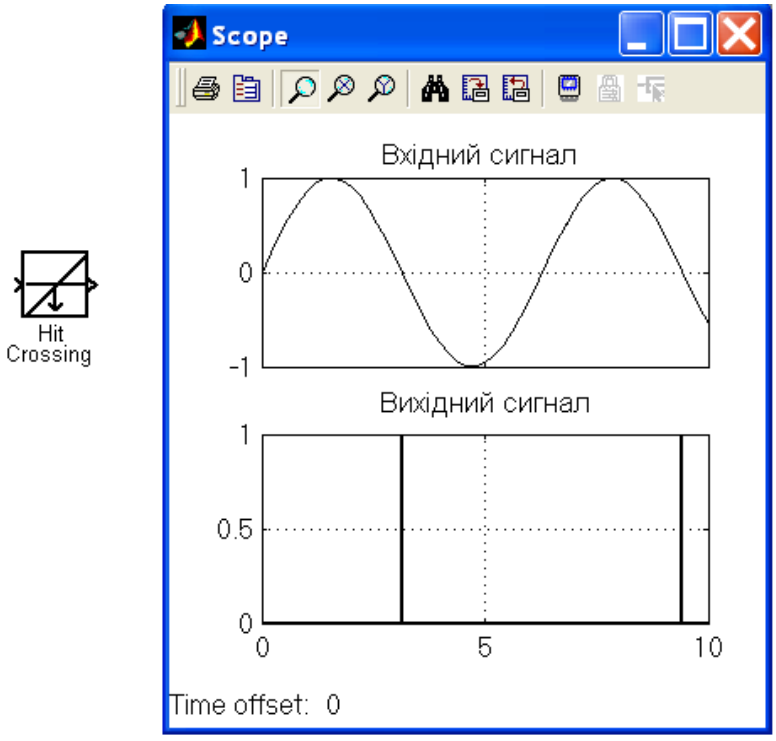


Рис. 17.9. Зовнішній вигляд і вихідний сигнал блока `Hit Crossing` при виборі значення **rising** параметра **Hit crossing offset**

Крім того, є ще два параметри, які визначаються встановленням прапорця:

- **Show output port** – Показати вихідний порт. У тому випадку, якщо цей прапорець зняти, точка перетинання сигналом граничного рівня визначається, але вихідний сигнал блоком не генерується;

- **Enable zero crossing detection** - Дозволити визначення перетинання нуля.

17.6 Блок `Quantizer`

Блок `Quantizer` (Квантувач) перетворює безперервну величину в дискретну із квантуванням за рівнем (рис. 17.1, *д)*. Таку характеристику мають, наприклад, аналогово-цифрові перетворювачі. На рис. 17.10 представлене діалогове вікно параметрів блока `Quantizer`.

Діалогове вікно містить поле введення **Quantization interval** (Шаг квантування) і один прапорець: **Treat as gain when linearizing** (Розглядати як підсилювач при лінеаризації).

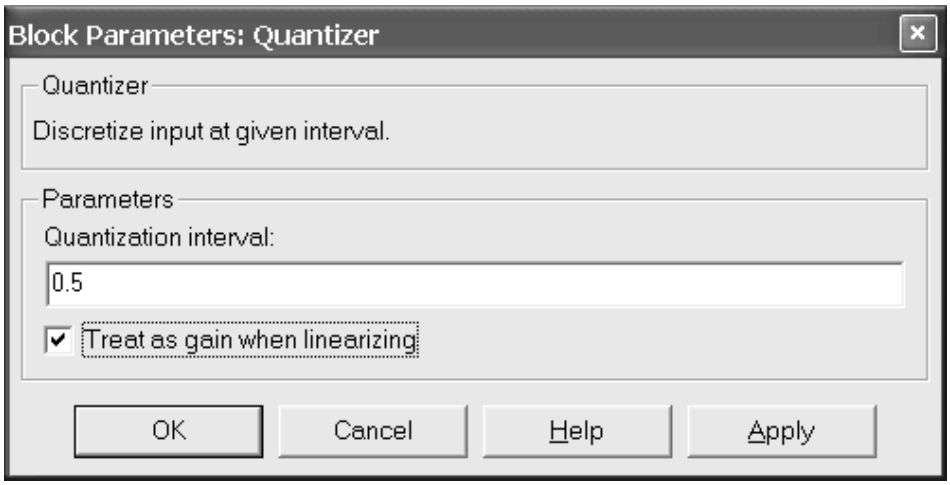


Рис. 17.10. Діалогове вікно параметрів елемента `Quantizer` і його характеристики

На рис. 17.11 показаний приклад використання блока `Quantizer`, що виконує квантування за рівнем синусоїдального сигналу із шагом 0,5. Для більшої зручності, схема включення блока відмінна від схеми, показаної на рис. 17.2.

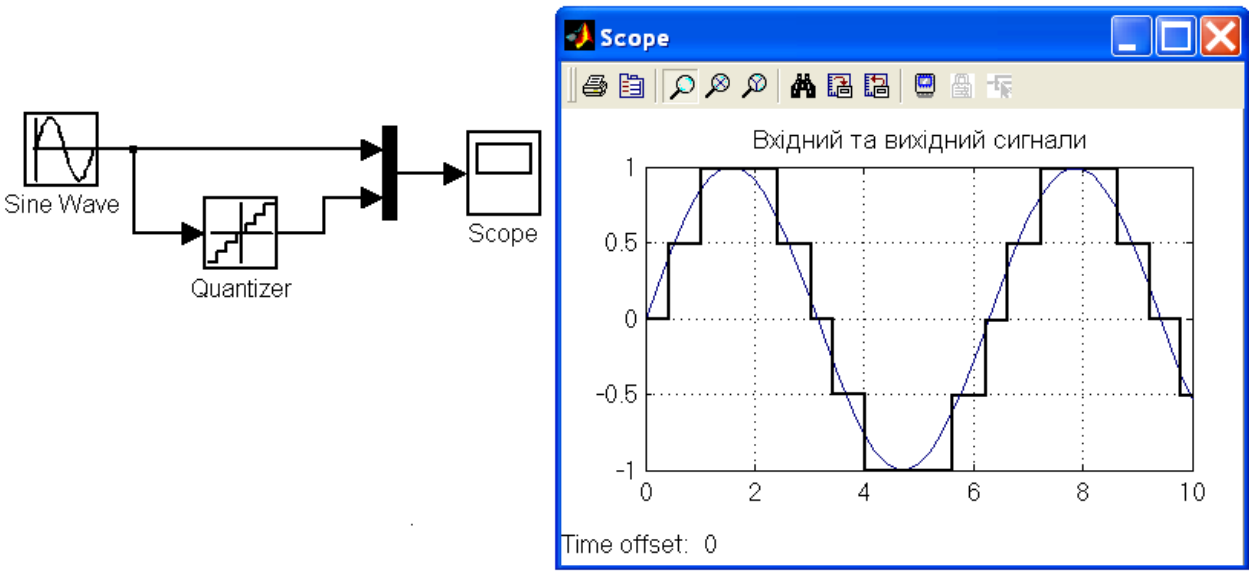
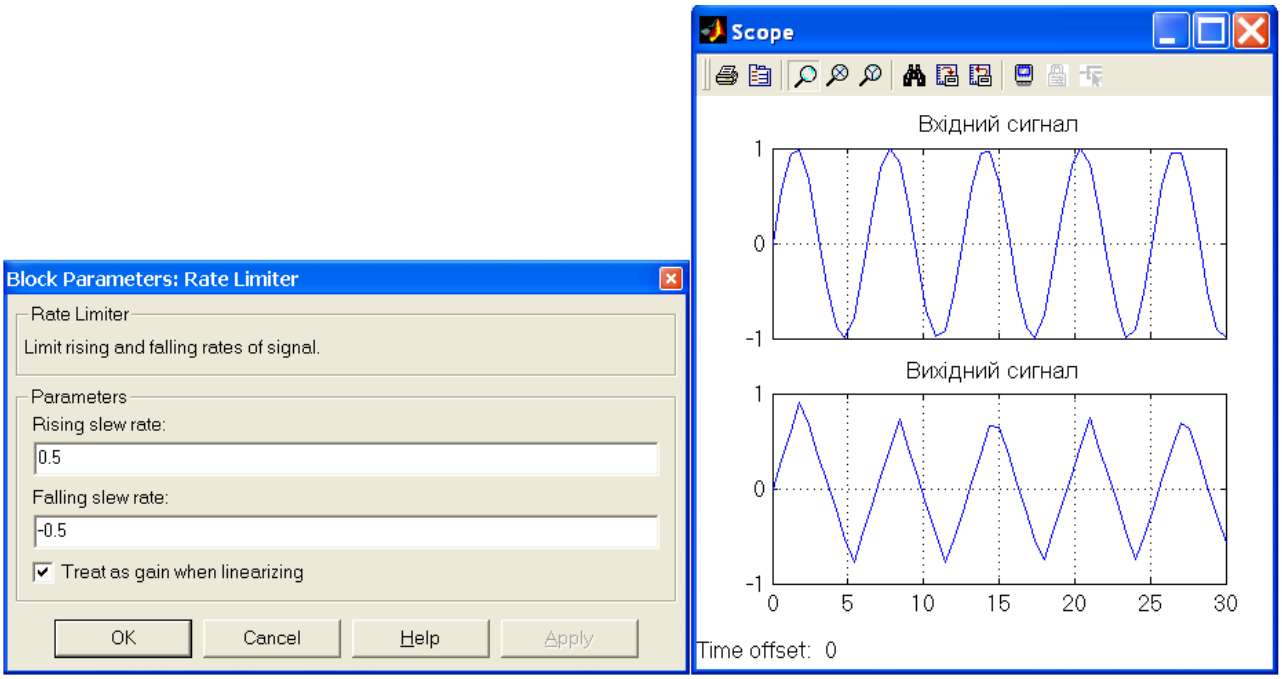


Рис. 17.11. Ілюстрація роботи блока `Quantizer`

17.7 Блок `Rate Limiter`

Блок `Rate Limiter` (Обмежувач швидкості) забезпечує обмеження швидкості зміни вхідного сигналу. Вікно параметрів цього блока і реакція на синусоїду представлені на рис. 17.12.



а *б*

Рис. 17.12. Вікно параметрів блока `Rate Limiter` і його реакція на синусоїду

Вікно містить два поля введення:

- **Rising slew rate** – Поріг, що обмежує швидкість при збільшенні вхідного сигналу;

- **Falling slew rate** - Поріг, що обмежує швидкість при зменшенні вхідного сигналу;

і один прапорець:

- **Treat as gain when linearizing** - Розглядати як підсилювач при лінеаризації.

Якщо швидкість зміни вхідного сигналу вище значення *R*, зазначеного в поле **Rising slew rate**, то вихідний сигнал розраховується за формулою:

$$y(i) = \Delta t \cdot R + y(i-1), \qquad (17.3)$$

де $y(i)$ і $y(i\text{-}1)$ - значення вихідного сигналу на поточному і попередньому шагу;

$$\Delta t = t(i) - t(i\text{-}1);$$

$t(i)$ - час на поточному шагу;

$t(i\text{-}1)$ - час на попередньому шагу.

Якщо швидкість зміни вхідного сигналу менше, ніж значення *F* параметра **Falling slew rate**, то вихідний сигнал блока обчислюється за формулою:

$$y(i) = \Delta t \cdot F + y(i-1). \qquad (17.4)$$

Якщо ж швидкість зміни сигналу лежить у межах між *F* і *R*, то вихідний сигнал блока дорівнює вхідному.

17.8 Блок `Relay`

Цей блок (рис. 17.1, *ж)* являє собою модель двопозиційного реле. Вікно параметрів блока `Relay`, прийняте за замовчуванням, зображене на рис. 17.13.

Якщо вхідний сигнал блока більше граничного значення, заданого в полі параметрів **Switch on point**, то на виході з'являється постійний сигнал, значення якого задається в полі **Output when on**. Якщо ж вхідний сигнал менше граничного значення, заданого в

полі параметрів **Switch off point**, то на виході блока з'являється постійний сигнал, значення якого задається в полі **Output when off**.

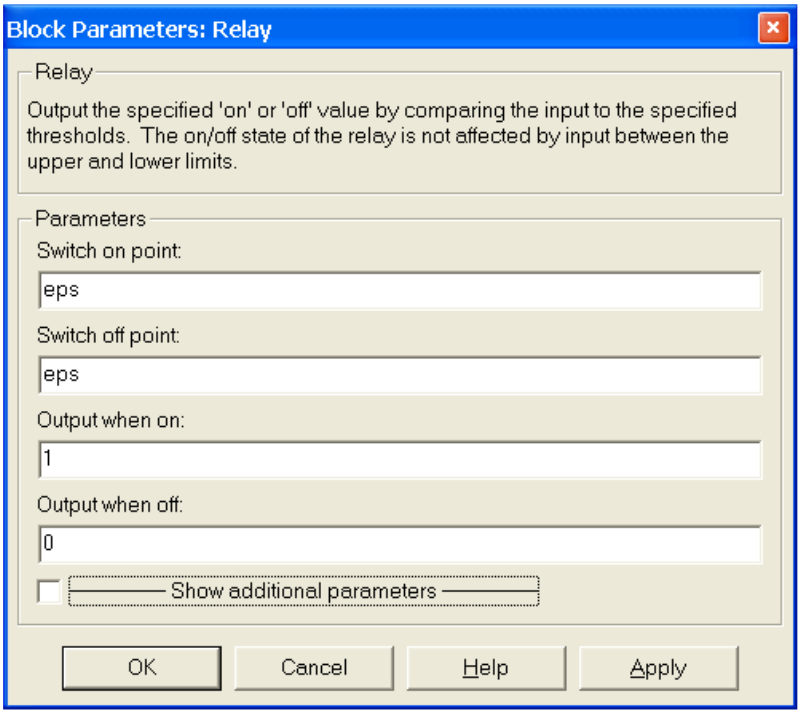


Рис. 17.13. Вікно параметрів блока `Relay`, прийняте за замовчуванням

Значення параметра, зазначеного в полі **Switch on point,** повинне бути більше значення, зазначеного в полі **Switch off point** (якщо необхідно враховувати явище гістерезису), або дорівнювати йому (якщо моделюється ідеальне реле). Рис. 17.14 показує роботу блока `Relay` з однаковими за абсолютною величиною й дуже малими (**eps**) порогами при подачі на вхід синусоїдального сигналу.

Якщо поставити прапорець у полі **Show additional parameters**, то стануть доступними додаткові параметри блока (рис. 17.15), які, втім, подібні з параметрами багатьох інших блоків (наприклад, `Sum`).

У списку, що розкривається, **Output data type mode** задається тип і масштаб вихідних даних: вони такі ж, як і у вхідних даних **(All ports same datatype**), або успадковуються зі зворотного зв'язку **(Inherit via back propagation**). При виборі параметра **Specify via dialog** стануть видимими додаткові поля **Output data type**, **Output scaling value** й **Parameter Scaling,** у які вручну можна задавати тип і масштаб вихідних даних.

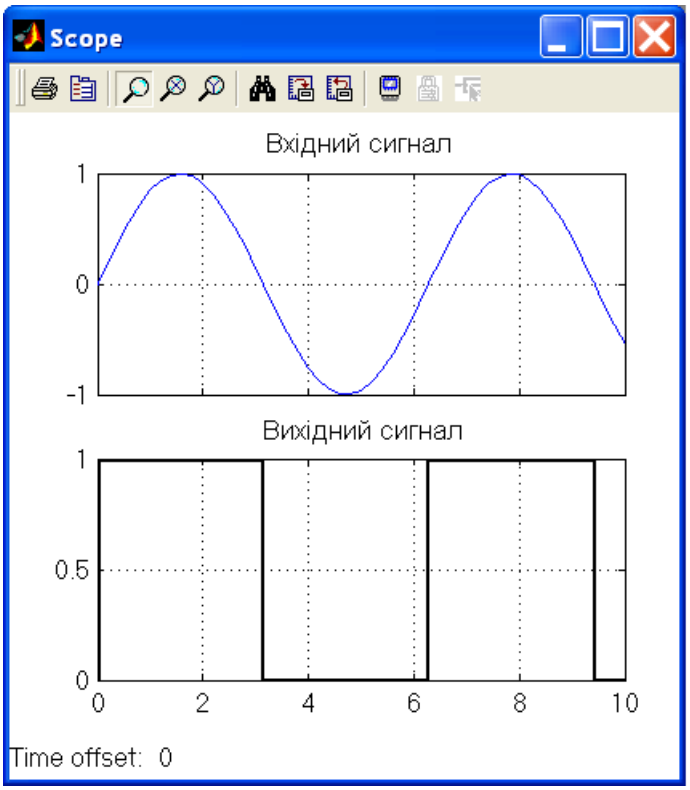


Рис. 17.14. Робота блока `Relay`

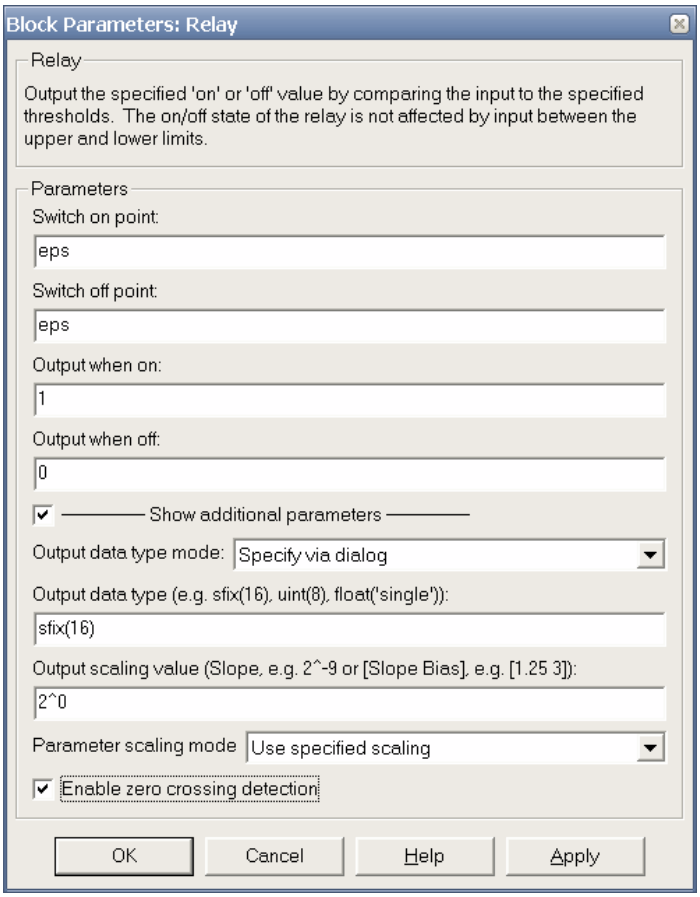


Рис. 17.15. Вікно параметрів блока з усіма активними вкладками

Нарешті, встановлюючи прапорець напроти параметра **Enable zero crossing detection,** можна дозволити або заборонити визначення перетинання нуля.

17.9 Блок `Saturation`

Даний блок являє собою модель ще однієї розповсюдженої нелінійності - «насичення» або «обмеження». Сигнал на виході блока дорівнює вхідному сигналу доти, поки не досягне заданих порогів обмеження: верхнього або нижнього. Після цього сигнал перестає змінюватися (рис. 17.16). Подібною характеристикою володіють багато реальних пристроїв, наприклад, підсилювачі, які обмежені по потужності в області більших вхідних сигналів.

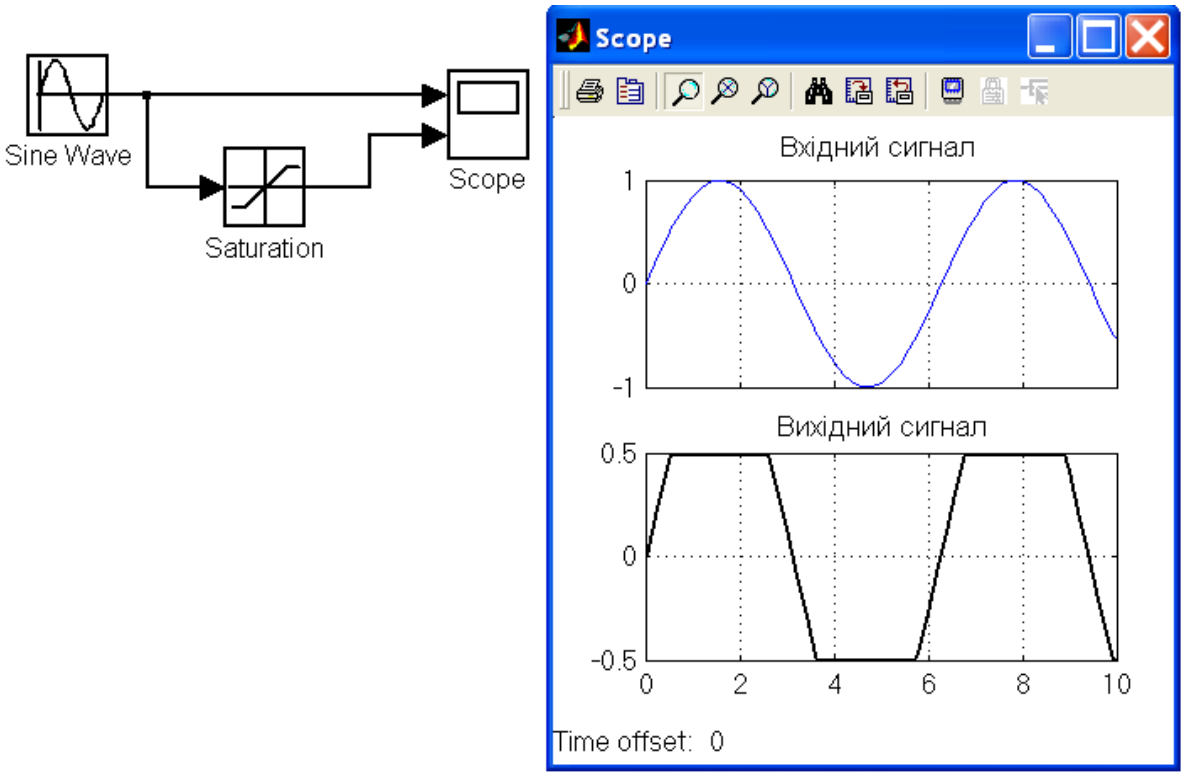


Рис. 17.16. Робота блока `Saturation`

Вікно параметрів блока `Saturation` представлене на рис. 17.17.

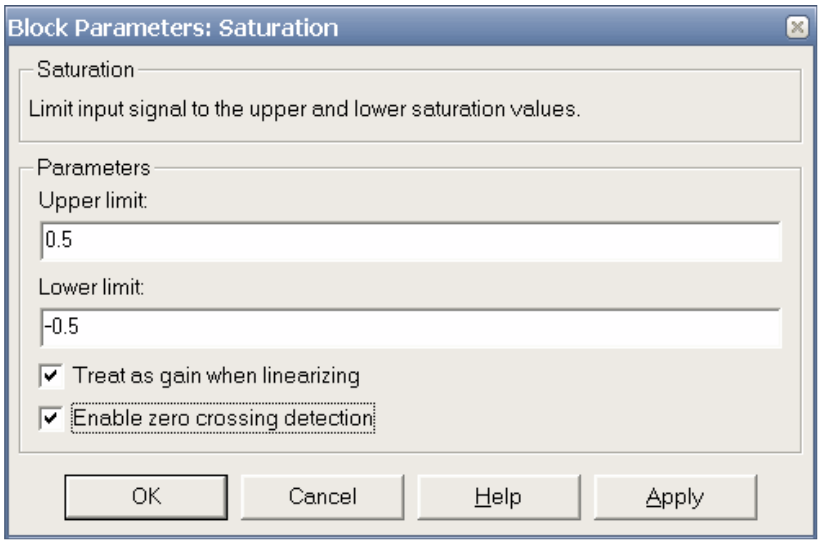


Рис. 17.17. Діалогове вікно параметрів елемента `Saturation`

Вікно містить два поля введення:

- **Upper limit:** - Верхня межа насичення;
- **Lower limit:** - Нижня межа насичення;

і два вже знайомих параметри:

- **Treat as gain when linearizing** - Розглядати як підсилювач при лінеаризації;
- **Enable zero crossing detection** - Дозволити визначення перетинання нуля.

17.10 Створення складних нелінійностей

17.10.1 З'єднання нелінійних блоків

Звичайно, розглянуті вище блоки не охоплюють всі нелінійні ефекти, які можуть зустрітися в системах керування. Так це й не можливо. Нелінійності, які не представлені окремими блоками в бібліотеці `Discontinuities`, можна задати комбінацією двох або декількох блоків. Наприклад, на рис. 17.18 показана модель трьохпозиційного реле з гістерезисом. На цій схемі задіяний новий блок `XY Graph` (Графопобудовник) з бібліотеки блоків реєстрації сигналів `Sinks`. Блок `XY Graph` будує графік виду $Y(X)$ і має два входи. Верхній вхід призначений для подачі сигналу, що є аргументом (X), нижній – для подачі значень функції (Y).

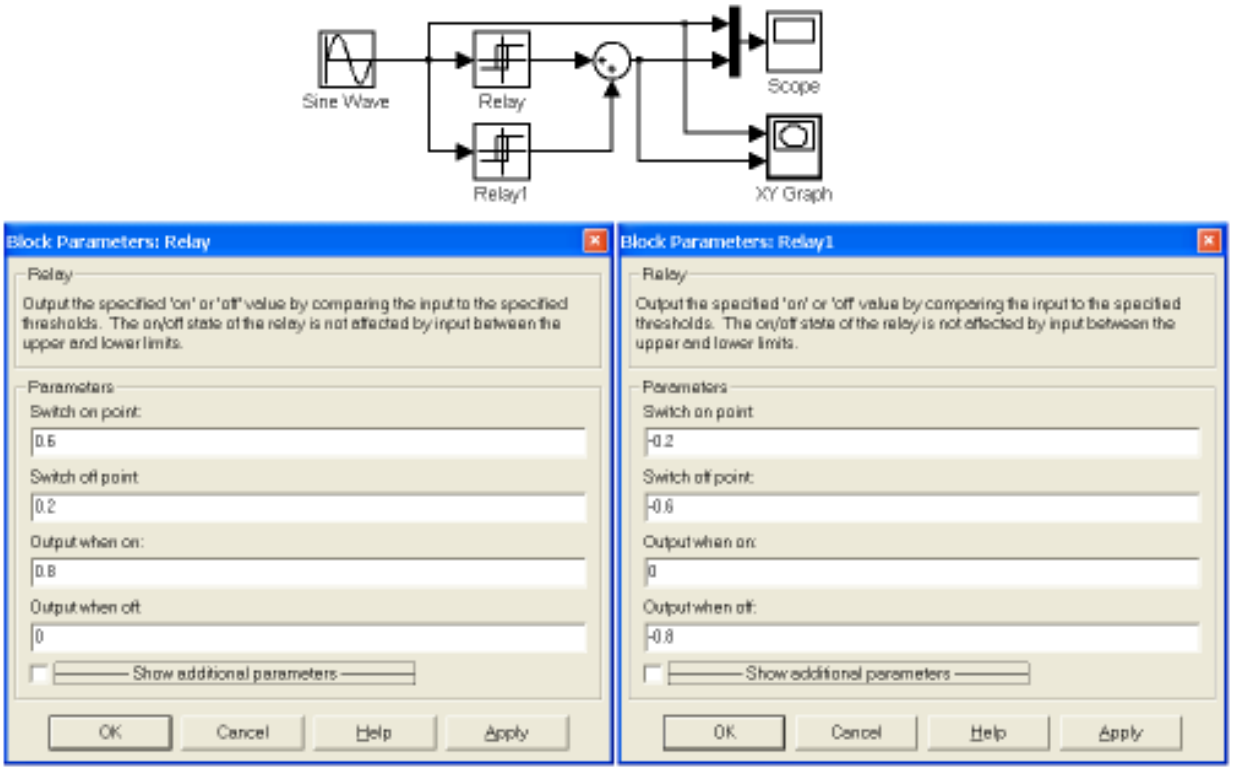


Рис. 17.18. Simulink-модель трипозиційного реле з гістерезисом і параметри настроювань блоків `Relay`

У діалоговому вікні параметрів блока (рис. 17.19) задаються граничні значення за вісями координат (**x-min**, **x-max**, **y-min**, **y-max**) і шаг квантування (**Sample time**).

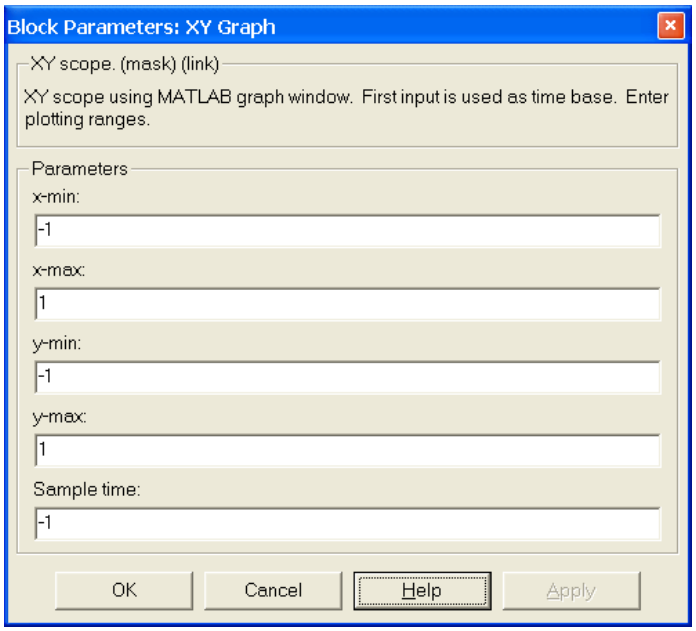
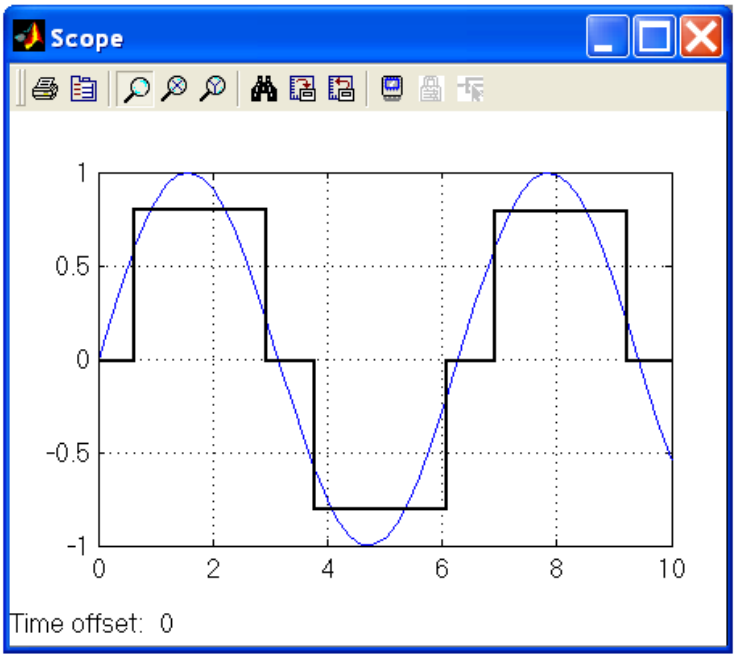
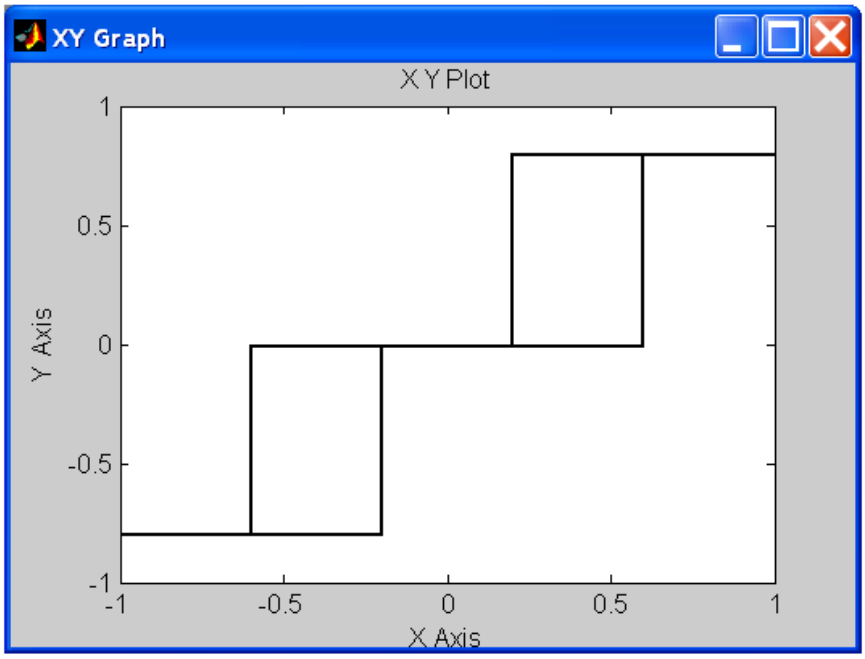


Рис. 17.19. Вікно параметрів блока `XY Graph`

Результат моделювання трьохпозиційного реле з гістерезисом наведений на рис. 17.20, *а*. Використання блоку `XY Graph` дозволяє побудувати статичну характеристику результуючої нелінійності (рис. 17.20, *б*).



а

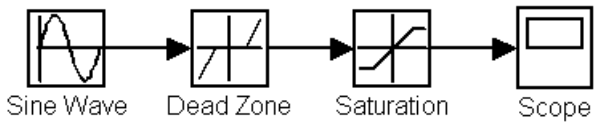


б

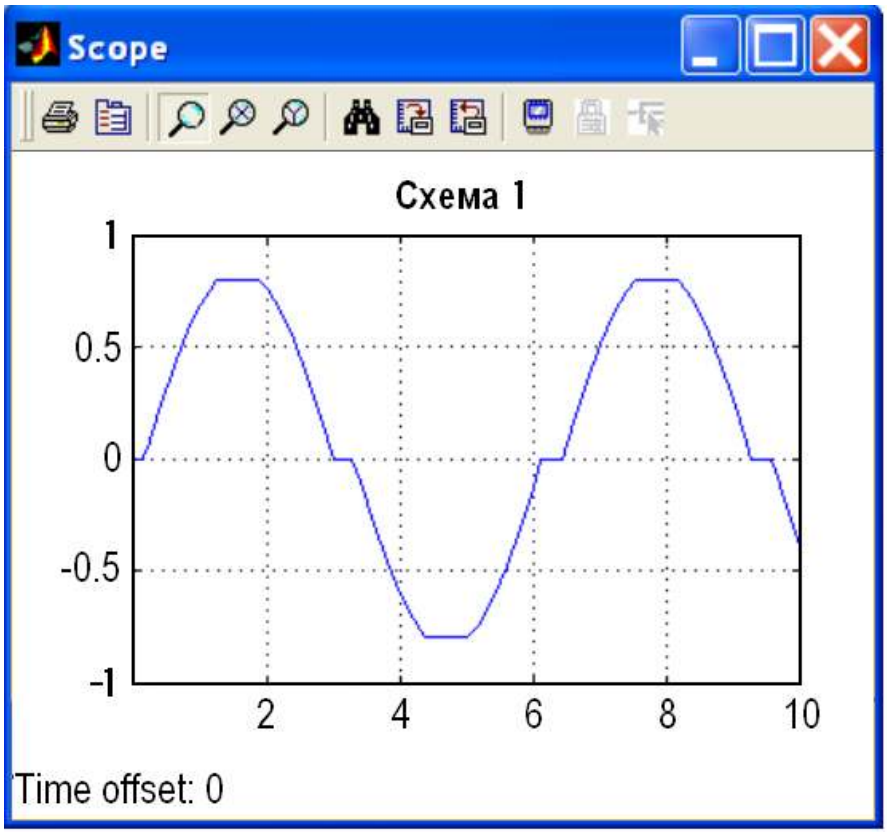
Рис. 17.20. Результати моделювання трипозиційного реле з гістерезисом:
а - реакція на синусоїду; *б* - статична характеристика

Розглянемо ще один приклад. Багато реальних елементів в області малих вхідних сигналів мають зону нечутливості, а в області великих сигналів мають ефект насичення. Спробуємо побудувати Simulink-модель такої нелінійної ланки з наступними параметрами: зона нечутливості лежить у межах -0,15 < *u* < 0,15, а ефект насичення наступає при ±0,8*y*, де *u* – вхідний сигнал, а *y* – вихідний сигнал. Очевидно, що модель буде складатися з двох послідовно з'єднаних нелінійних блоків: `Dead Zone` і `Saturation`. Однак тут дуже важливий порядок з'єднання нелінійних блоків.

Спочатку розглянемо спосіб з'єднання блоків, представлений на рис. 17.21, *а*. Результат моделювання представлений на рис. 17.21, *б*.



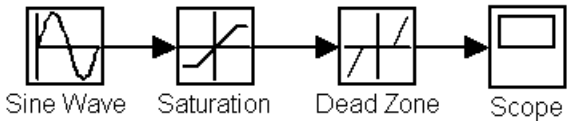
а



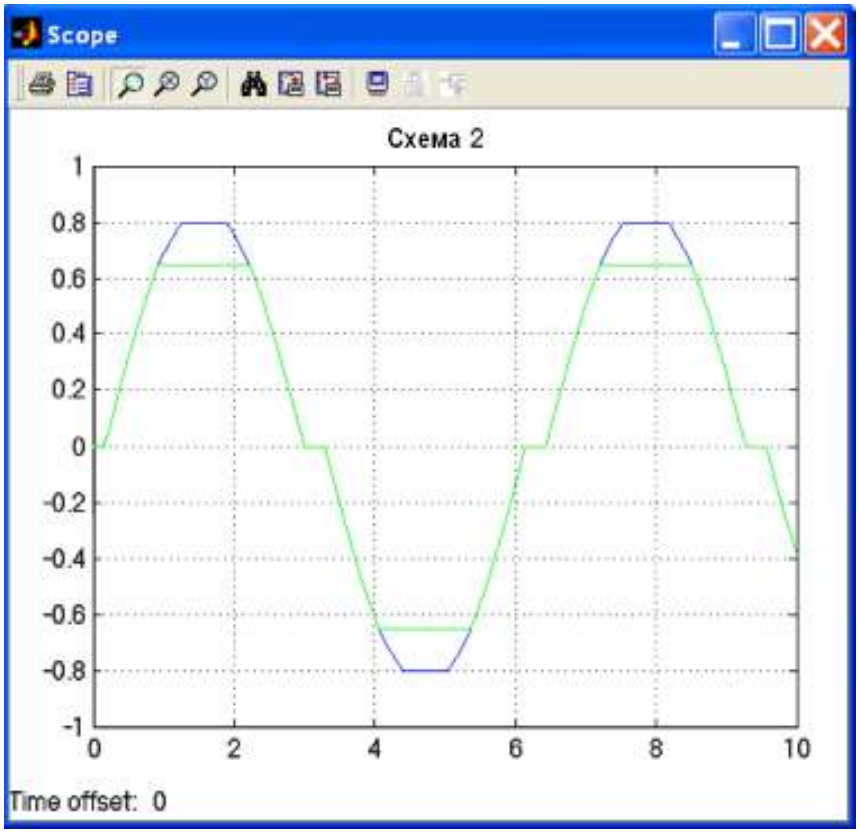
б

Рис. 17.21. Перший варіант з'єднання нелінійних блоків (*а*) і результат моделювання (*б)*

Тепер розглянемо другий спосіб з'єднання блоків (рис. 17.22, *а*). Сполучивши для зручності отриманий результат з вихідним сигналом попередньої моделі (рис. 17.22, *б*) добре видно, що отримані криві відрізняються.



а

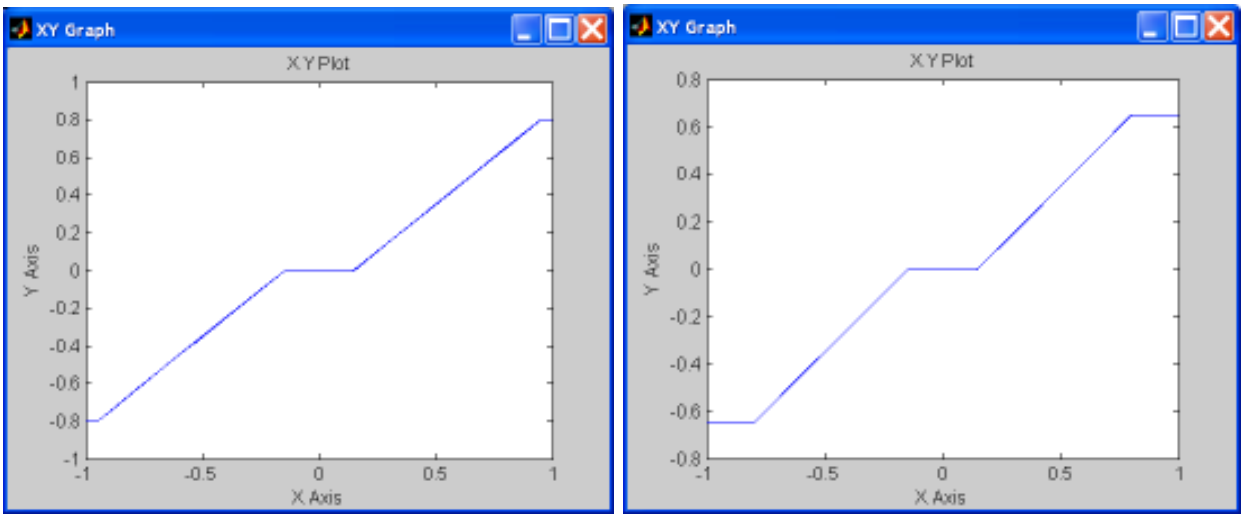


б

Рис. 17.22. Другий варіант з'єднання нелінійних блоків (*а*) і результат моделювання (*б)*

Це пов'язане з тим, що на відміну від лінійних блоків, *нелінійні блоки не можна міняти місцями,* тому що в результаті можна одержати нелінійність, що істотно відрізняється від бажаної. У нашому випадку загальний вид статичної характеристики у залежності від послідовності з'єднання блоків принципово не міняється. Міняється тільки ширина смуги пропускання. Це добре видно на статичних характеристиках результуючих нелінійностей

(рис. 17.23). Таким чином, для моделювання нелінійності типу «зона нечутливості з обмеженням» варто використати першу модель, що показана на рис. 17.21, *а*.



а *б*

Рис. 17.23. Статичні характеристики нелінійності при різній послідовності з'єднань блоків `Dead Zone` і `Saturation`: *а* - `Dead Zone` → `Saturation`; *б* - `Saturation` → `Dead Zone`

17.10.2 Блоки функцій

У тому випадку, якщо нелінійність не можна описати яким-небудь блоком з бібліотеки `Discontinuities` або їхнім з'єднанням, можна використати блоки `Fcn` (Функція) і `MATLAB Fcn` (Функція MATLAB) бібліотеки `User-Defined Functions` (Функції, які визначаються користувачем).

Блок `Fcn` формує скалярний вихідний сигнал як функцію від вхідної змінної *u* або вектора вхідних змінних *u*(*n*) (*n* – номер елемента вхідного вектора). Зображення і діалогове вікно даного блока наведені на рис. 17.24.

Діалогове вікно блока в групі опцій **Parameters** містить поле **Expression** (Вираження), де вводиться вираження мовою С, що задає необхідну функцію. У вираженні можуть використовуватися такі функції, як `abs, exp, sin, cos, asin, acos, ln, log, log10` та ін., прийняті в мові С знаки арифметичних

операцій (+, -, ^, *, /), оператори відношення (= =, !=, >, >=, <, <=), знаки логічних операцій ((&& (логічне І) і || (логічне АБО)).

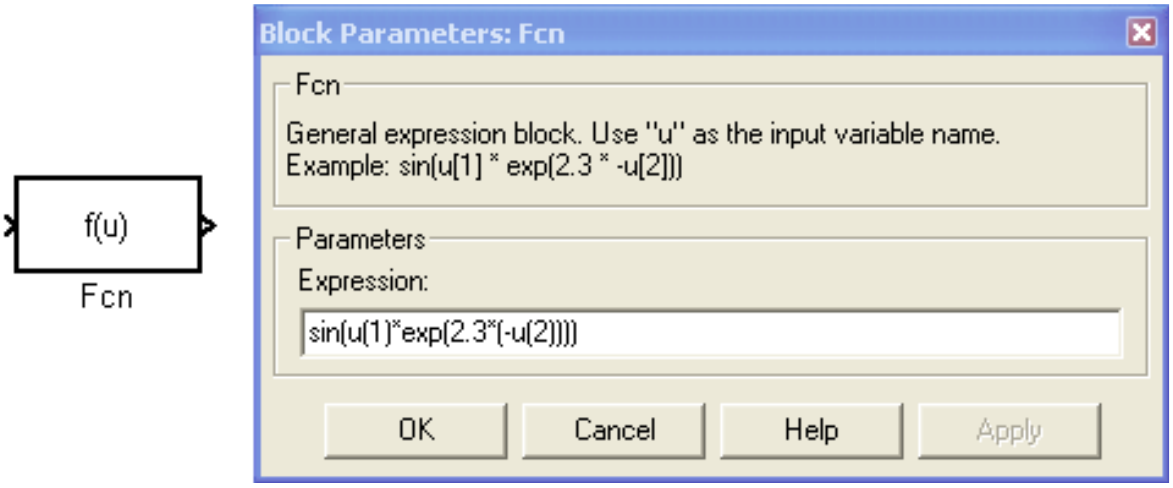


Рис. 17.24. Блок `Fcn` і діалогове вікно його параметрів

Блок `MATLAB Fcn` (рис. 17.25) дозволяє задавати функцію за правилами, прийнятим у мові програмування MATLAB. На відміну від попереднього блока, блок `MATLAB Fcn` дозволяє виконувати матричні операції та формувати вихідну змінну у вигляді вектора. Однак це зменшує швидкість виконання операцій.

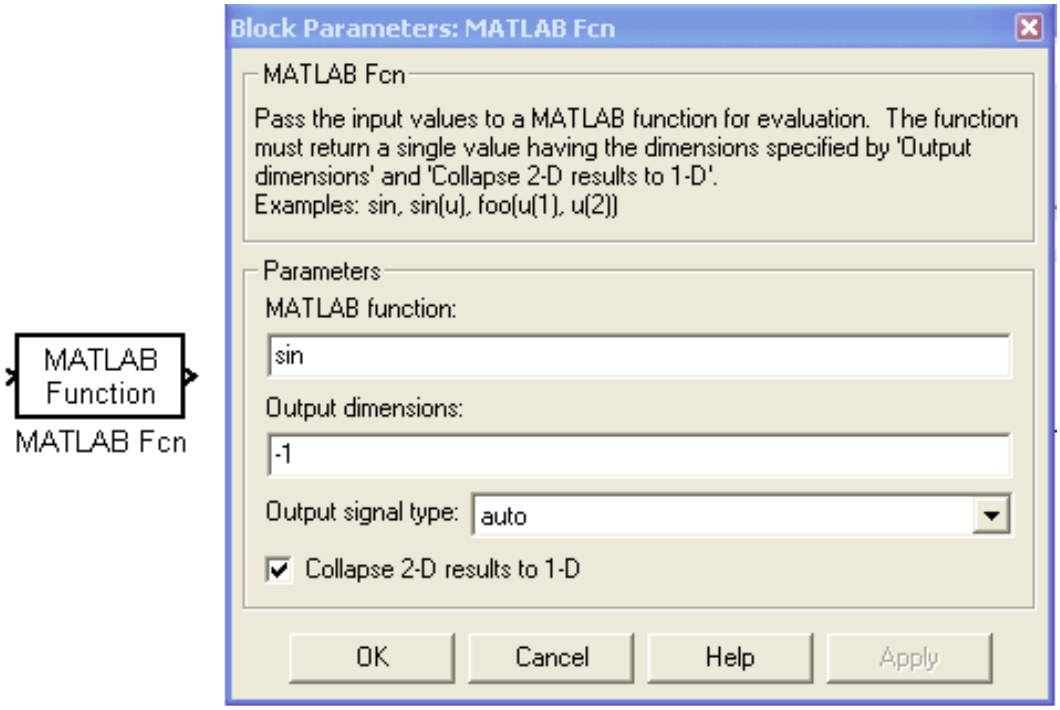


Рис. 17.25. Блок `MATLAB Fcn` і діалогове вікно його параметрів

Діалогове вікно елемента `MATLAB Fcn` містить два текстові поля: **MATLAB Function** і **Output dimensions**, список, що розкривається, **Output signal type** та один прапорець **Collapse 2-D results to 1-D**.

У полі **MATLAB Function** вказується залежність вихідного сигналу від вхідного *u* мовою MATLAB. Якщо вхідний сигнал є вектором, то необхідно вказувати номер елемента вектора в круглих або квадратних дужках. Якщо вираження складається з однієї функції, то її можна задати без вказування параметрів. Вираження може містити також власні функції користувача, написані мовою MATLAB і оформлені у вигляді М-файлів. Ім'я m-файлу не повинне збігатися з ім'ям моделі (mdl-файлом).

В полі **Output dimensions** (Розмірність вихідного сигналу) задається розмірність вектора вихідної змінної. За замовчуванням цей параметр має значення, що дорівнює -1. Це означає, що розмірність вектора вихідного сигналу збігається з розмірністю вектора вхідного сигналу.

У списку, що розкривається, **Output signal type** вказується тип вихідного сигналу:

- **auto** – тип сигналу визначається автоматично;
- **real** – речовинний сигнал;
- **complex** – комплексний сигнал.

Встановленням прапорця напроти параметра **Collapse 2-D results to 1-D** дозволяється або забороняється перетворення вектора вихідної змінної в скаляр.

Завдання для самостійної роботи

1. Нелінійна система має структуру, що представлена на рис. 17.26, де

$$W(s) = \frac{4}{s(0,05s^2 + 0,5s + 1)}$$

а) За допомогою команди `ss` переведіть модель лінійної частини у простір станів.

б) Дослідить автоколивання системи при $a = 0$; $b = 1$ та $c = 4$.

в) Дослідить вплив значення параметру *c* нелінійної ланки на поведінку системи.

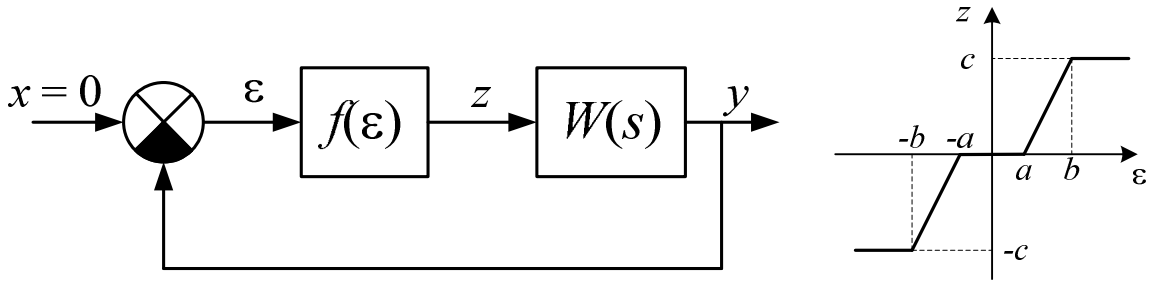


Рис. 17.26. Структура нелінійної системи та тип нелінійності

2. На рис. 17.26 параметри нелінійна ланка має наступні параметри: $a = 0$; $b = 1$ та $c = 0{,}5$, а передаточна функція лінійної частини $W(s) = 1/s^2$.

а) Побудуйте вихідний сигнал системи при початкових умовах $\mathbf{x}(0) = [1\ 1]^T$ та визначить амплітуду і частоту коливань.

б) Змініть початкові умови та проаналізуйте одержані результати. Чи є вихідні коливання системи автоколиваннями? Чому?

в) Дослідить вплив значення параметру *c* нелінійної ланки на поведінку системи.

г) Повторіть дії п.п. *а*) – *в*) при наступних параметрах нелінійної ланки: $a = 0{,}1$; $b = 0{,}5$ та $c = 0{,}6$.

3. На рис. 17.26 нелінійність *f*(ε) – ланка типу «люфт» з величиною люфту $a = 1$, а передаточна функція лінійної частини

$$W(s) = \frac{4}{s(0{,}5s+1)(s+1)}.$$

а) Визначить поведінку вихідного сигналу при початкових умовах $\mathbf{x}(0) = [0{,}1\ \ 0\ \ 0]^T$ та $\mathbf{x}(0) = [1\ \ 0\ \ 0]^T$. Як із збільшенням початкових умов змінюється вихідний сигнал?

б) Шляхом зміні величини люфту *a* зробіть висновок про його вплив на можливість появи незатухаючих коливань.

4. За допомогою блоку `Fcn` побудуйте графіки наступних функцій від *u,* якщо *u* – синусоїдальний сигнал:

а) $e^{2u}\cos 3u$; *б*) $\lg(2+u) - 2u$; *в*) $(u-3)\cdot\cos u$, *г*) $(u+1) - \sqrt{u^2+4}$.

18. ДИСКРЕТНІ СИСТЕМИ В SIMULINK

Пакет Simulink має широкі можливості з моделювання дискретних систем, для цього він має бібліотеку блоків `Discrete`. В Simulink 5.0 бібліотека `Discrete` містить наступні блоки: `Discrete Transfer Fcn` (Дискретна передаточна функція), `Discrete Zero-Pole` (Дискретна передаточна функція в ZPG-форматі), `Discrete Filter` (Дискретний фільтр), `Discrete State-Space` (Дискретна система в просторі станів), `Discrete-Time Integrator` (Чисельне інтегрування), `Zero-Order Hold` (Екстраполятор нульового порядку), `First-Order Hold` (Екстраполятор першого порядку), `Memory` (Пам'ять) і `Unit Delay` (Затримка).

При моделюванні дискретних систем передбачається, що кожен блок дискретної системи має квантувач на вході та екстраполятор нульового порядку на виході. Крім того, загальним для дискретних блоків є такий параметр як період квантування **(Sample time),** що або є скаляром - проміжком часу між моментами квантування, або вектором розмірності 1x2, де першим компонентом є величина T - інтервал між імпульсами, а другим компонентом - значення часу зсуву (**offset**).

18.1 Блок `Discrete Transfer Fcn`

Блок `Discrete Transfer Fcn` (рис. 18.1) задає дискретну передаточну функцію $W(z)$ у вигляді:

$$W(z) = \frac{\mathbf{num}(z)}{\mathbf{den}(z)} = \frac{\mathrm{num}_m z^m + \mathrm{num}_{m-1} z^{m-1} + ... + \mathrm{num}_1 z + \mathrm{num}_0}{\mathrm{den}_n z^n + \mathrm{den}_{n-1} z^{n-1} + ... + \mathrm{den}_1 z + \mathrm{den}_0}, \quad (18.1)$$

де **num**(z) - вектор або матриця коефіцієнтів чисельника,

den(z) - вектор коефіцієнтів знаменника,

m, n - порядок чисельника і знаменника, відповідно.

Зрозуміло, порядок чисельника не повинен перевищувати порядок знаменника. Вхідний сигнал блоку повинен бути скалярним.

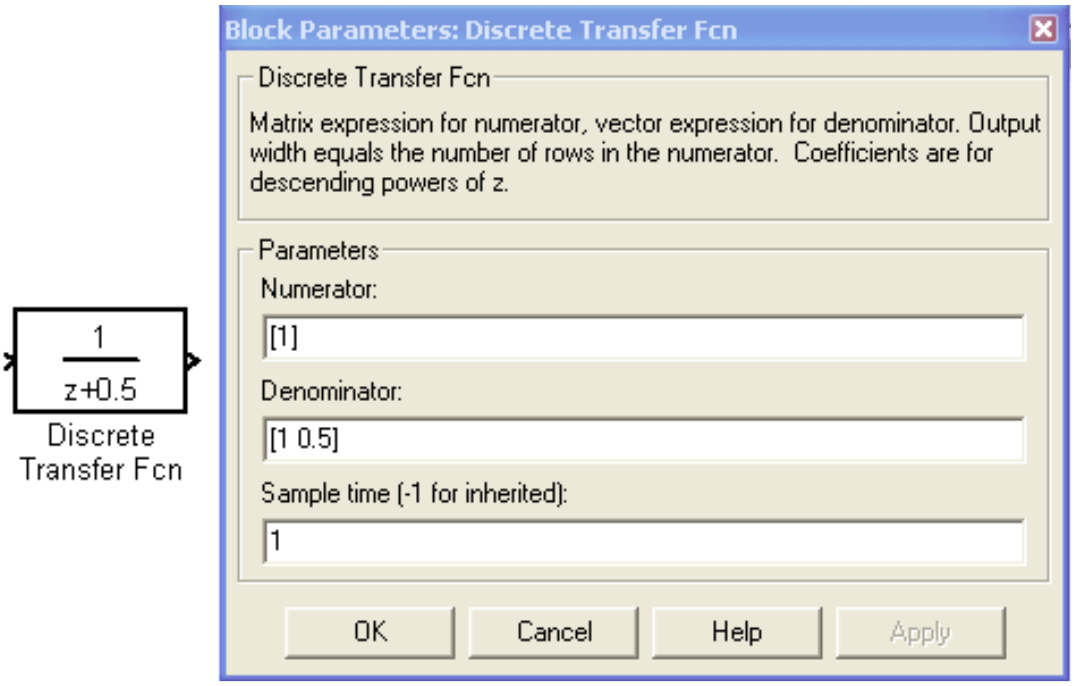


Рис. 18.1. Блок `Discrete Transfer Fcn`

Діалогове вікно блока містить три поля: **Numerator** (Чисельник), у яке вводяться коефіцієнти чисельника дискретної передаточної функції; **Denominator** (Знаменник), де вводиться вектор коефіцієнтів знаменника і поле **Sample time (-1 for inherited)** (Період квантування (-1 для спадкування)), у якому вказується період квантування. Правила введення коефіцієнтів чисельника та знаменника дискретної передаточної функції аналогічні правилам для блоку `Transfer Fcn` бібліотеки `Continuous`.

На рис. 18.2 показаний приклад використання блока `Discrete Transfer Fcn`. У прикладі розраховується реакція на одиничний східчастий вплив дискретної передаточної функції

$$W(z) = \frac{0{,}01465z + 0{,}01377}{z^2 - 1{,}801z + 0{,}829},$$

яка є дискретним аналогом коливальної ланки:

$$W(z) = \frac{1}{2s^2 + 1{,}5s + 1},$$

при періоді квантування 0,25 с і знайдена в MATLAB за допомогою команди `c2d`.

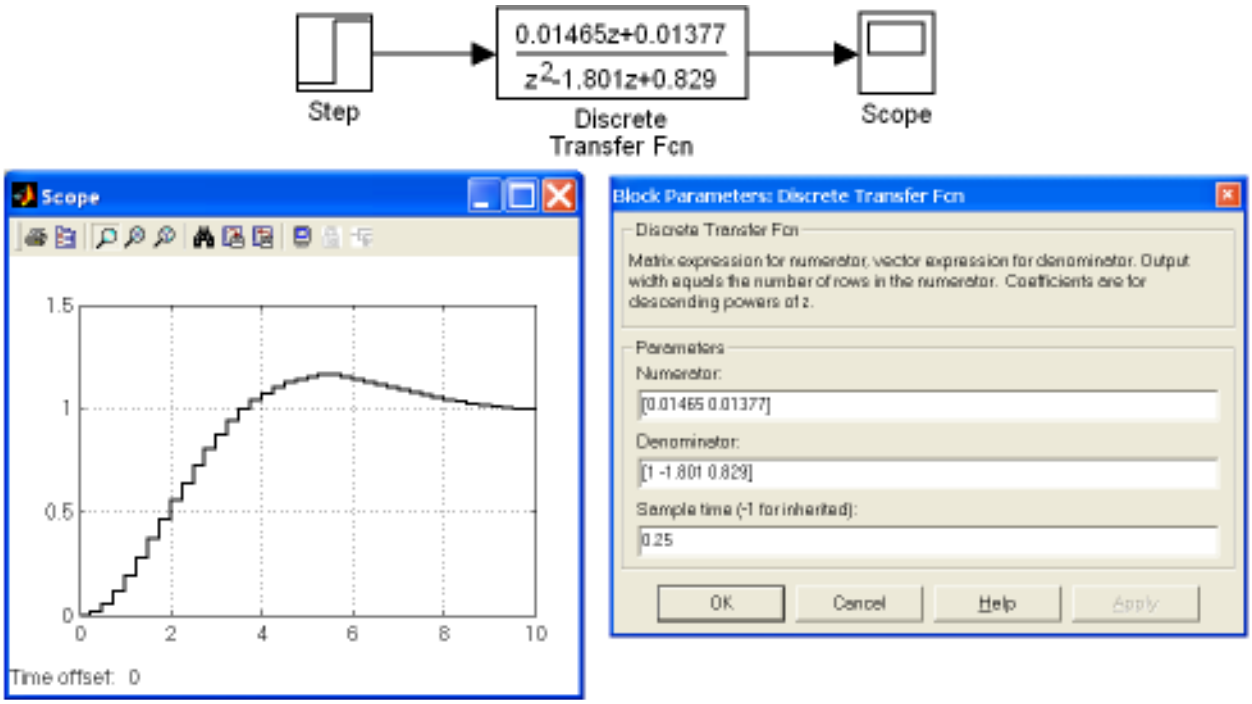


Рис. 18.2. Приклад використання блока
`Discrete Transfer Fcn`

18.2 Блок `Discrete Zero-Pole`

Блок `Discrete Zero-Pole` визначає дискретну передаточну функцію із заданими полюсами і нулями:

$$W(z) = \boldsymbol{K}\frac{(z-Z_1)(z-Z_2)\ldots(z-Z_m)}{(z-P_1)(z-P_2)\ldots(z-P_n)}, \qquad (18.2)$$

де ***Z*** – вектор або матриця нулів передаточної функції,

P – вектор полюсів передаточної функції,

K – коефіцієнт підсилення передаточної функції, або вектор коефіцієнтів, якщо нулі передаточної функції задані матрицею. При цьому розмірність вектора ***K*** визначається числом рядків матриці нулів.

Зазначені параметри блока задаються у відповідних полях **Zeros** (нулі), **Poles** (полюси) і **Gain** (підсилення). Період квантування вказується в поле **Sample time (-1 for inherited)** (рис. 18.3).

Sample time - Шаг квантування за часом.

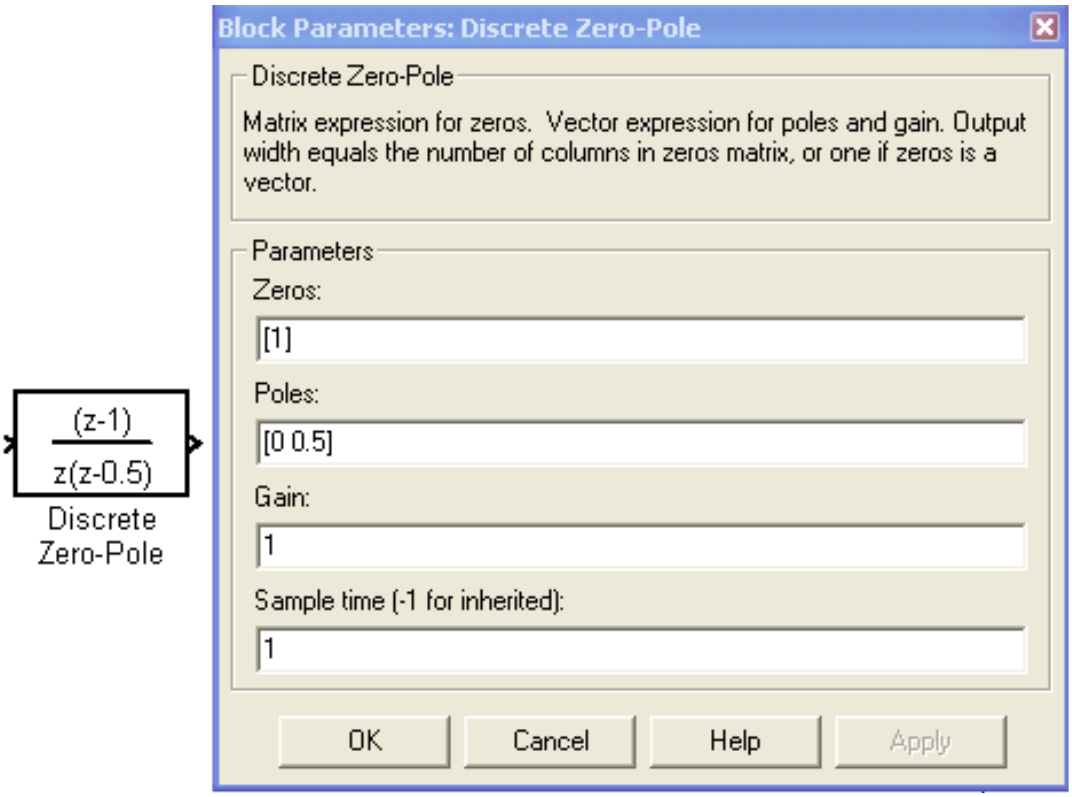


Рис. 18.3. Блок `Discrete Zero-Pole`

Кількість нулів не повинна перевищувати число полюсів передаточної функції.

У тому випадку, якщо нулі передаточної функції задані матрицею, то блок `Discrete Zero-Pole` моделює векторну передаточну функцію.

Нулі або полюси можуть бути задані комплексними числами. У цьому випадку нулі або полюси повинні бути задані комплексно-сполученими парами полюсів або нулів відповідно.

Приклад використання блока `Discrete Zero-Pole` наведений на рис. 18.4. Тут будується перехідна функція системи, модель якої представлена в ZPG-формі:

$$W(z) = \frac{0,014655(z + 0,9394)}{(z - 0,9005 + 0,1345i)(z - 0,9005 - 0,1345i)},$$

що відповідає дискретній передаточній функції з попереднього прикладу. Для її знаходження можна використати наступний script-file.

```
W=tf(1,[2 1.5 1])%Безперервна передаточна функція
Wd=c2d(W,0.25)  %Дискретна передаточна функція
                 %з періодом квантування 0,25
Wdzpk=zpk(Wd)   %Перетворення в ZPG-форму
p=[1 -1.801 0.829] %Знаходимо
roots(p)           %полюси передаточної функції
```

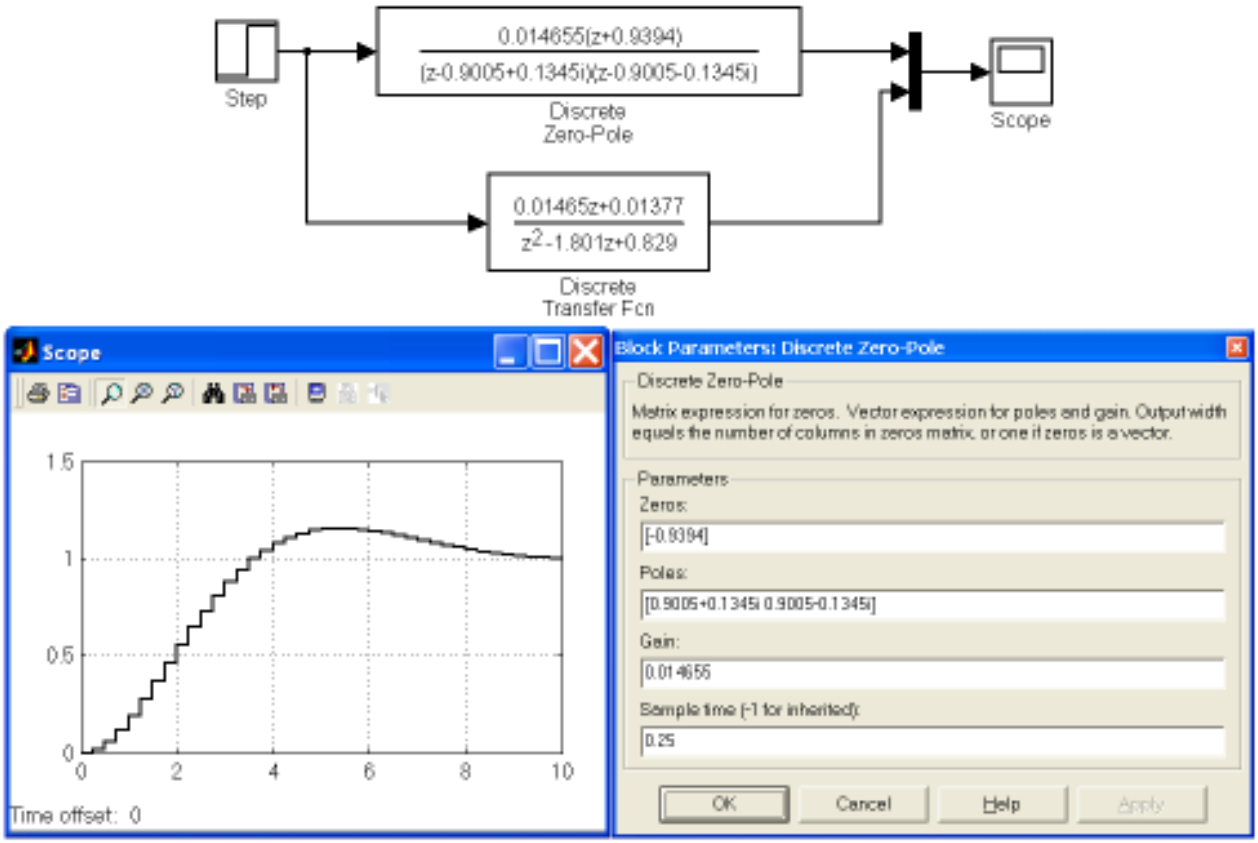


Рис. 18.4. Приклад використання блока `Discrete Zero-Pole`

Початкові умови при використанні блоків `Discrete Transfer Fcn` і `Discrete Zero-Pole` вважаються нульовими. Якщо необхідно задати ненульові початкові умови, то можна використати відповідні блоки з додаткової бібліотеки Simulink Extras або застосувати блок `Discrete State-Space`.

18.3 Блок `Discrete State-Space`

Блок моделює динамічний об'єкт у просторі станів, тобто описує його роботу за допомогою рівнянь (12.5) або інакше:

$$\begin{cases} \mathbf{x}(n+1) = \mathbf{A}\mathbf{x}(n) + \mathbf{B}\mathbf{u}(n); \\ \mathbf{y}(n) = \mathbf{C}\mathbf{x}(n) + \mathbf{D}\mathbf{x}(n). \end{cases} \quad (18.3)$$

де n - номер моменту квантування.

На рис. 18.5 показаний приклад моделювання об'єкта за допомогою блока `Discrete State-Space`.

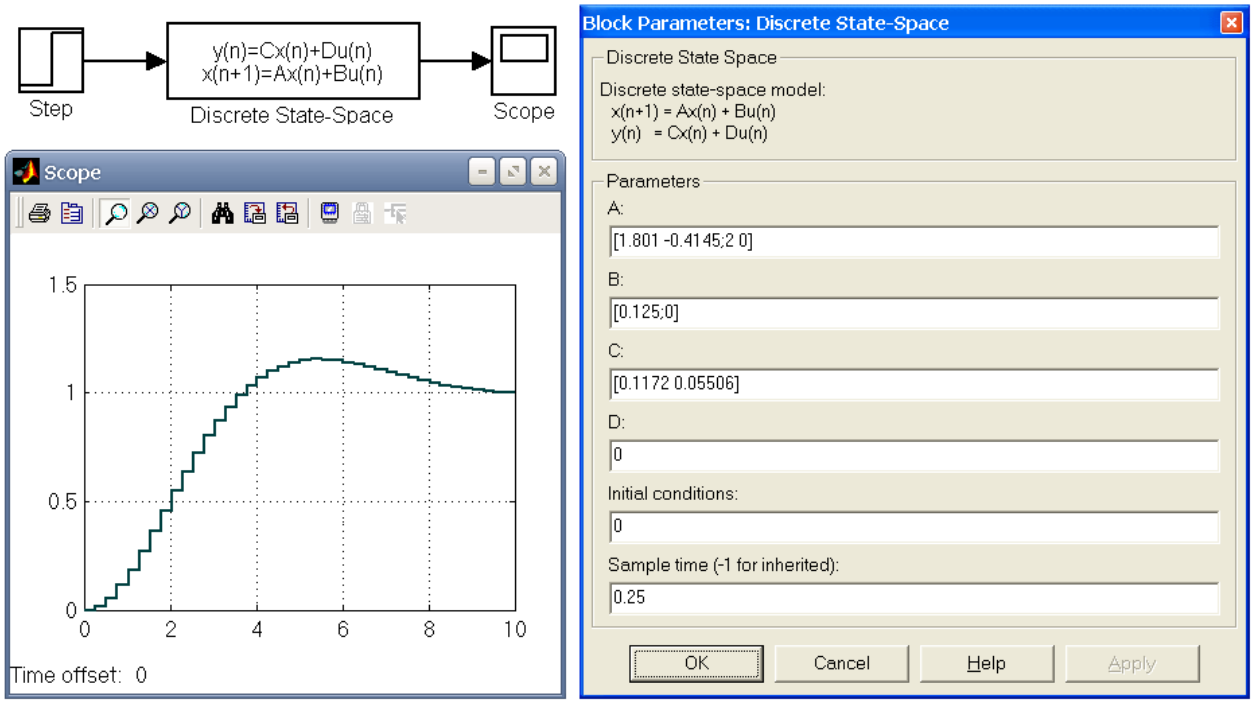


Рис. 18.5. Приклад роботи блока `Discrete State-Space`

Матриці блока мають наступні значення:

$$\mathbf{A} = \begin{bmatrix} 1{,}801 & -0{,}4145 \\ 2 & 0 \end{bmatrix}; \ \mathbf{B} = \begin{bmatrix} 0{,}125 \\ 0 \end{bmatrix}; \ \mathbf{C} = \begin{bmatrix} 0{,}1172 & 0{,}05506 \end{bmatrix}; \ \mathbf{D} = 0,$$

що відповідає дискретній передаточній функції, яка була розглянута на рис. 18.2.

18.4 Блок `Discrete Filter`

Блок дискретного фільтра `Discrete Filter` (рис. 18.6) задає дискретну передаточну функцію від зворотного аргументу z^{-1}. Таким чином, на відміну від блока `Discrete Transfer Fcn`, у

якому коефіцієнти багаточленів розташовуються в порядку убування змінної z, у блоці `Discrete Filter` коефіцієнти розташовуються в порядку зростання ступенів змінної z^{-1}:

$$W(z^{-1}) = \frac{\mathbf{num}(z^{-1})}{\mathbf{den}(z^{-1})} = \\ = \frac{\mathrm{num}_0 z^0 + \mathrm{num}_1 z^{-1} + \mathrm{num}_2 z^{-2} + \ldots + \mathrm{num}_m z^{-m}}{\mathrm{den}_0 z^0 + \mathrm{den}_1 z^{-1} + \mathrm{den}_2 z^{-2} + \ldots + \mathrm{den}_n z^{-n}}, \qquad (18.4)$$

де **num** - вектор або матриця коефіцієнтів чисельника,

den - вектор коефіцієнтів знаменника,

m+1 та n+1 - кількість коефіцієнтів чисельника і знаменника відповідно.

По суті, блоки `Discrete Transfer Fcn` і `Discrete Filter` відрізняються формою подання передаточної функції.

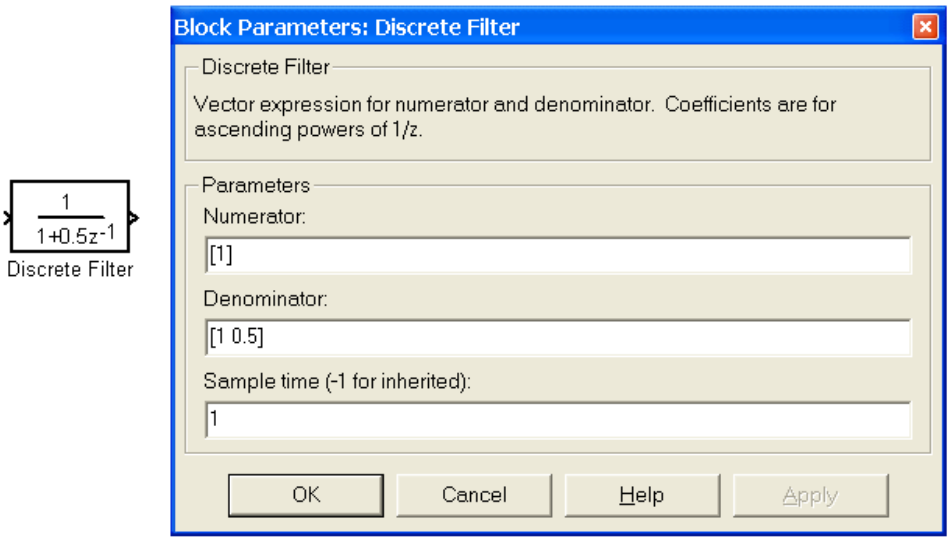


Рис. 18.6. Блок `Discrete Filter` і вікно його параметрів

Параметри блока (рис. 18.6) подібні параметрам блока `Discrete Transfer Fcn`. У полі **Numerator** вводяться коефіцієнти чисельника фільтра; у полі **Denominator** – коефіцієнти його знаменника, а в полі **Sample time (-1 for inherited)** задається період квантування.

Роботу блоку `Discrete Filter` ілюструє рис. 18.7. Поліном, зазначений у блоці на цьому рисунку, відповідає дискретній передаточній функції

$$W(z) = \frac{0{,}2212}{z - 0{,}7788},$$

яка, у свою чергу, є дискретним аналогом передаточної функції:

$$W(s) = \frac{1}{2s + 1}.$$

Шаг дискретизації обраний рівним 0,5 с.

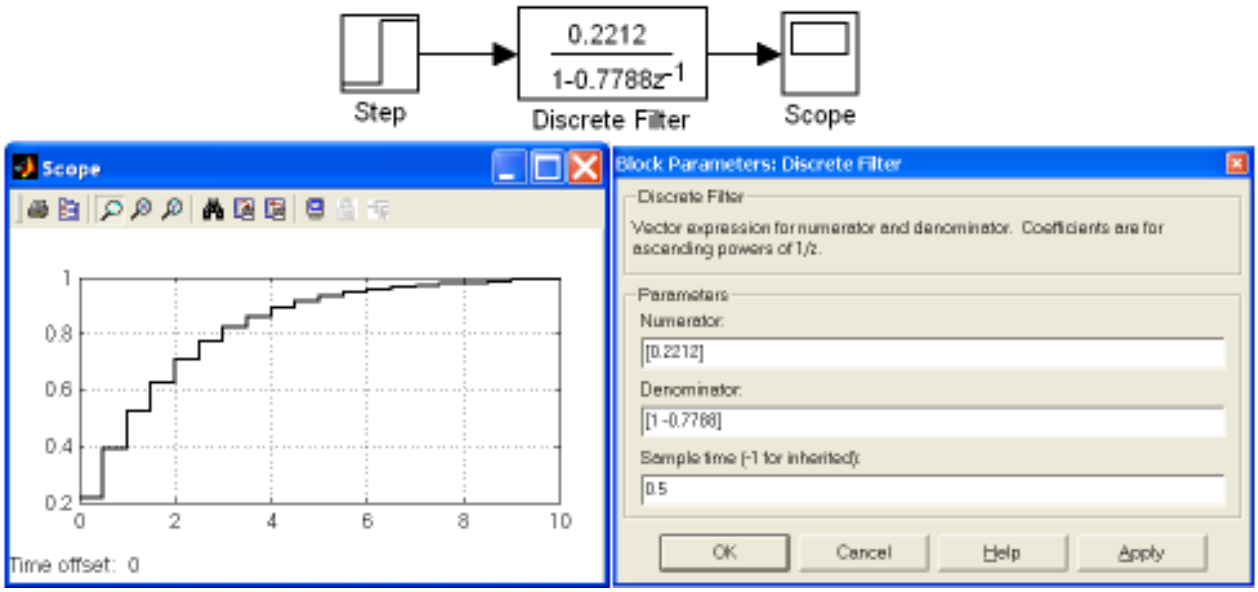


Рис. 18.7. Робота блока `Discrete Filter`

18.5 Блок `Unit Delay`

Блок `Unit Delay` (Затримка) встановлює затримку вихідного сигналу на час, рівний періоду квантування T, тобто вихідною змінною блока `Unit Delay` є значення вхідної змінної в момент часу, попередній моменту квантування. Таким чином, блок `Unit Delay` описується різницевим рівнянням:

$$y(k) = u(k\text{-}1), \qquad (18.5)$$

де $u(k\text{-}1)$ - вхідна змінна в моменти часу $(k\text{-}1)T$;

$y(k)$ - вихідна змінна в моменти часу kT;

k - номер шагу моделювання.

Вікно параметрів блока `Unit Delay` (рис. 18.8) має два текстові поля. У текстовому полі **Initial condition** (Початкові умови) задається значення вихідного сигналу блока в початковий момент часу. Призначення другого поля - **Sample time (-1 for inherited)** (Період квантування (-1 для спадкування)) – аналогічно попереднім блокам: тут встановлюється значення періоду квантування *T*.

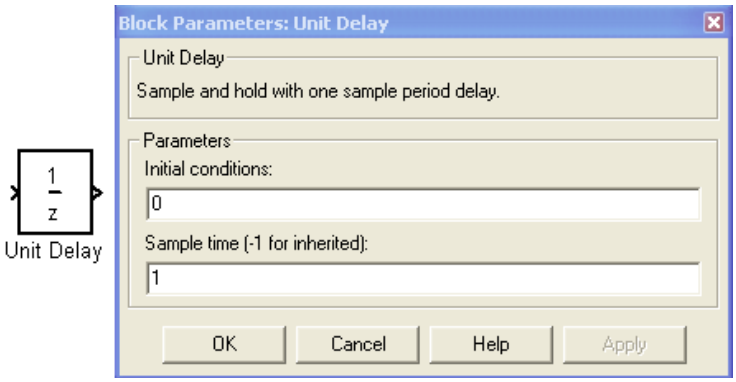


Рис. 18.8. Зображення та вікно параметрів блока `Unit Delay`

Вхідний сигнал блока може бути як скалярним, так і векторним. При векторному вхідному сигналі затримка виконується для кожного елемента вектора. На рис. 18.9 показаний приклад використання блока для затримки дискретного сигналу на 0,1 с.

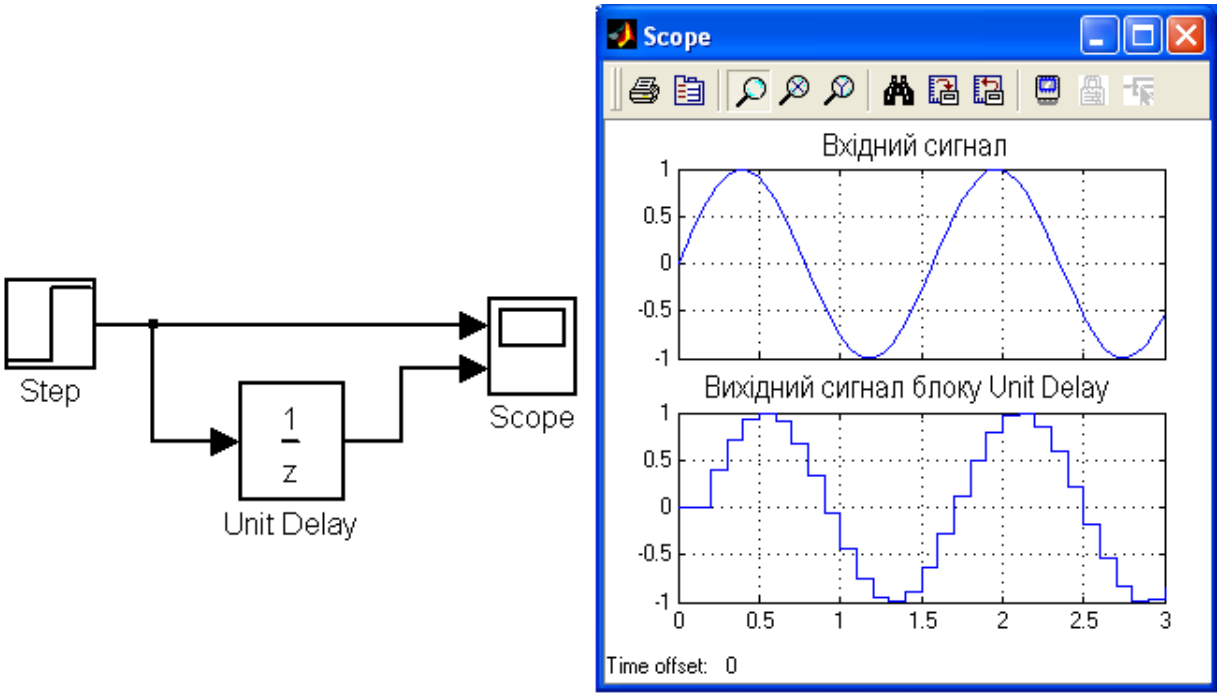


Рис. 18.9. Приклад використання блока `Unit Delay`

18.6 Блок Memory

Блок Memory (Пам'ять) (рис. 18.10) робить затримку вхідного сигналу на один такт. У принципі, подібну операцію можна виконати і за допомогою блоків затримки. Однак введення транспортного запізнювання може привести до зменшення запасів стійкості моделі системи керування, тому в деяких випадках використання блоку Memory є дуже зручним.

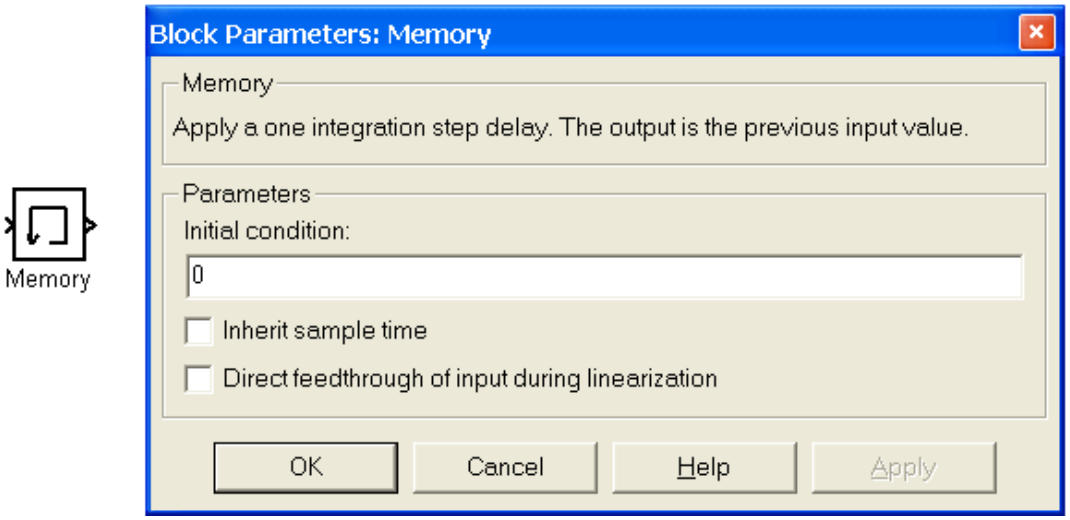


Рис. 18.10. Піктограма і діалогове вікно блока Memory

Вікно параметрів блока (рис. 18.10) містить текстове поле **Initial condition**, де задається початкове значення вихідного сигналу, а також два параметри, які визначаються прапорцями. Якщо прапорець встановлений напроти параметра **Inherit sample time**, то блок Memory використовує такий же період квантування (**Sample time**), як і у попередньому блоці. У противному випадку використовується шаг, рівний 0,1 модельного часу.

На рис. 18.11 показаний приклад роботи блока Memory для затримки дискретного сигналу. У блоці Sine Wave параметр **Sample time** встановлений на 0,4.

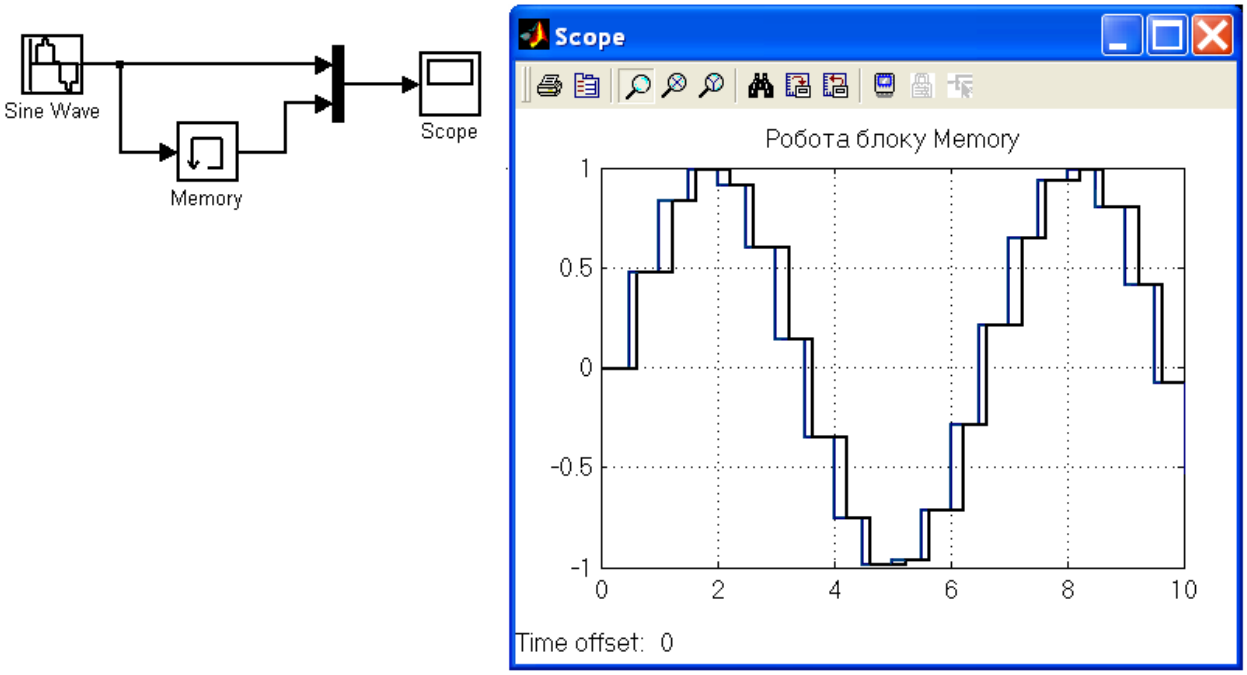


Рис. 18.11. Приклад роботи блоку `Memory`

18.7 Блок `Discrete-Time Integrator`

Операція інтегрування в дискретних системах здійснюється чисельними методами, як правило, методами прямокутників Ейлера і методом трапецій. Моделлю дискретного інтегратора є блок `Discrete-Time Integrator`. Зображення блока `Discrete-Time Integrator`, прийняте за замовчуванням, і вікно його параметрів наведені на рис. 18.12. На відміну від блока `Integrator` бібліотеки `Continuous` вікно блока `Discrete-Time Integrator` має список, що розкривається, **Integrator method** (Метод інтегрування) і текстове поле **Sample time (-1 for inherited)** (Період квантування (-1 для спадкування)).

Список, що розкривається, **Integration method** містить наступні пункти.

Forward Euler - Прямій метод Ейлера. Цей метод ще називають явним методом Ейлера або методом лівих прямокутників. Його суть ілюструє рис. 18.13. Інтеграл від функції $u(t)$ визначається різницевим рівнянням:

$$y(k) = y(k–1) + Tu(k–1), \qquad (18.6)$$

де u - вхідний сигнал інтегратора,
y - вихідний сигнал інтегратора,
T - шаг дискретизації,
k - номер шагу моделювання.

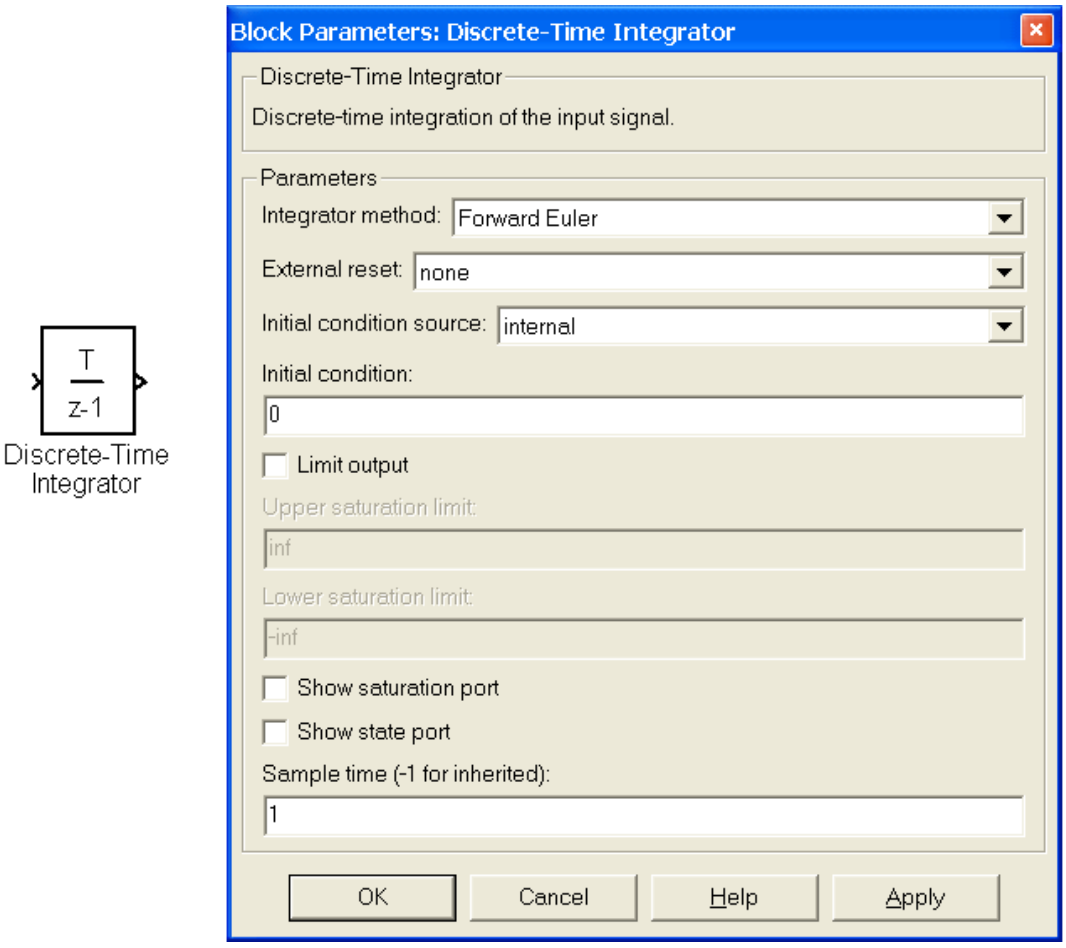


Рис. 18.12. Зображення блока `Discrete-Time Integrator` при виборі методу інтегрування **Forward Euler**

Застосувавши до рівняння (20.2) z-перетворення, при нульових початкових умовах одержимо:

$$Y(z) = Y(z)\frac{1}{z} + TU(z)\frac{1}{z}. \qquad (18.7)$$

Тоді дискретна передаточна функція інтегратора, що реалізує прямий метод Ейлера, має такий вигляд:

$$W(z) = \frac{T}{z-1}, \qquad (18.8)$$

що й відображається усередині піктограми блока `Discrete-Time Integrator` (рис. 18.12).

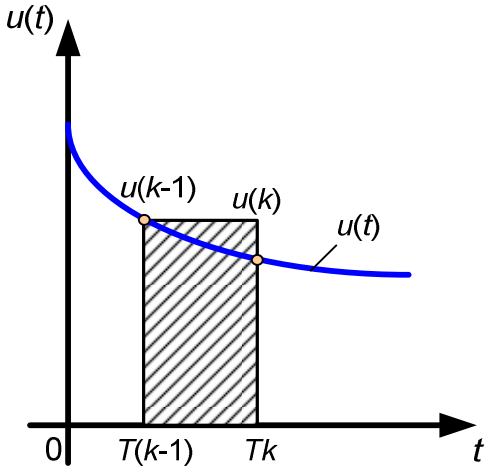


Рис. 18.13. Чисельне інтегрування прямим методом Ейлера

Backward Euler - зворотний метод Ейлера (неявний метод або метод правих прямокутників). Цей метод чисельного інтегрування ілюструє рис. 18.14. Вихідний сигнал блока розраховується за вираженням:

$$y(k) = y(k–1) + Tu(k), \tag{18.9}$$

а відповідна дискретна передаточна функція має вигляд:

$$W(z) = \frac{Tz}{z-1}. \tag{18.10}$$

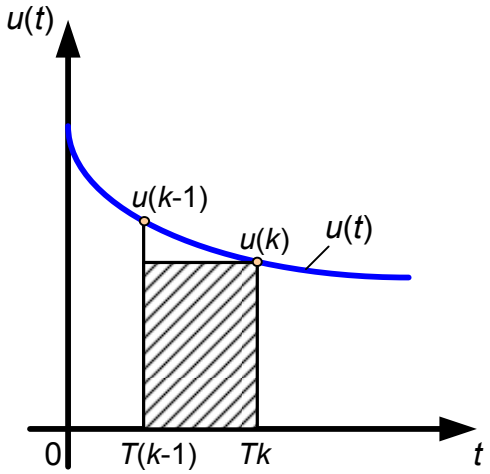
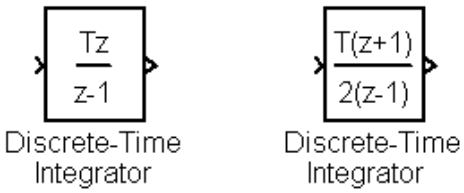


Рис. 7.14. Чисельне інтегрування зворотним методом Ейлера

При цьому передаточна функція виду (18.10) з'являється усередині піктограми блока (рис. 18.15, *а*).



а *б*

Рис. 18.15. Зображення блока `Discrete-Time Integrator` при виборі методу інтегрування **Backward Euler** (*а*) і **Trapezoidal** (*б*)

Trapezoidal - метод трапецій. При виборі цього методу чисельного інтегрування фігура під кривій $u(t)$ представляється не прямокутником, а трапецією (рис. 18.16).

Основи трапеції визначаються значеннями безперервного сигналу в моменти квантування $T(k\text{-}1)$ і Tk. Площа трапеції визначається як сума основ, помножена на половину висоти. Тому метод трапецій описується рівнянням:

$$y(k) = y(k-1) + \frac{[u(k) + u(k-1)]T}{2}, \qquad (18.11)$$

а дискретна передаточна функція записується у вигляді:

$$W(z) = \frac{T(z+1)}{2(z-1)}. \qquad (18.12)$$

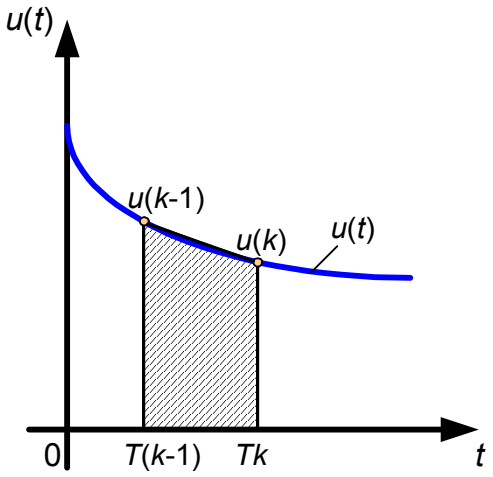


Рис. 18.16. Чисельне інтегрування методом трапеції

Зовнішній вигляд блока `Discrete-Time Integrator` при виборі в поле **Integration method** методу інтегрування **Trapezoidal** наведений на рис. 18.15, *б*.

Параметр **Sample time**, як і у попередньому блоці, встановлює значення періоду квантування *T*.

Інші параметри дискретного інтегратора такі ж, що й у блоці аналогового інтегратора `Integrator` бібліотеки `Continuous`.

На рис. 18.17 показаний приклад, що демонструє всі три способи чисельного інтегрування блока `Discrete-Time Integrator`.

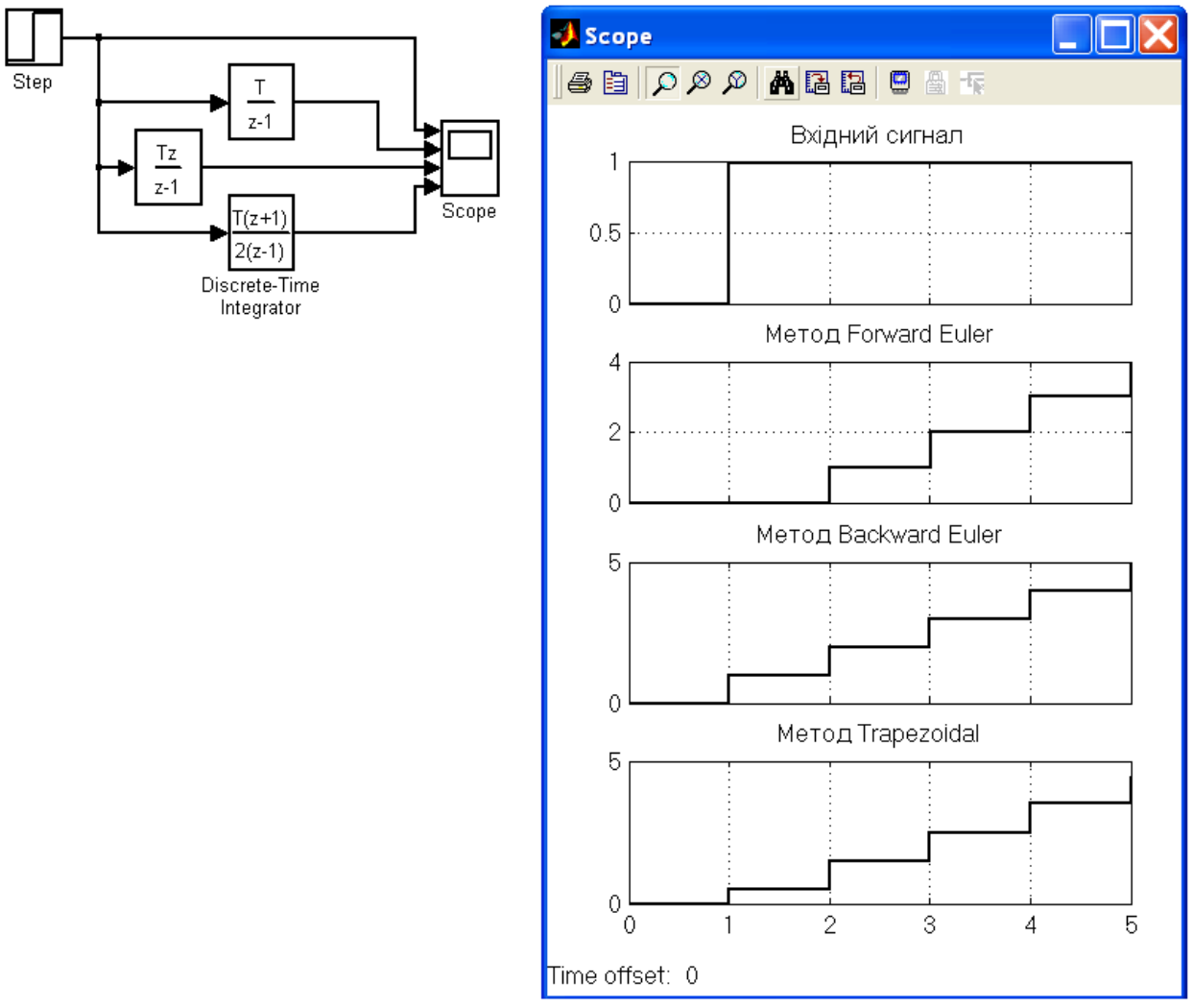


Рис. 18.17. Ілюстрація інтегрування вхідного сигналу блока `Discrete-Time Integrator` різними методами

18.8 Блок `Zero-Order Hold`

Для відновлення безперервного сигналу з дискретного застосовуються екстраполятори (див. главу 12). Екстраполяторами ці пристрої називаються тому, що вони, на підставі інформації про попередні значення дискретного сигналу *прогнозують значення* сигналу протягом чергового інтервалу квантування, тобто виконують екстраполяцію.

Найпростішим екстраполятором є екстраполятор нульового порядку. Його вихідний сигнал протягом усього періоду квантування зберігає постійне значення, яке дорівнює значенню вихідного безперервного сигналу в момент квантування.

В Simulink екстраполятор нульового порядку моделюється блоком `Zero-Order Hold` (рис. 18.18).

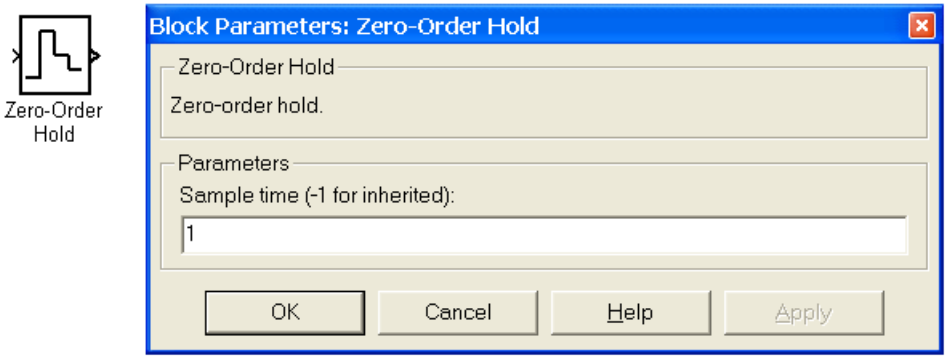


Рис. 18.18. Зображення і вікно параметрів блока `Zero-Order Hold`

Вікно параметрів блока (рис. 18.18) має всього лише одне поле **Sample time (-1 for inherited)**, у якому вказується період квантування.

На рис. 18.19 показаний приклад використання блока `Zero-Order Hold` для формування сигналу з періодом квантування 0,5 с.

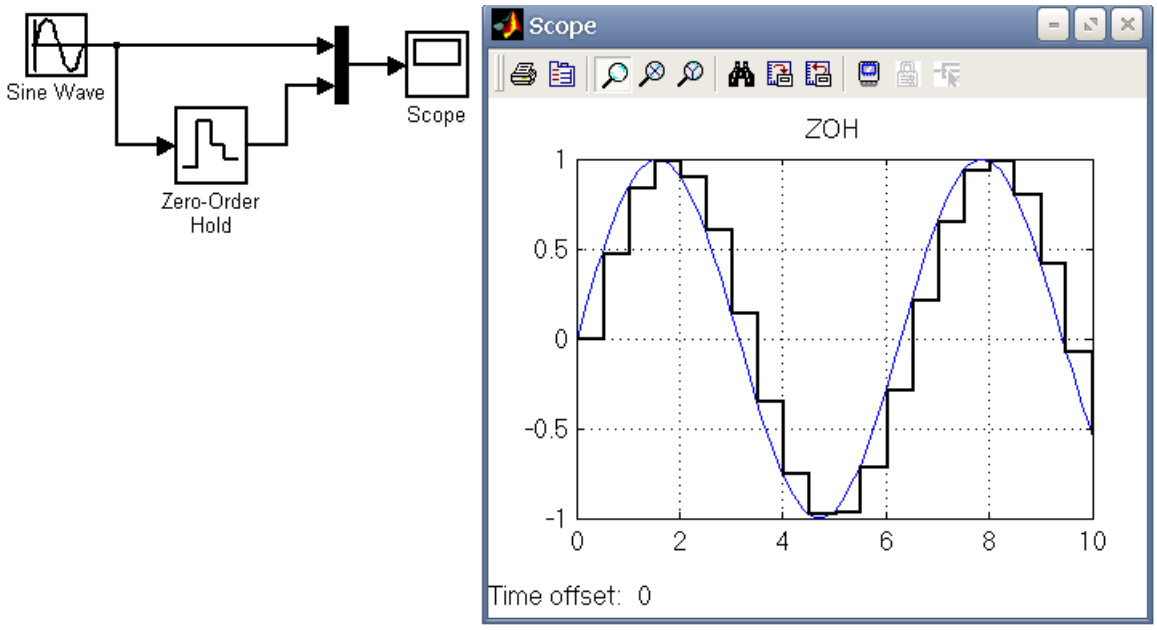


Рис. 18.19. Приклад роботи блока `Zero-Order Hold`

18.9 Блок `First-Order Hold`

Блок `First-Order Hold` (рис. 18.20) моделює екстраполятор першого порядку. Вікно параметрів блока аналогічно блока `Zero-Order Hold`.

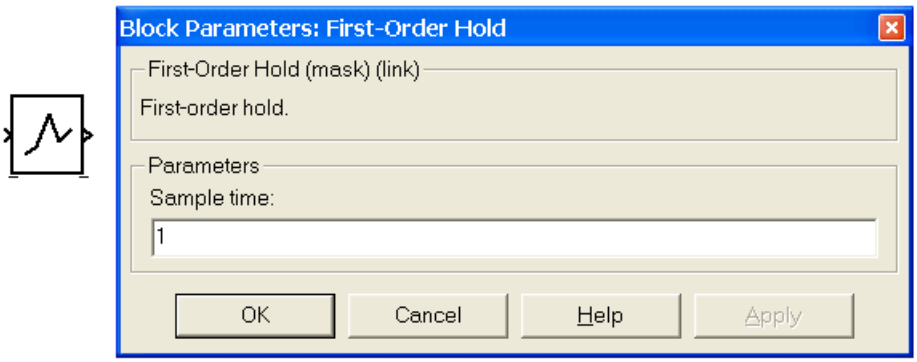


Рис. 18.20. Зображення і вікно параметрів блока `First-Order Hold`

На рис. 18.21 показаний приклад екстраполяції цим блоком синусоїдального сигналу; тут період квантування обраний рівним 0,5 с.

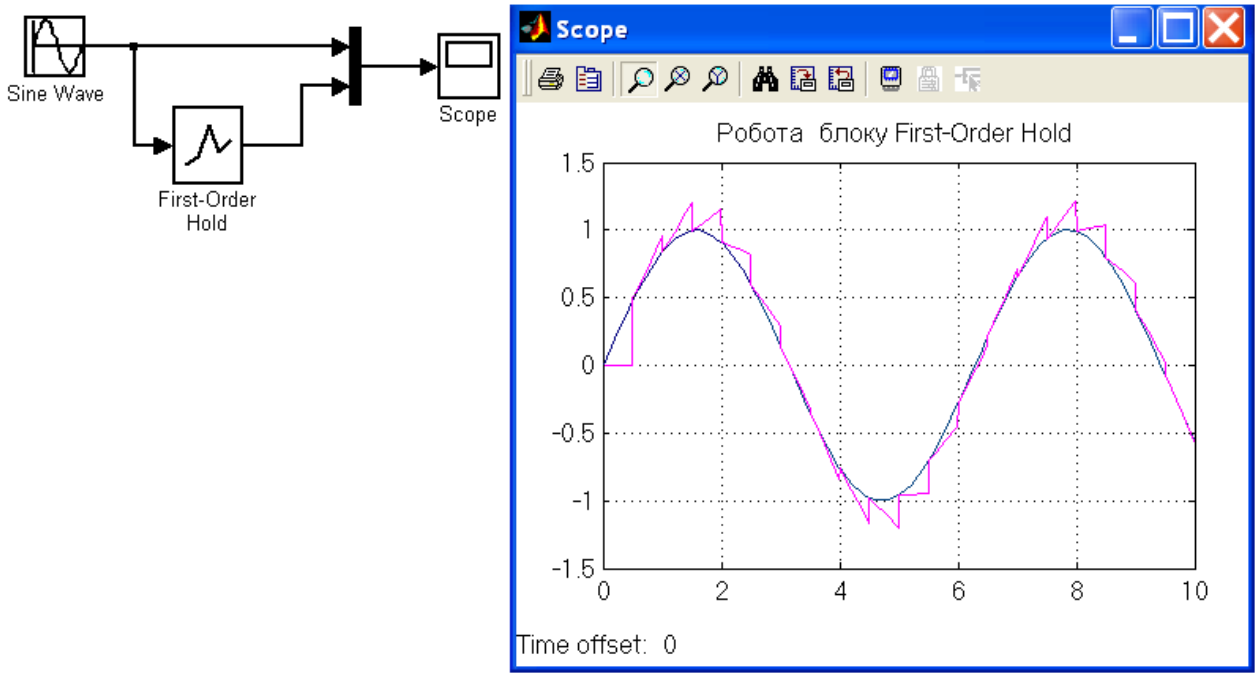


Рис. 18.21. Використання блока `First-Order Hold`

18.10 Приклад моделювання дискретної системи

Розглянемо систему з такою ж незмінною частиною, як і на рис. 15.29, але з цифровим, а не аналоговим ПІД-регулятором. Один з найпоширеніших на практиці методів розрахунку цифрових регуляторів полягає в тім, що об'єкт керування вважається безперервним і синтез регулятора проводиться методами, відомими з теорії безперервних систем. Отриманий при цьому безперервний регулятор апроксимується цифровим.

У п. 15.10 були запропоновані такі коефіцієнти ПІД-регулятора: $K_p = 3{,}7$, $K_i = 0{,}9$ і $K_d = 2$. Застосуємо ці ж коефіцієнти й у цифровому ПІД-регуляторі, а період квантування T приймемо рівним 0,1 с. Одержати модель цифрового ПІД-регулятора можна за допомогою команд Control System Toolbox, при цьому для відновлення даних будемо використовувати екстраполятор нульового порядку:

```
>> Wi=tf(0.9,[1 0]);%Передаточна функція
 %інтегрального каналу безперервного регулятора;
>> Wd=tf([2 0],[0.1 1]); %Передаточна функція
 %диференціючого каналу безперервного регулятора;
>> Wiz=c2d(Wi,0.1,'zoh');
```

```
>> Wdz=c2d(Wd,0.1,'zoh');
>> Wiz

Transfer function:
0.09
-----
z - 1

Sampling time: 0.1
>> Wdz

Transfer function:
20 z - 20
----------
z - 0.3679

Sampling time: 0.1
```

Simulink-модель системи керування із цифровим ПІД-регулятором представлена на рис. 18.22. В ній з метою запобігання небажаного ефекту розгойдування системи в момент стрибкоподібної зміни сигналу помилки застосований блок `Saturation`. Параметри блока `Saturation`: **Upper limit:** 0.01, **Lower limit:** -1.

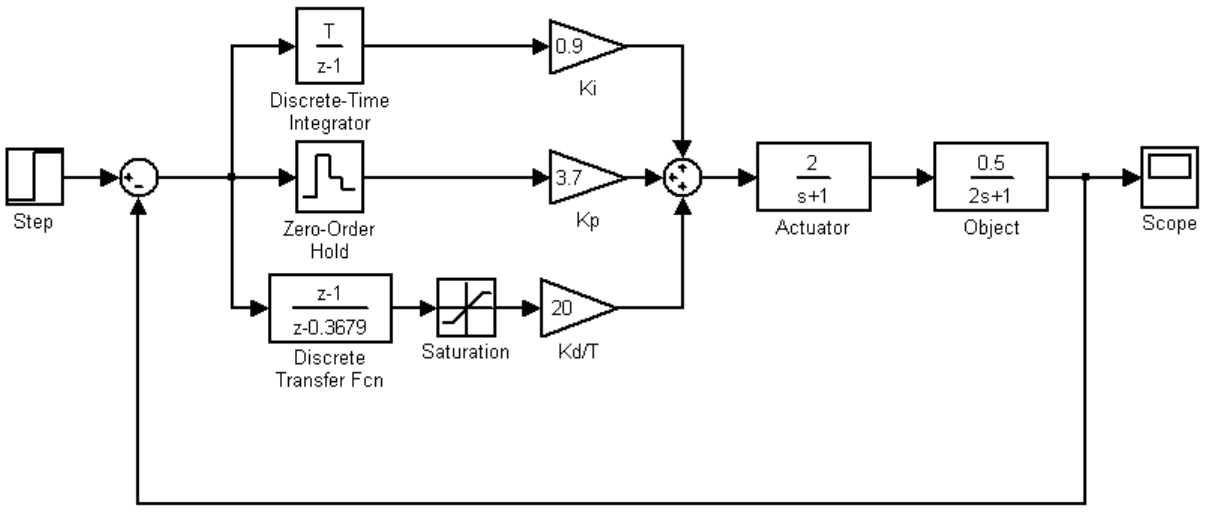


Рис. 18.22. Система керування із цифровим ПІД-регулятором

Перехідна функція системи з таким регулятором представлена на рис. 18.23.

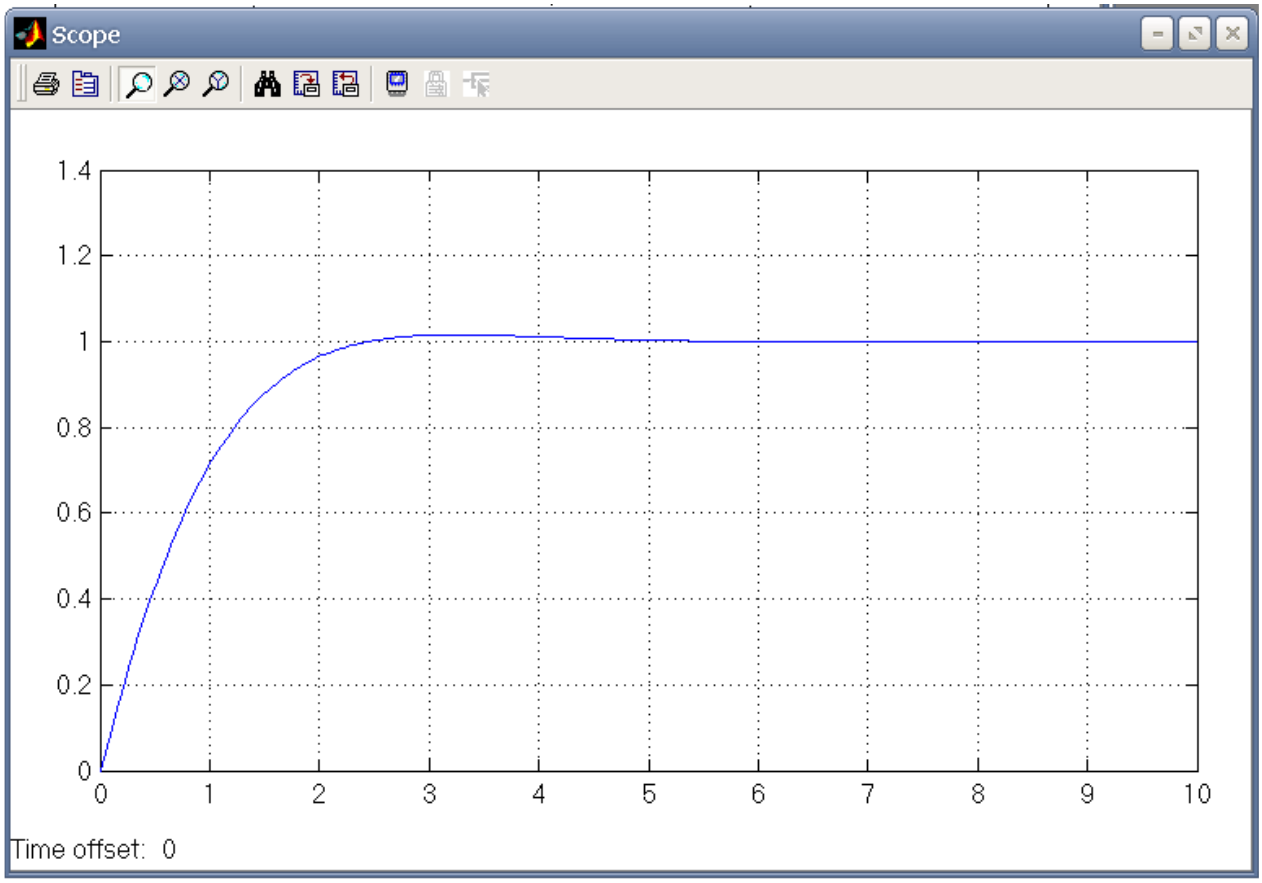


Рис. 18.23. Перехідна функція системи із цифровим ПІД-регулятором

Як бачимо, якість цифрової системи трохи гірше, ніж у безперервної (див. рис. 15.26). Це пояснюється втратою інформації при квантуванні та погрішностями чисельного інтегрування і диференціювання. Зі зменшенням періоду квантування варто очікувати поліпшення якості регулювання за рахунок підвищення точності апроксимації безперервного регулятора. Однак це вимагає використання мікропроцесорів з підвищеною швидкодією, що приводить до подорожчання системи.

Завдання для самостійної роботи

1. Безперервний вхідний сигнал $x(t)$ піддається квантуванню, а потім відновленню. Побудуйте в одному графічному вікні графіки сигналу $x(t)$ та виходу екстраполятору при двох значеннях періоду квантування $T = 0{,}2$ с та $T = 0{,}5$ с. У якості екстраполяторів використайте блоки `Zero-Order Hold` та `First-Order Hold`. Зробить висновок щодо доцільності використання того чи іншого

способу відновлення для кожного з безперервних сигналів. (Вказівка: для формування вектору часу *t* зручно використати блок `Clock` з бібліотеки `Sources`).

а) $x(t) = e^{-t}$; *б)* $x(t) = t$; *в)* $x(t) = \sin t$. *г)* $x(t) = t^2$.

2. Дані передаточні функції безперервних систем. Побудуйте у вікні одного блоку `Scope` перехідні функції безперервних систем і відповідних їм дискретних систем при періоді квантування $T = 0{,}1$ с. Перехід від безперервної моделі до дискретної можна виконувати за допомогою команди `c2d`. Змінюючи значення T зробіть висновок щодо впливу періоду квантування на властивості системи.

а) $W(s) = \dfrac{1}{s(0{,}1s+1)}$; *б)* $W(s) = \dfrac{1}{s(s+1)^2}$;

в) $W(s) = \dfrac{s+0{,}2}{(s+1)(s+2)}$; *г)* $W(s) = \dfrac{1}{s+1}$;

д) $W(s) = \dfrac{2}{s^2+0{,}5s+1}$; *е)* $W(s) = \dfrac{s}{s^2+4}$.

3. Переведіть дискретні передаточні функції, що були одержані в завданні 1 в дискретний простір станів. Перевірте результат за допомогою Simulink.

4. Дискретна система з періодом квантування $T = 0{,}5$с в розімкненому виді має наступну передаточну функцію

$$W(z) = K\frac{0{,}1065z + 0{,}0902}{z^2 - 1{,}6065z + 0{,}6065}.$$

За допомогою блоку `Slider Gain` визначте діапазон значень коефіцієнту підсилення K, при яких замкнена система буде стійкою.

5. Об'єкт управління безперервної системи описується передаточною функцією

$$W_o(s) = \frac{50}{(0,2s+1)(0,5s+1)(s+1)}.$$

Аналоговий ПІ-регулятор, що коректує роботу замкненої системи з цим об'єктом має вигляд

$$W_{ПІ}(s) = 0,68 + \frac{0,11}{s}.$$

а) Визначите еквівалентну передаточну функцію дискретного ПІ-регулятора при $T = 0,1$ с.

б) У одному блоці `Scope` побудуйте перехідні функції безперервної та дискретної систем і зробіть висновок щодо впливу квантування на роботу системи.

19. СТВОРЕННЯ ПІДСИСТЕМ

Модель системи керування може мати складну структуру, що істотно утруднює роботу. Для більш наочного подання структури моделі її представляють у вигляді набору окремих блоків користувача, не показуючи їхнього вмісту. Такі блоки називають *підсистемами*. Використання підсистем зменшує кількість блоків, які відображаються на екрані, полегшує налагодження і редагування моделі, підвищує її інформативність.

19.1 Формування підсистеми

Роль підсистем і процес їхнього створення розглянемо на прикладі системи керування з ПІД-регулятором, представленої на рис. 19.1. У диференціюючому каналі ПІД-регулятора використовується реальна диференціююча ланка. Це дозволяє уникнути флуктуації керування, викликаної поміхами виміру вихідної величини і посиленої операцією диференціювання, а також зменшить амплітуду імпульсів на виході регулятора при стрибкоподібній зміні сигналу помилки.

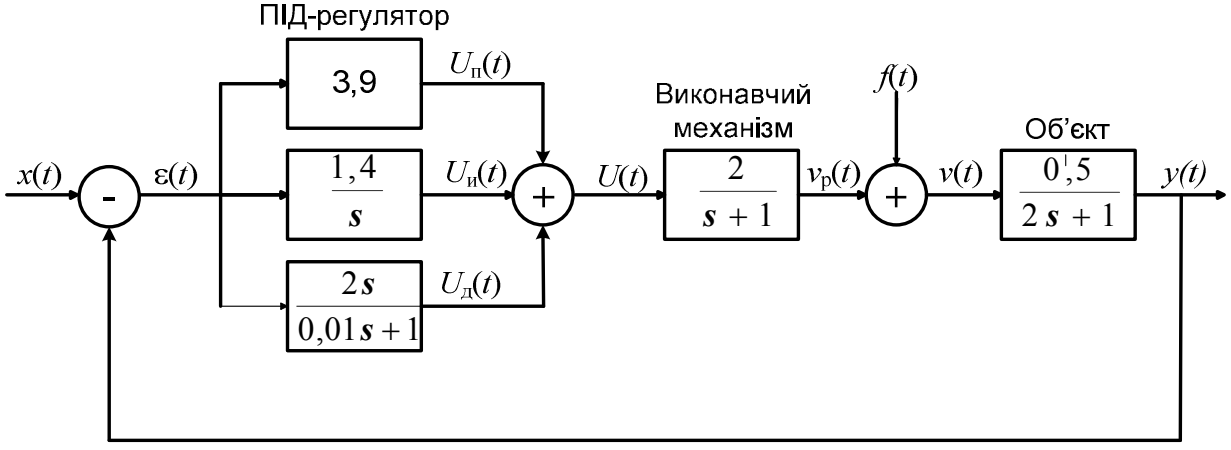


Рис. 19.1. Структура системи керування

Simulink-модель цієї системи керування наведена на рис. 19.2. Тут у якості джерела зовнішнього збурювання прийнятий блок `Step`. Приймемо, для простоти, $f(t) = 0$. Модель ПІД-регулятора зібрана за допомогою блоків з бібліотеки `Continuous`. У

розглянутій моделі можна виділити дві основні частини: регулятор і незмінна частина, що складається з виконавчого механізму та об'єкта керування. Представимо ці дві частини моделі у вигляді підсистем.

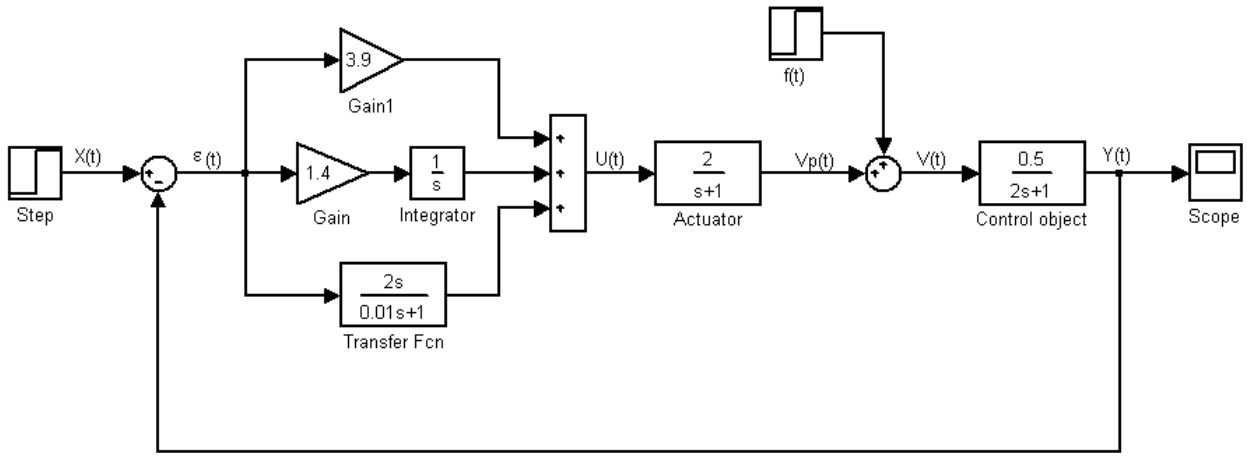


Рис. 19.2. Simulink-модель системи керування,
представленої на рис. 19.1

Для створення підсистеми необхідно спочатку виділити за допомогою миші з натиснутою лівою кнопкою всі блоки і лінії зв'язку, призначені для включення в підсистему. На рис. 19.3 показаний приклад виділення блоків, що входять у ПІД-регулятор. Іноді попередньо потрібно пересунути деякі блоки так, щоб зайві блоки не були виділені.

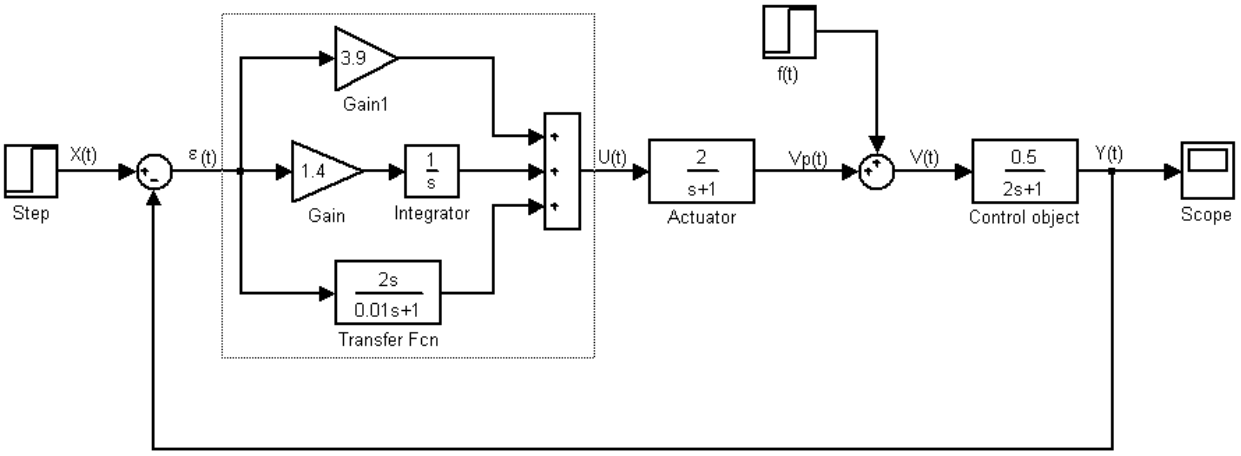


Рис. 19.3. Виділення блоків, що входять у підсистему

Тепер безпосередньо формуємо підсистему. Це можна зробити декількома способами.

Перший спосіб полягає у використанні команди **Create subsystem** меню **Edit** вікна моделі (рис. 19.4). Цю же команду можна вибрати в контекстному меню, що викликається щигликом правої кнопки миші на виділеній частині системи.

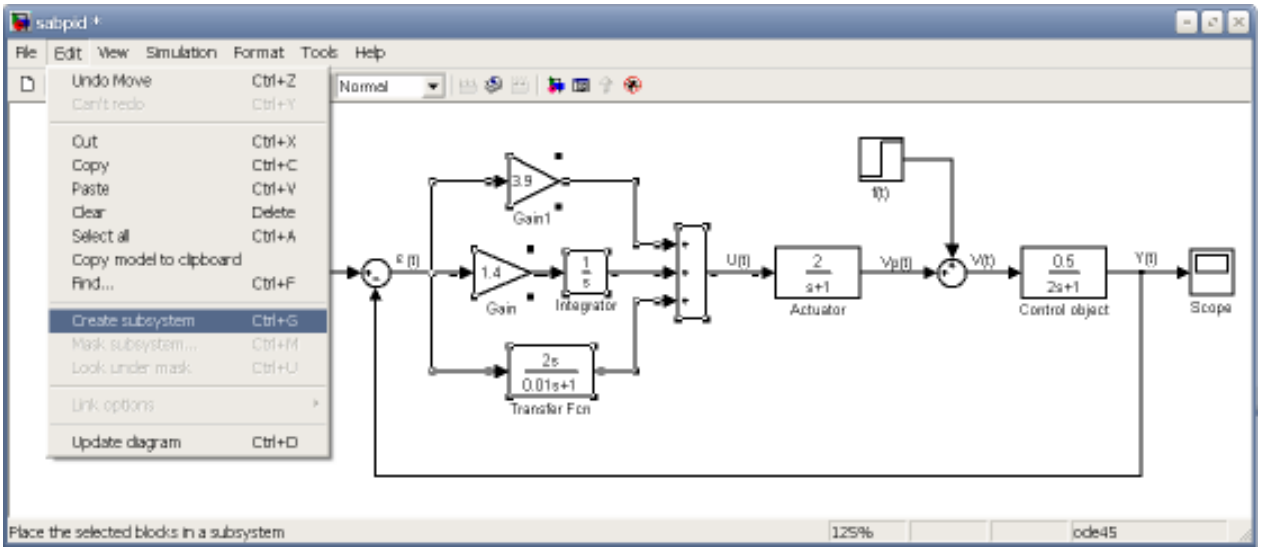


Рис. 19.4. Створення підсистеми з меню **Edit**

Другий спосіб – натискання комбінації клавіш Ctrl+G.

Структурна схема системи керування прийме вид представлений на рис. 19.5. На місці виділених блоків з'явився блок `Subsystem` із вхідним портом `In1` і вихідним портом `Out1`. У загальному випадку вхідних і вихідних портів стільки ж, скільки вхідних і вихідній змінних у підсистемі.

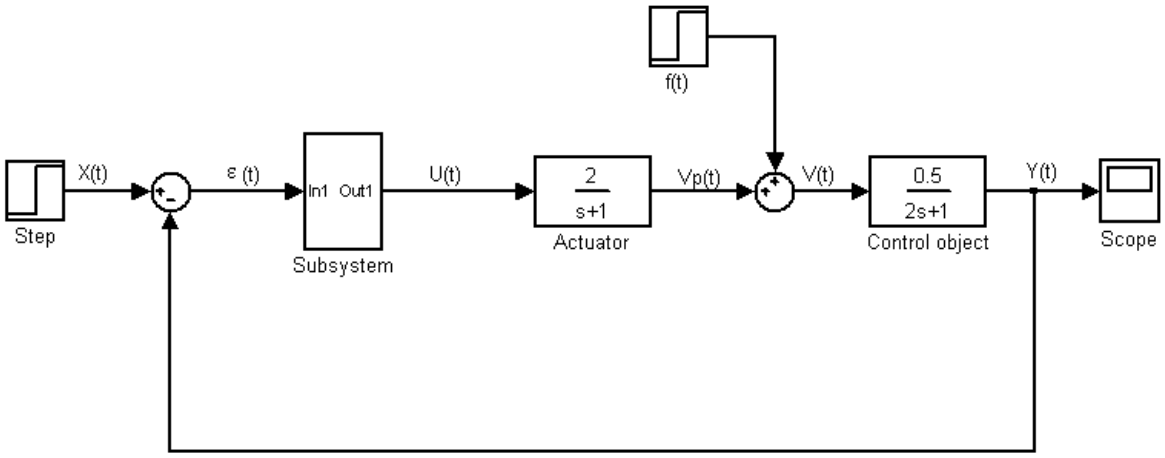


Рис. 19.5. ПІД-регулятор у вигляді підсистеми

Якщо розкрити вікно блока `Subsystem`, двічі клацнувши на ньому, то Simulink відобразить блок-схему створеної підсистеми (рис. 19.6). Як видно усередині підсистеми з'явилися блоки `In` та `Out` для відображення входів і виходів у систему вищого рівня.

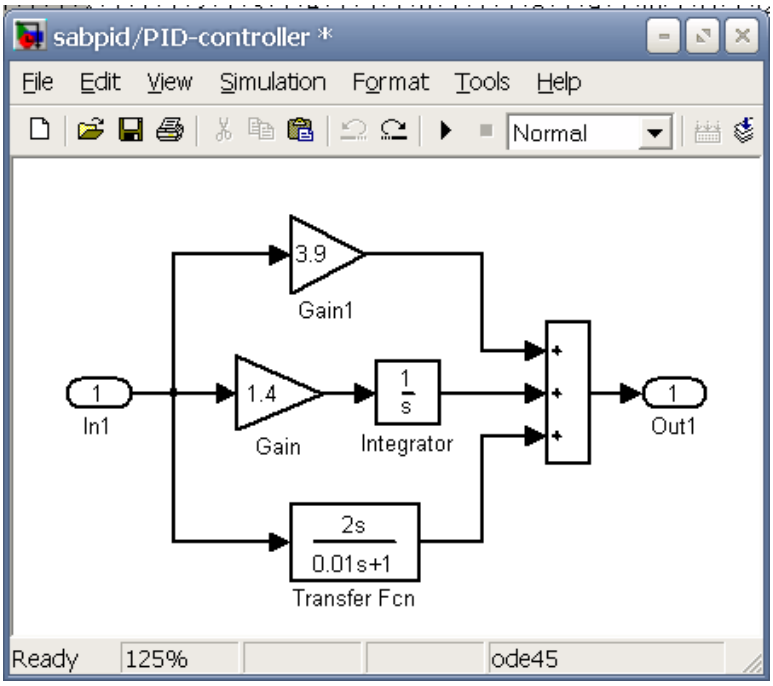


Рис. 19.6. Перегляд вмісту підсистеми

Третій спосіб створення підсистеми - використання блока `Subsystem` (Підсистема) з бібліотеки блоків `Ports & Subsystems`. Цей спосіб зручний при створенні підсистеми безпосередньо в процесі побудови моделі.

Для створення підсистеми з використанням блока `Subsystem` потрібно перенести його у вікно моделі й двічі натиснути ліву кнопку миші на зображенні цього блока. У результаті відкриється вікно підсистеми (рис. 19.7).

Далі необхідно сформувати структурну схему підсистеми. Всі вхідні в підсистему лінії зв'язку повинні бути з'єднані з вихідними портами блоків `In`, а всі вихідні з підсистеми лінії зв'язку повинні бути пов'язані із вхідними портами блоків `Out`.

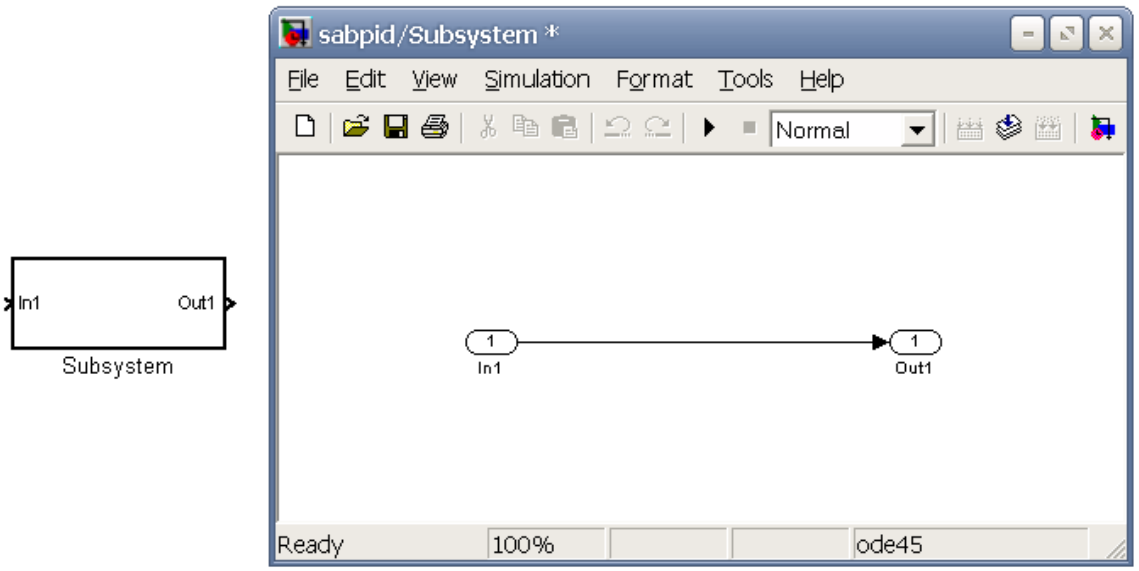


Рис. 19.7. Блок `Subsystem` з бібліотеки `Ports & Subsystems`

Сформуємо тепер підсистему з незмінною частиною (рис. 19.8). Як бачимо, вона має два вхідних порти – `In1` для зовнішнього збурювання f(t) і `In2` для керуючого впливу від регулятора $U(t)$.

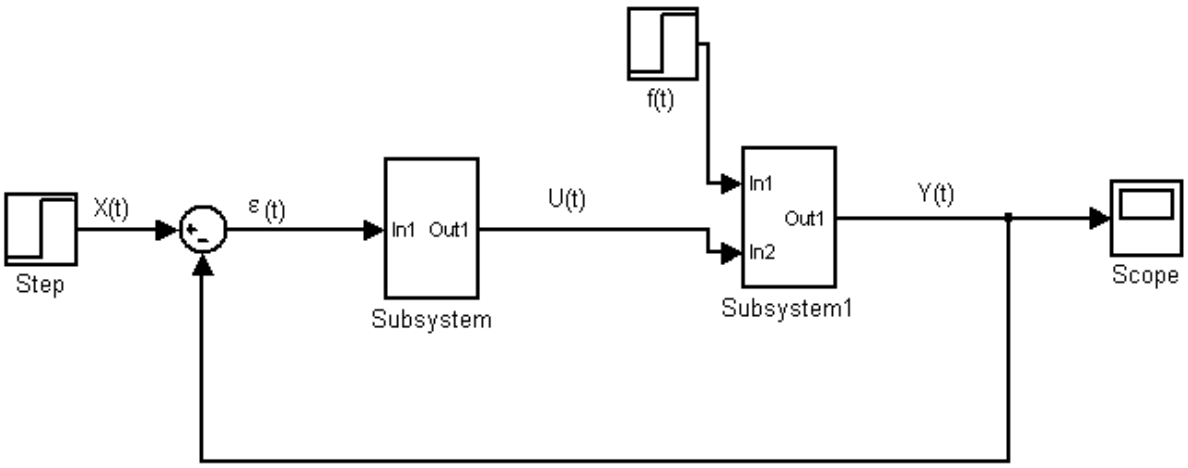


Рис. 19.8. Регулятор і незмінна частина у вигляді підсистем

Тепер перейдемо до оформлення моделі. Отримані підсистеми необхідно підписати, кожну залежно від її призначення, так як ім'я `Subsystem` про призначення даного блока нічого не говорить. Бажано також перейменувати входи `In` і виходи `Out` підсистем, щоб мати уявлення про вхідні та вихідні змінні. Нагадаємо, що увійти в

режим редагування імені блока можна, клацнувши лівою кнопкою миші на старому імені. Остаточний вид моделі системи керування представлений на рис. 19.9.

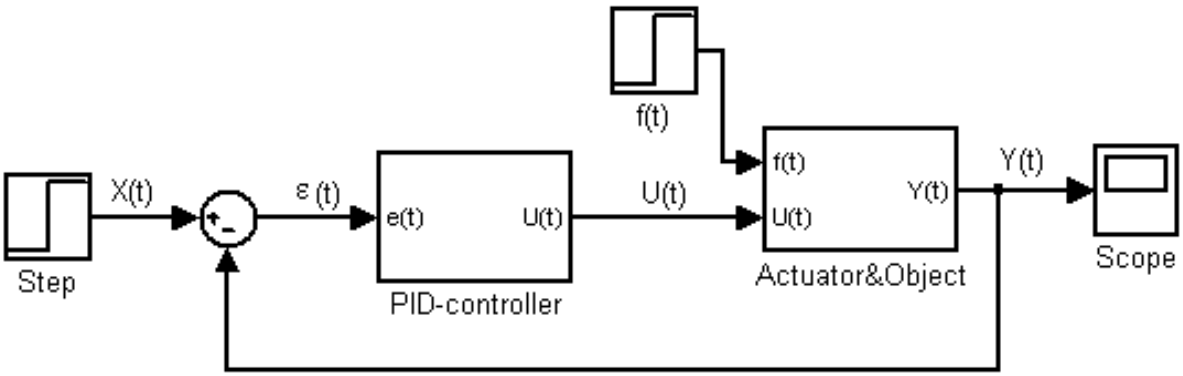


Рис. 19.9. Структурна схема з двома підсистемами

19.2 Маскування підсисеми

19.2.1 Загальні відомості

Simulink представляє користувачеві можливість оформляти підсистеми як звичайні бібліотечні блоки. Ця операція називається *маскуванням*. Маскованому блоку можна задати своє графічне зображення, сформувати діалогове вікно, у якому можна вказувати параметри блока, створити довідкову інформацію про його призначення і роботу. Таким чином, маскування забезпечує гнучкість при роботі з моделлю, спрощує побудову складних моделей, підвищує наочність моделі, охороняє підсистему від несанкціонованої зміни. До речі, блоки з бібліотеки `Simulink Extras` являють собою масковані підсистеми.

Для того щоб маскувати підсистему її необхідно виділити та вибрати команду **Mask subsystem** з меню **Edit** у вікні моделі або з контекстного меню, що викликається щигликом правої кнопки миші на блоці підсистеми. Можна також натиснути комбінацію клавіш <Ctrl+M>. При цьому з'явиться вікно редактора маски **Mask Editor** (рис. 19.10).

Вікно **Mask Editor** має чотири вкладки: **Icon**, **Parameters**, **Initialization** і **Documentation**. Зупинимося на призначенні кожної з цих вкладок.

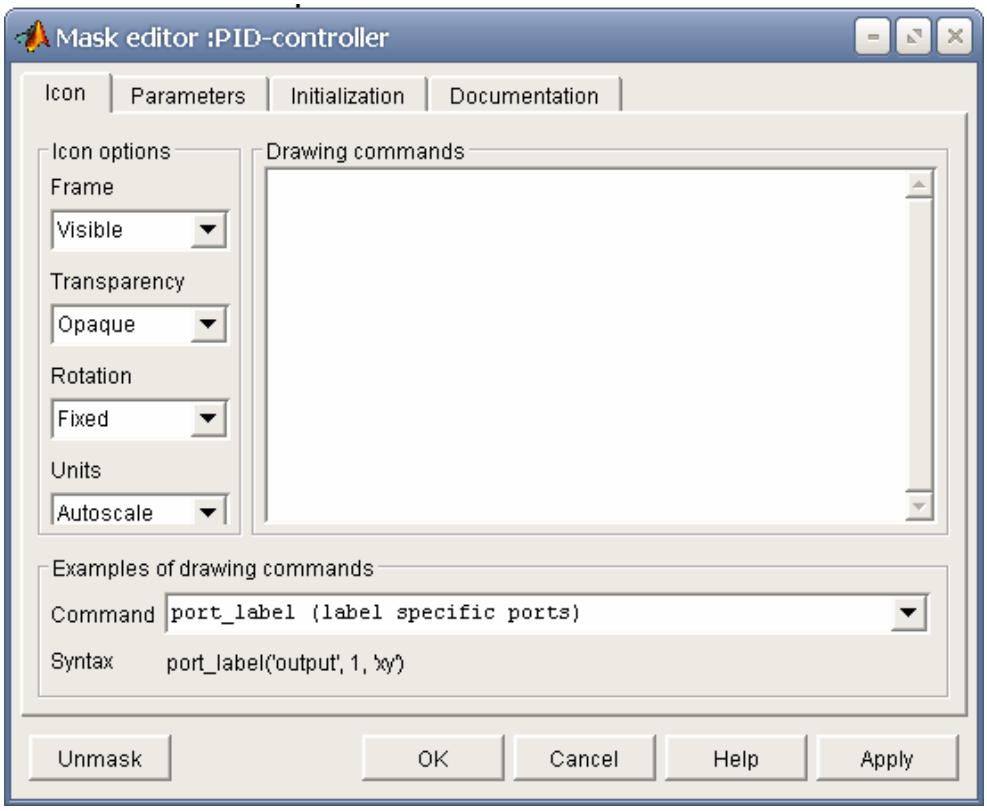


Рис. 19.10. Вікно **Mask Editor** для підсистеми `PID-controller`

19.2.2 Створення графічного зображення підсистеми

Перша вкладка вікна **Mask Editor**, відкрита за замовчуванням, – вкладка **Icon** (*англ. іконка, зображення*). Як слід з назви, тут формується зображення маскованого блока. Це зображення, або інакше, іконка, піктограма, може містити текст, вираження, рисунок або графік.

Вкладка **Icon** розділена на три частини. Більшу частину вкладки займає поле **Drawing commands**, у якому за допомогою команд мовою MATLAB формується графічне зображення блока. Приклади написання деяких із цих команд приводяться в полі **Examples of drawing commands**. Тут у списку **Command**, що розкривається, дані назви команд і у дужках їхнє призначення, а нижче приводиться приклад синтаксису команди. При цьому можливий вид зображення блока відображається в правому нижньому куті вкладки **Icon**. На рис. 19.11 показано, як відобразити передаточну функцію ПІД-регулятора на блоці підсистеми.

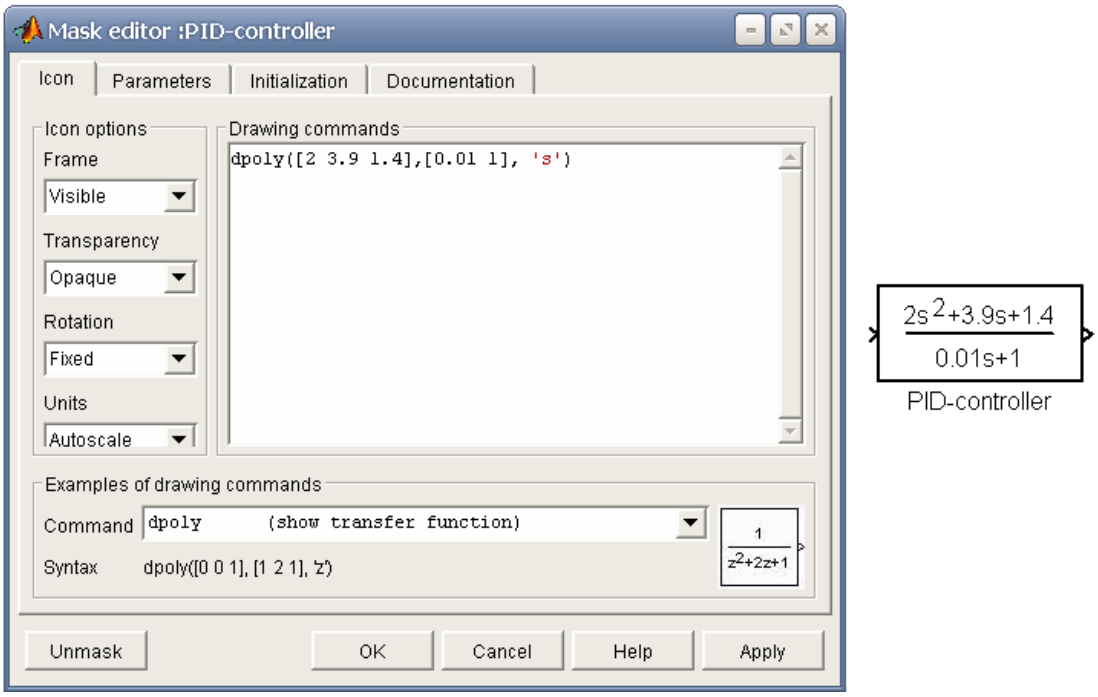


Рис. 19.11. Відображення передаточної функції на блоці підсистеми

Дуже зручною є можливість відображення на блоці підсистеми рисунка. Це істотно підвищує наочність моделі.

Для відображення на блоці рисунка, що міститься у файлі, використовується команда

```
image(imread('ім'я_файлу'))
```

де `ім'я_файлу` – ім'я графічного файлу з розширенням.

Для коректної роботи цієї команди файл із рисунком повинен знаходитися в одній папці з файлом моделі, у противному випадку необхідно вказати повний шлях до нього. На рис. 19.12 показаний приклад відображення рисунка на блоці підсистеми.

Для створення рисунка на блоці можна також використати спеціальний редактор, що викликається наступною командою в робочому рядку MATLAB:

```
>>iconedit('ім'я_моделі','ім'я_підсистеми')
```

Наприклад,

```
>> iconedit('subpid','PID-controller')
```

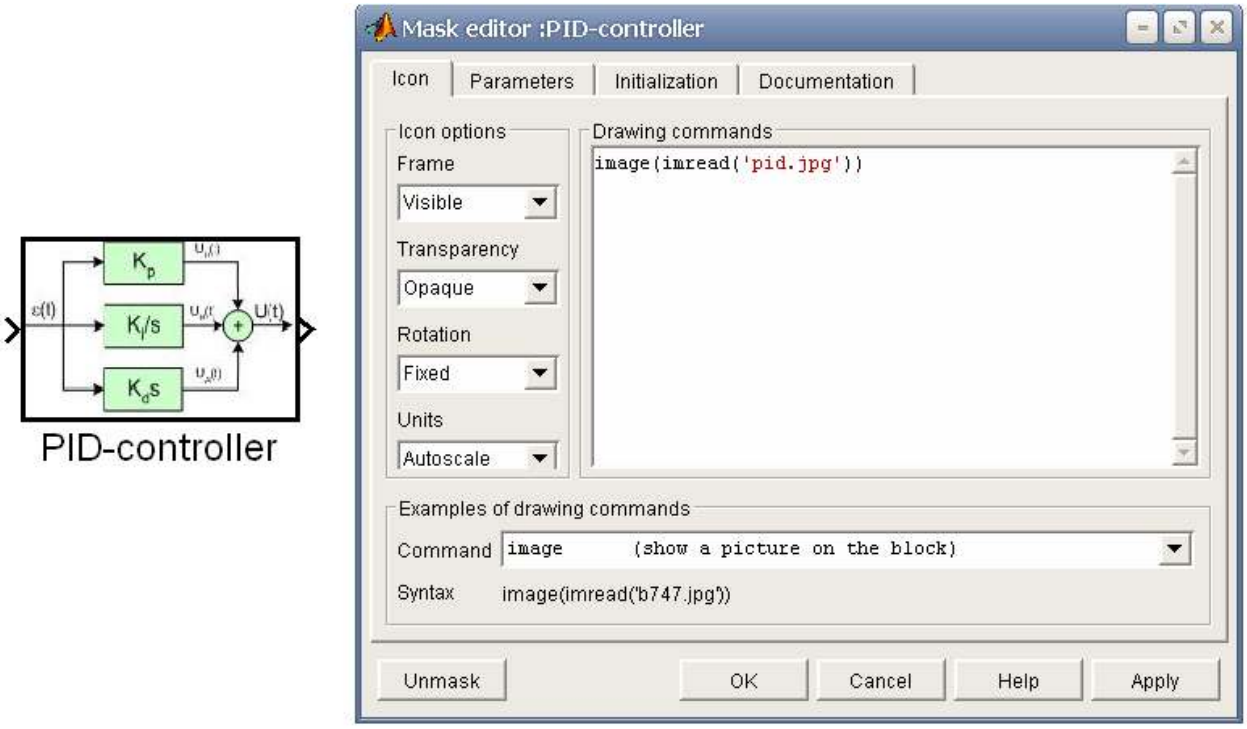


Рис. 19.12. Відображення рисунка на блоці підсистеми

У результаті з'являється графічне вікно **Block Icon Editor** (рис. 19.13) у якому можна створити рисунок для підсистеми.

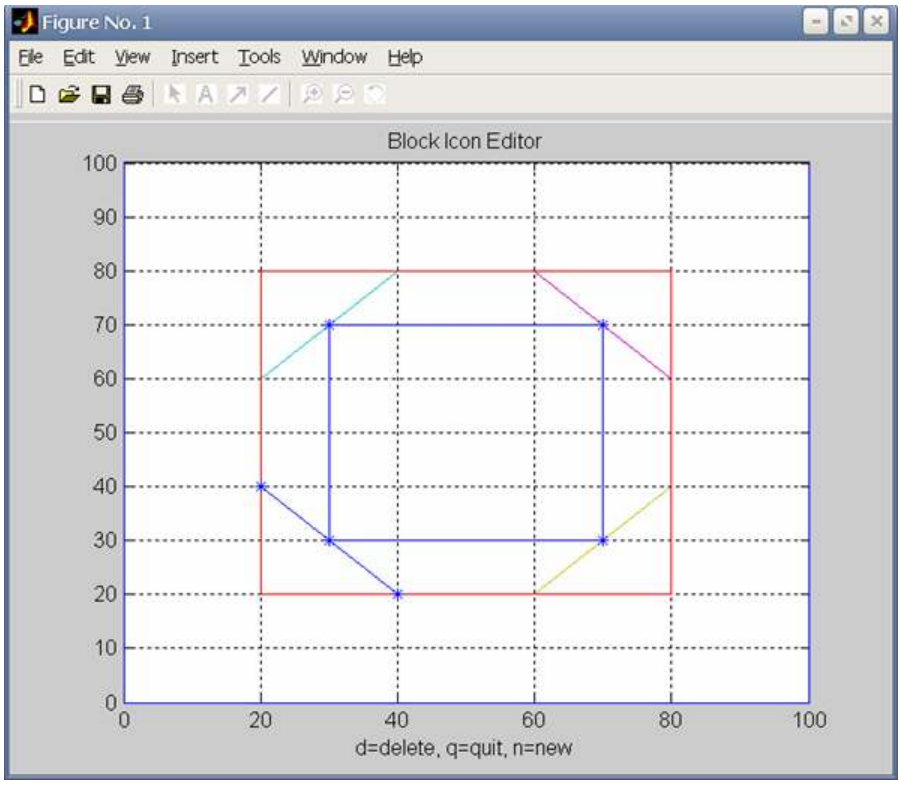


Рис. 19.13. Вікно **Block Icon Editor**

Рисунок створюється по точках, розташування яких вказується за допомогою миші. Між собою точки з'єднуються прямими лініями. Для того щоб почати нову лінію необхідно натиснути клавішу **n** на клавіатурі. Для скасування створення останньої точки використовується клавіша **d**. Вихід з режиму рисування здійснюється клавішею **q**.

Після виходу з редактора рисунок у блоці підсистеми обновляється (рис. 19.14), а в полі **Drawing commands** і у командному рядку MATLAB виводиться команда, що забезпечує побудову рисунка.

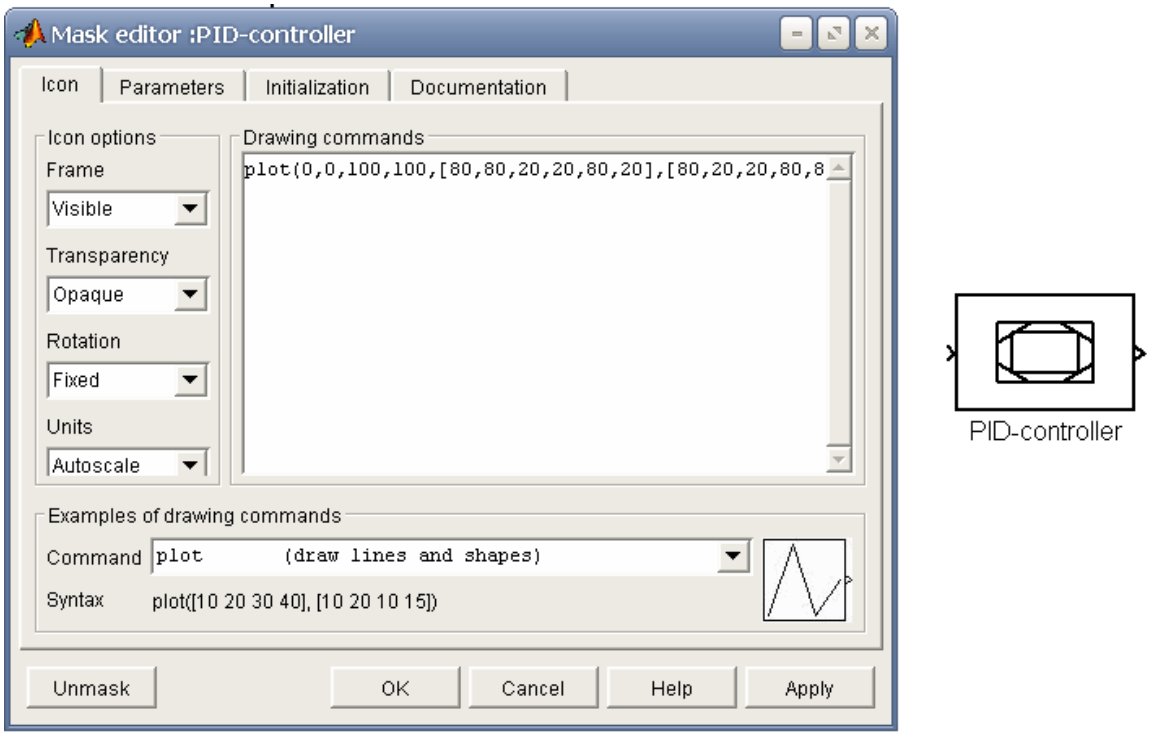


Рис. 19.14. Вид блока зі створеним рисунком і поле **Drawing commands** с відповідною командою

Наприклад, для фігури, зображеної на рис. 19.13:

```
ans =

>>plot(0,0,100,100,[20,20,80,80,20],...
[80,20,20,80,80],[40,20],[80,60],[80,60],...
[60,80],[60,80],[20,40],[20,40],[40,20],...
[30,30,70,70,30],[70,30,30,70,70])
```

На зображенні блока підсистеми можна виводити і текст, наприклад, за допомогою команди `disp('текст')`. Так на рис. 19.15 показані команди виводу тексту в центрі блока підсистеми.

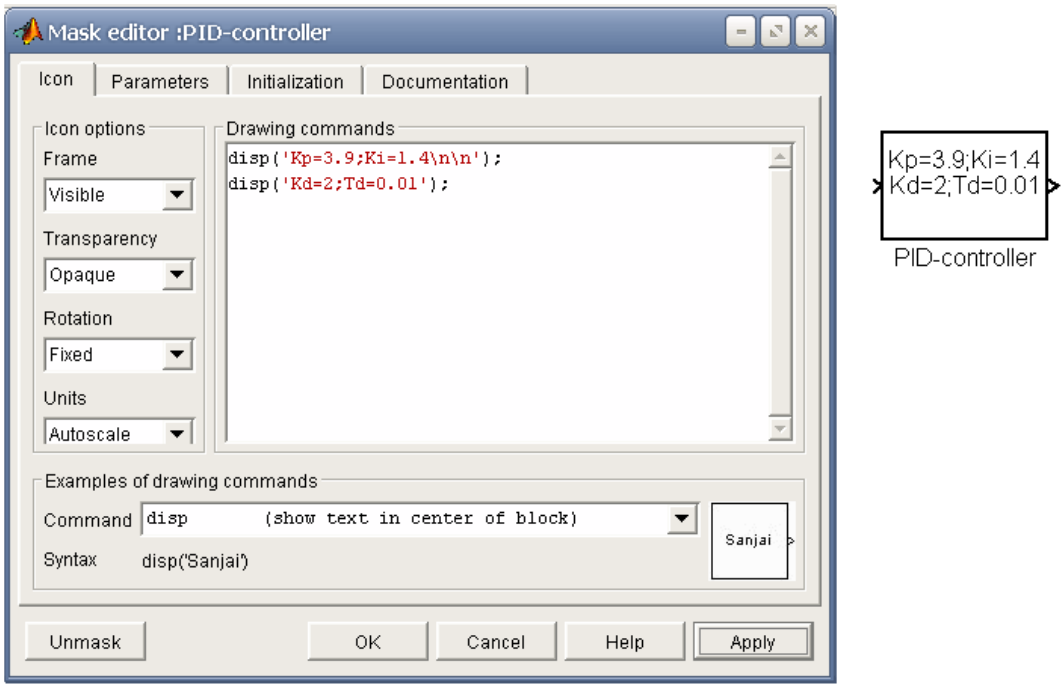


Рис. 19.15. Вивід тексту на блок підсистеми

У лівій частині вкладки **Icon** розташована група параметрів **Icon options** (Опції зображення). Ця група містить чотири списки, що розкриваються.

Frame – вибір способу відображення рамки блока: **Visible** – рамка видна; **Invisible** - рамка не видна.

Transparency – настроювання прозорості зображення блока: **Opaque** – зображення не прозоре; **Transparent** - зображення прозоре.

Rotation – настроювання можливості обертання рисунка при повороті блока: **Fixed** – при повороті блока орієнтація рисунка не змінюється; **Rotates** – рисунок обертається разом із блоком.

Units – завдання способу масштабування зображення: **Autoscale** – автоматичне масштабування, рисунок займає максимально можливу площу всередині блока; **Pixels** – розмір рисунка задається в пікселях і не змінюється при зміні масштабу блока; **Normalized** – постійний масштаб рисунка, при зміні розміру блока підсистеми пропорційно змінюється і масштаб рисунка.

19.2.3 Завдання параметрів підсистеми

Вкладка **Parameters** – дозволяє створювати власне діалогове вікно параметрів маскованої підсистеми і задавати змінні, які відповідають цим параметрам. Вкладка складається з наступних елементів (рис. 19.16): панелі **Dialog Parameters**, панелі **Option for selected parameter** і кнопок у лівій частині вкладки.

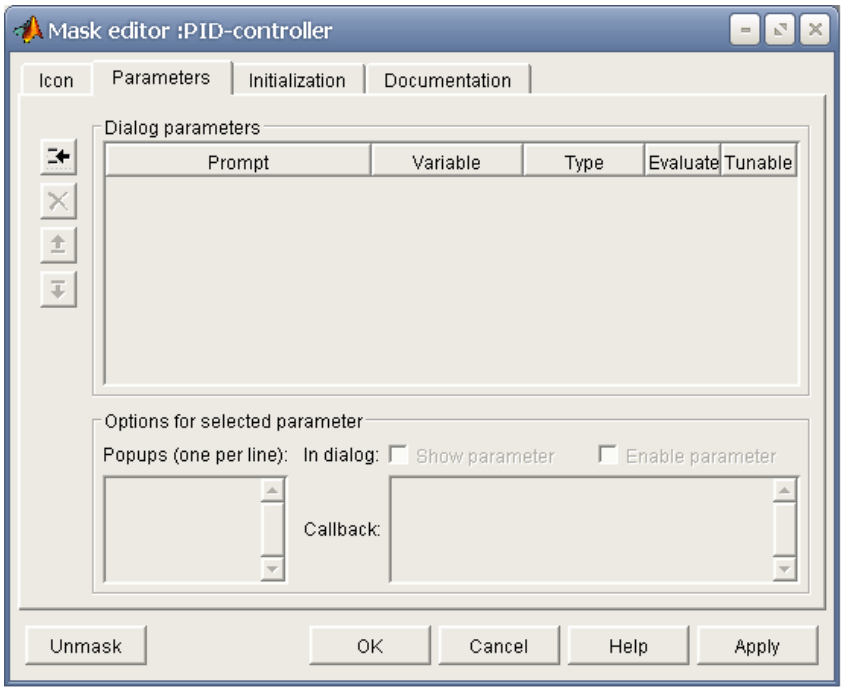


Рис. 19.16. Вкладка **Parameters** діалогового вікна **Mask Editor**

Панель **Dialog Parameters** дозволяє встановлювати і редагувати параметри маскованої підсистеми. Вона являє собою таблицю, кожен рядок якої відображає основні властивості одного з параметрів підсистеми. Таблиця має п'ять стовпців.

Promt – тут вводиться текст, що буде в діалоговому вікні описувати даний параметр.

Variable – у цьому стовпці задаються імена локальних змінних, значення яким будуть потім привласнені в діалоговому вікні маскованої підсистеми. Ім'я змінної повинне збігатися зі змінною, зазначеною в елементах підсистеми, якщо ні, то тоді ці змінні необхідно дорівняти друг до друга на вкладці **Initialization**.

Слід зазначити, що блоки у маскованій підсистемі не мають доступу до змінних, розміщених у робочій області MATLAB.

Маскована підсистема має свою власну область пам'яті, незалежну від робочої області MATLAB та інших маскованих підсистем у моделі. Це усуває можливість конфліктів імен змінних. Крім того, Simulink не робить розходжень між прописними й малими літерами в імені локальної змінної маскованої підсистеми, тому, наприклад Gain, GAIN і gain трактуються як те саме ім'я.

У стовпці **Type** у списку, що розкривається, вибирається спосіб завдання значення даного параметра. Список, що розкривається, містить три позиції**: edit, checkbox** і **popup**.

Вибор позиції **edit**, прийнятої за замовчуванням, дозволяє вводити значення параметра в спеціальному текстовому полі діалогового вікна параметрів підсистеми.

При виборі позиції **checkbox** у діалоговому вікні параметрів маскованої підсистеми буде сформований прапорець, що дозволяє вибрати значення змінної.

Позиція **popup** дозволяє користувачеві вибрати необхідне значення параметра зі списку, що розкривається. Сам список можливих значень задається у вікні **Popups** у поле **Option for selected parameter**.

Опція **Evaluate** визначає тип змінної даного параметра. У алежності від того, встановлений прапорець чи ні, значення змінної може приймати числове або символьне значення (табл. 19.1).

Таблиця 19.1

Значення локальної змінної при різних установках опцій Type і Evaluate

Type	**Evaluate**	
	☑	☐
edit	У полі вводиться числове значення або вираження	Змінній привласнюється символьне значення
checkbox	Прапорець встановлений - змінній привласнюється значення 1. Прапорець знятий - змінній привласнюється значення 0	Прапорець встановлений - змінній привласнюється значення 'on'. Прапорець знятий - змінній привласнюється значення 'off'
popup	Змінній, що пов’язана зі списком, привласнюється значення, яке дорівнює порядковому номеру пункту	Змінній привласнюється значення символьного рядка, що відповідає обраному пункту

Tunable – вибір цієї опції дозволяє користувачеві змінювати значення даного параметра безпосередньо протягом процесу моделювання.

В лівої частини вкладки **Parameters** вікна **Mask Editor** розташовані кнопки, що служать для керування списком параметрів підсистеми.

 (**Add**) – додає новий параметр у список параметрів маскованої підсистеми. При натисканні цієї кнопки на панелі **Dialog Parameters** стає активної новий рядок.

 (**Delete**) – ця кнопка дозволяє видалити поточний параметр.

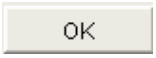
 (**Move up**) – переміщає виділений параметр на один рядок нагору.

 (**Move down**) – переміщає виділений параметр на один рядок униз.

Розглянемо для прикладу процес завдання параметрів для маскованої підсистеми `PID-controller` з рис. 19.9.

Для того щоб викликати редактор маски **Mask Editor** вже маскованої підсистеми, необхідно виділити підсистему і вибрати команду **Edit mask** з меню **Edit** у вікні моделі або з контекстного меню.

У вікні **Mask Editor,** що з'явиться, виберемо вкладку *Parameters*. За допомогою кнопки  створюємо локальні змінні (рис. 19.17), при цьому в поле **Prompt** указуємо опис змінних, а в поле **Variable** – їхні імена. Ці ж імена вкажемо в блоках, що входять до складу підсистеми (рис. 19.18). Інші параметри залишимо прийнятими за замовчуванням.

Створення локальних змінних завершене. Натискаємо кнопку  і у вікні моделі двічі клацаємо лівою кнопкою миші на маскованій підсистемі. Підсистема не відкривається, зате з'являється діалогове вікно (рис.19.19), у якому пропонується вказати значення локальних змінних маскованої підсистеми. Тепер з підсистемою можна працювати як зі звичайним блоком Simulink.

Відкрити масковану підсистему можна за допомогою команди **Look under mask** з меню **Edit** або з контекстного меню, а також натиснувши комбінацію клавіш <Ctrl + U>.

Для зміни параметрів маскованої підсистеми, необхідно виділити підсистему і вибрати команду **Edit mask** з меню **Edit** у вікні моделі або з контекстного меню.

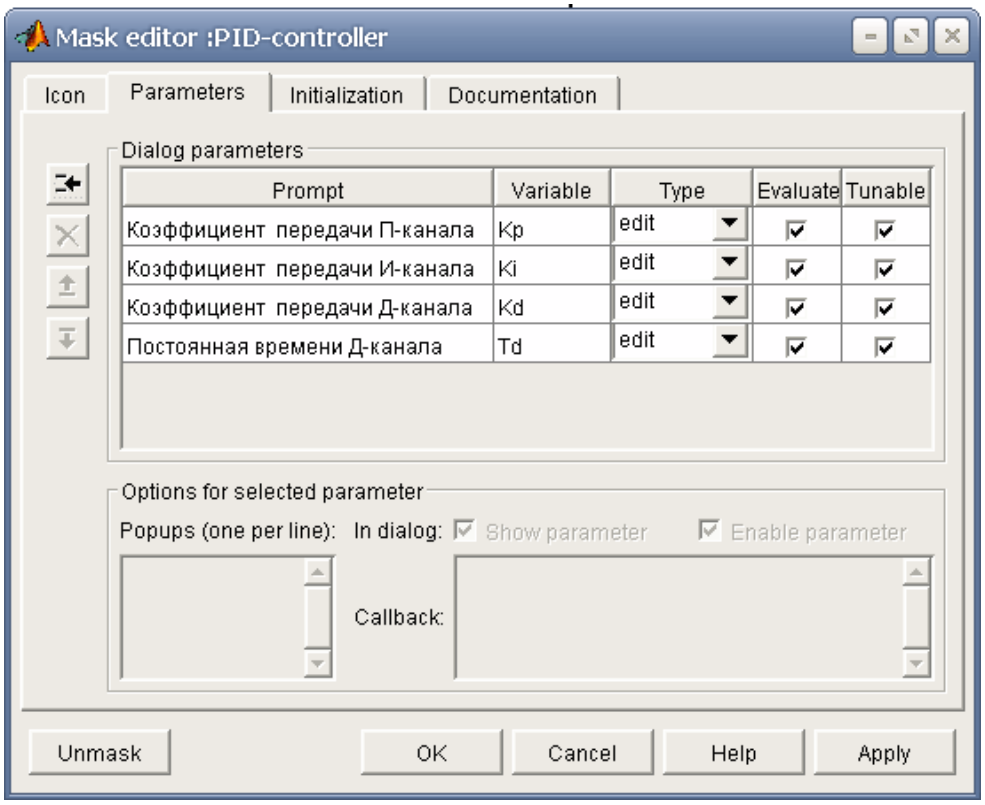


Рис. 19.17. Приклад завдання локальних змінних

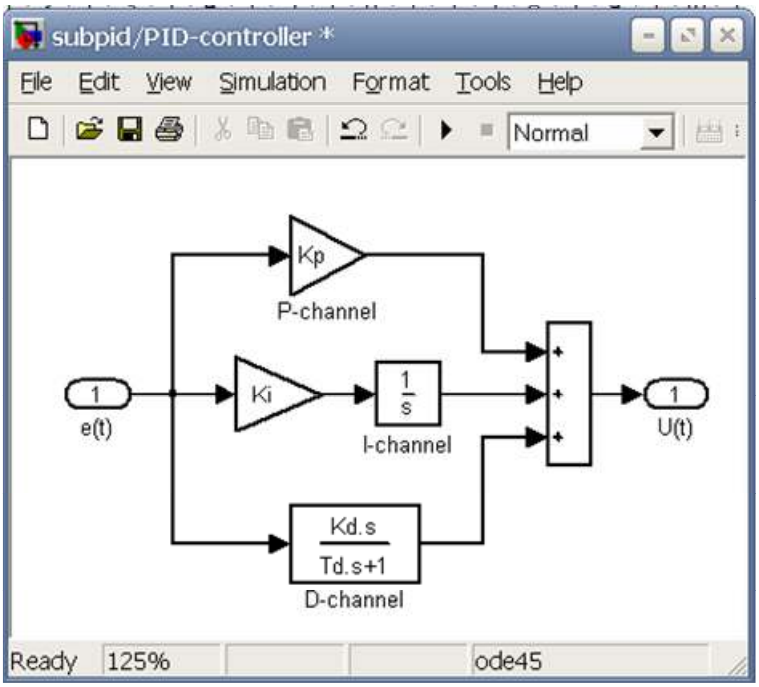


Рис. 19.18. Блоки підсистеми зі змінними параметрами

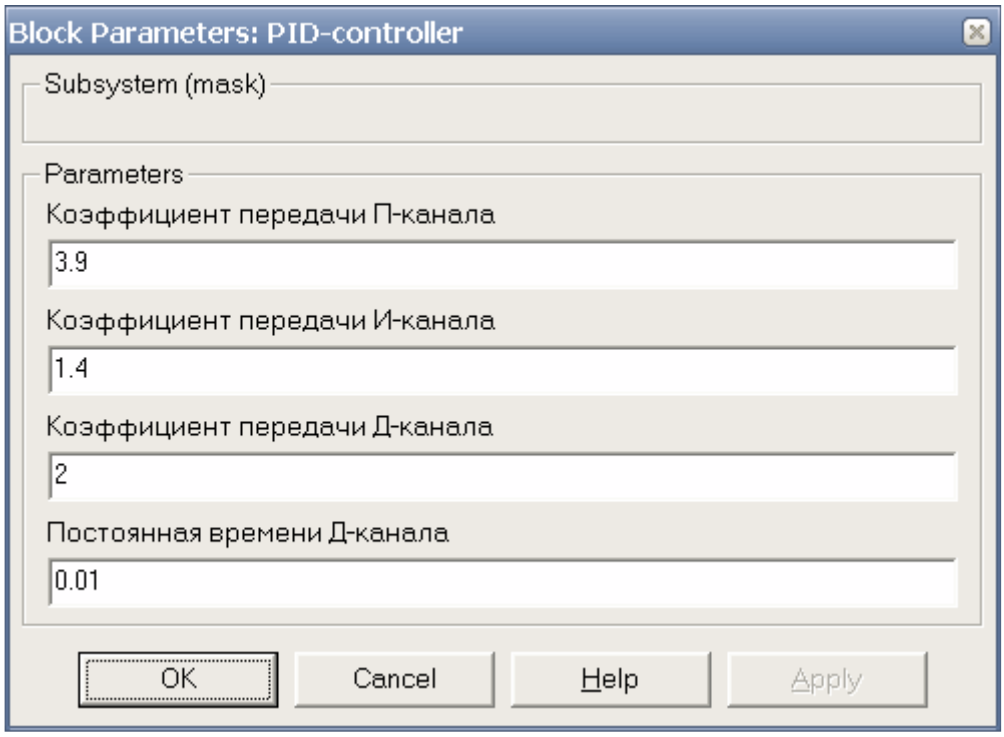


Рис. 19.19. Діалогове вікно параметрів підсистеми

Щоб скасувати маскування підсистеми, потрібно у вікні **Mask Editor** натиснути кнопку Unmask.

19.2.3 Вкладка Initialization

Вкладка **Initialization** (рис. 19.20) дозволяє користувачеві застосовувати команди MATLAB для додаткового настроювання маскованої підсистеми. Вкладка містить три поля.

Поле **Dialog variables** містить список змінних, які були задані на вкладці **Parameters**. При необхідності ці імена можна тут змінити, щоб не повертатися на попередню вкладку.

У полі **Initializations commands** вводяться команди для додаткового настроювання параметрів маскованої підсистеми. Тут, наприклад, можна задати початкові значення параметрів, встановити зв'язок між параметрами підсистеми, заданими на вкладці **Parameters** зі змінними, зазначеними в блоках підсистеми та ін. У полі **Initializations commands** можна використати будь-які команди та функції мовою MATLAB. Щоб результати виконання команд не відображалися в робочому вікні MATLAB, їх потрібно закривати крапкою з комою (`;`).

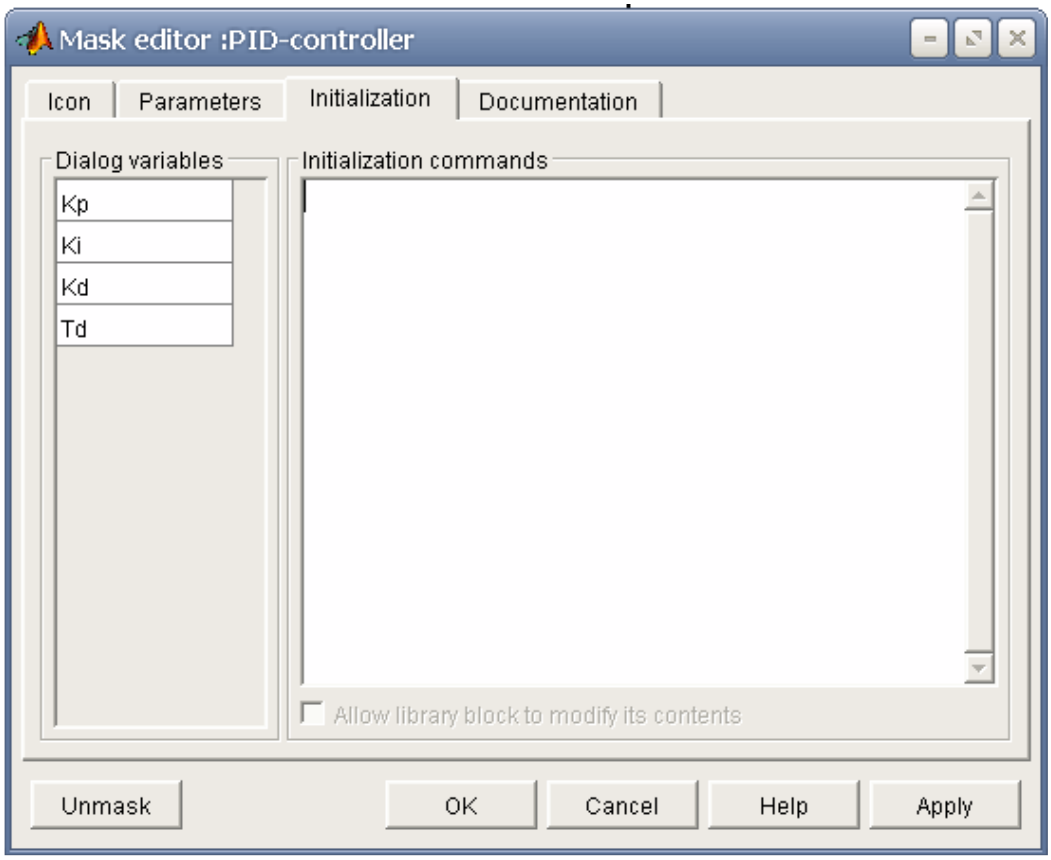


Рис. 19.20. Вкладка **Initialization** вікна **Mask Editor**

Прапорець `Allow library block to modify its components` стає доступним лише в тому випадку, якщо маскована підсистема поміщена в бібліотеку блоків. При встановленому прапорі дозволяється змінювати вміст маскованої підсистеми: додавати або видаляти блоки, змінювати їхні параметри.

19.2.4 Створення довідкової інформації

Вкладка **Documentation** (рис. 19.21) служить для створення довідкової інформації про масковану підсистему. Вкладка має три текстові поля.

У полі **Mask type** рекомендується вказати ім'я, що характеризує тип блока підсистеми, наприклад, «Регулятор». Це ім'я буде відображатися в заголовку діалогового вікна параметрів маскованої підсистеми (рис. 19.21). Щоб користувач міг відрізнити масковану підсистему від звичайного вбудованого блока, Simulink у вікні параметрів додає «`(mask)`».

Mask description - опис маски. У цьому полі вводиться коротка інформація про призначення підсистеми. Ця інформація також буде відображатися у вікні параметрів підсистеми.

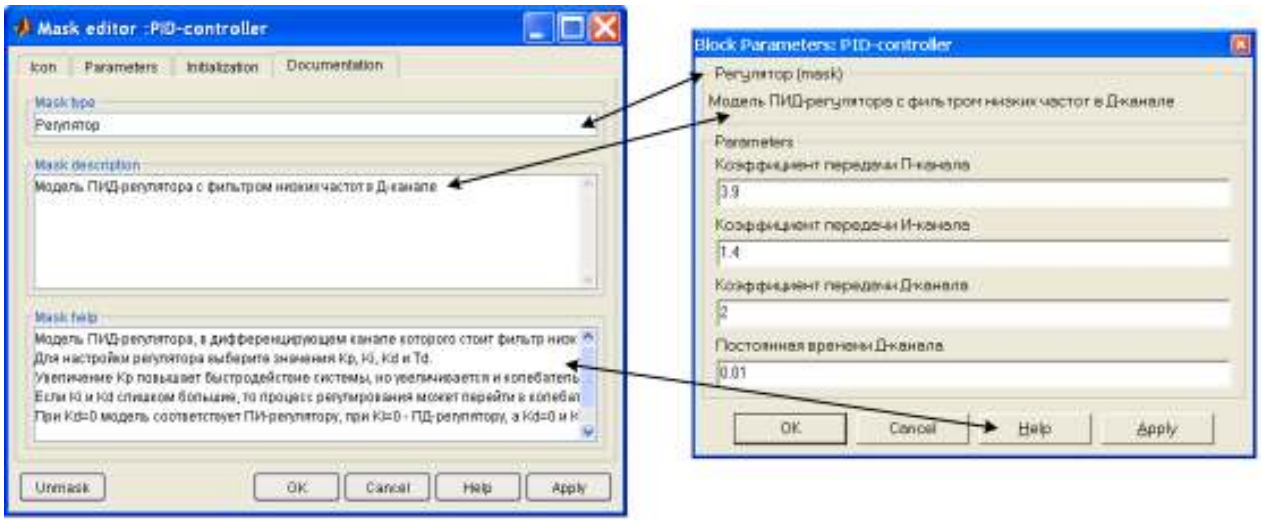


Рис. 19.21. Вкладка **Documentation** вікна **Mask Editor** й її зв'язок с вікном параметрів маскованої підсистеми

У полі **Mask help** рекомендується ввести докладну інформацію про призначення підсистеми і про те, як з нею працювати. Ця інформація буде розміщена в довідковій системі Simulink (рис. 19.22). Її можна буде переглянути, натиснувши на кнопку Help в діалоговому вікні параметрів маскованої підсистеми або вибравши команду **Help** з її контекстного меню.

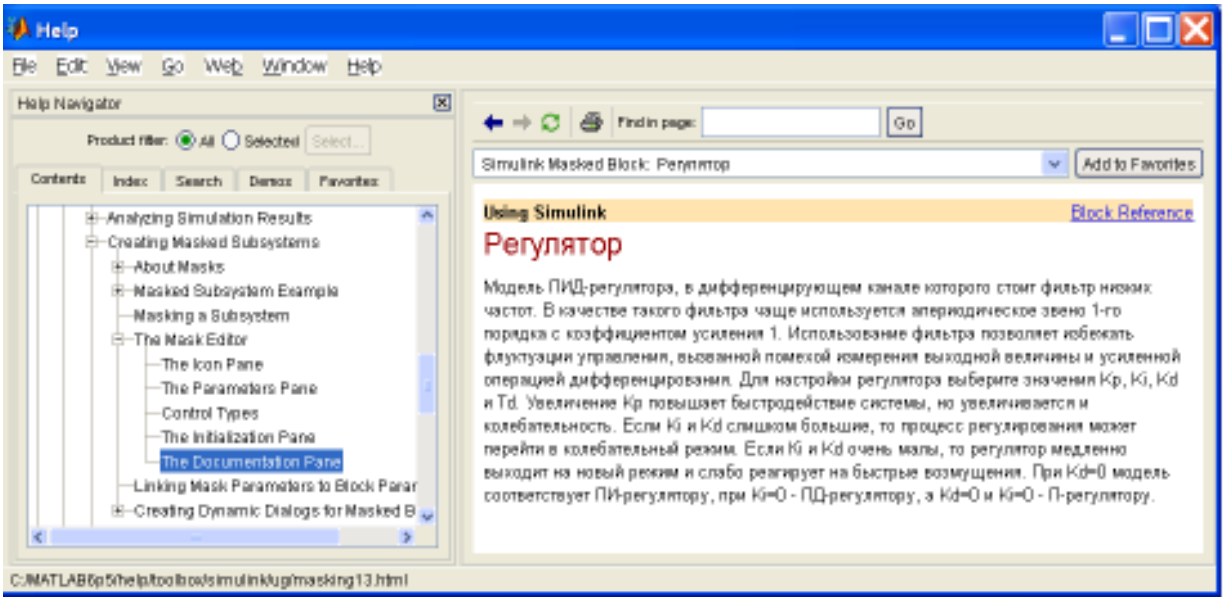


Рис. 19.22. Вивід довідки про масковану підсистему

Завдання для самостійної роботи

1. На рис. 9.7 зображена структурна схема системи керування літака з автопілотом.

а) Побудуйте Simulink-модель цієї системи без регулятора.

б) За допомогою команди **Create subsystem** з меню **Edit** вікна моделі об'єднайте блоки передаточних функцій приводу руля висоти та самого літака у підсистему.

в) Дайте створеній підсистемі ім'я „Aircraft".

2. Розгляньте систему керування з завдання 1.

а) Виконайте маскування створеної підсистеми.

б) Виведіть ім'я „Aircraft" маскованої підсистеми у центрі її піктограми.

в) Створіть довідкову інформацію про масковану підсистему з описом її складу та призначення.

3. Розгляньте систему керування з завдання 2.

а) Додайте у модель блок `Subsystem` з бібліотеки `Ports & Subsystems` та побудуйте в ньому ПІ-регулятор з передаточною функцією $W(s) = K_p + K_i/s$.

б) Виконайте маскування створеної підсистеми з ПІ-регулятором.

в) Створіть власне діалогове вікно параметрів маскованої підсистеми з ПІ-регулятором та задайте змінні K_p та K_i.

г) Встановіть K_p = 1,5 і K_i = 0,01 та побудуйте перехідну функцію системи. Порівняйте одержаний результат з рис. 9.8.

д) Змінюючи коефіцієнт підсилення K_i інтегруючого каналу ПІ-регулятора зробіть висновок щодо його впливу на якість системи.

е) Створіть довідкову інформацію про підсистему з ПІ-регулятором.

ж) Збережіть файл з моделлю у папці work.

20. СТВОРЕННЯ ЗВІТУ ПРО МОДЕЛЮВАННЯ

Як правило, після проведення моделювання його результати необхідно оформити у вигляді звіту. Звіт являє собою текстовий документ і може бути частиною курсового або дипломного проекту, звітом про наукову працю та ін. У більшості випадків при створенні текстових документів використовується редактор Microsoft Word, тому в даній главі буде докладно розглянуто, яким чином можна перенести в Microsoft Word створені в Simulink моделі та результати моделювання - графіки.

20.1 Перенос Simulink-моделей в Microsoft Word

Перенести побудовану в Simulink модель у документ Word можна за допомогою буфера обміну. Для цього у вікні моделі Simuink у меню **Edit** потрібно вибрати команду **Copy model to clipboard** (рис. 20.1), при цьому модель копіюється в буфер обміну (clipboard) Windows.

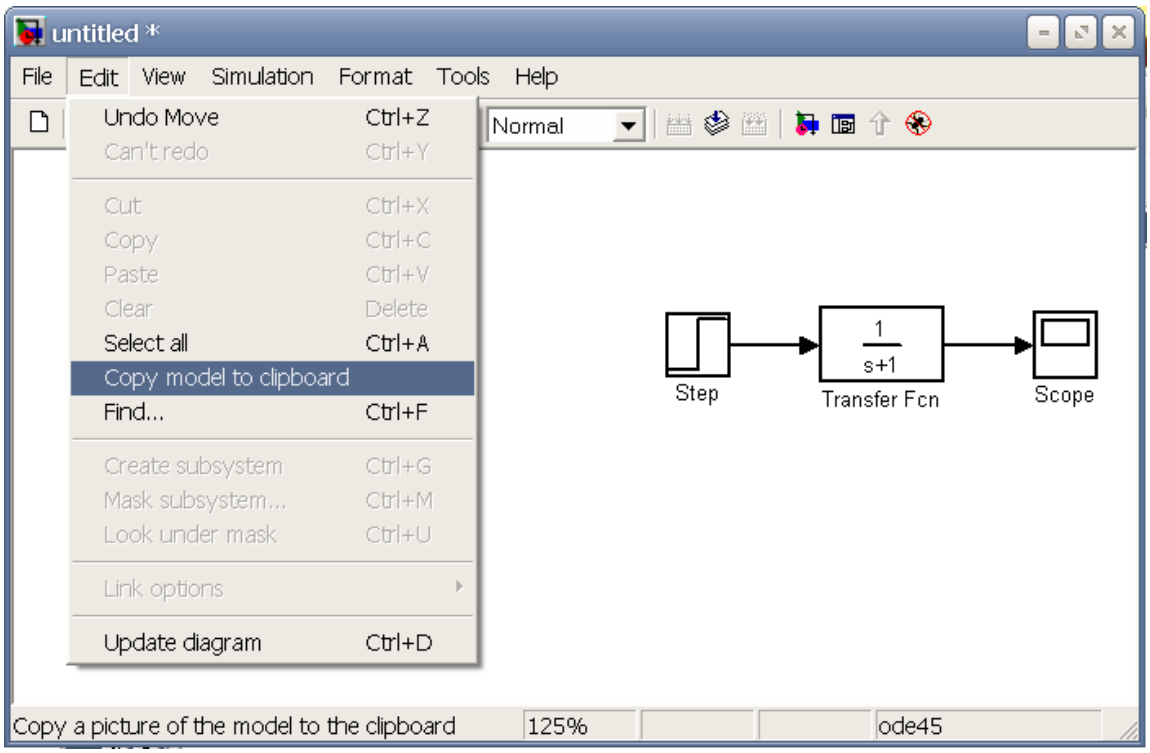


Рис. 20.1. Копіювання структури моделі з Simulink

Звичайно, можна скопіювати в буфер все вікно з моделлю одночасним натисканням клавіш <Alt+PrtScr>. Однак при цьому треба простежити, щоб вікно з необхідною моделлю було активним, і в ньому містилася вся модель (або та її частина, яку необхідно скопіювати). Тепер модель можна вставити в документ Word будь-яким зручним користувачеві способом. Наприклад, вставити модель із буфера обміну в текст документа, починаючи з позиції курсору, можна за допомогою команди **Вставить** з меню **Правка** (рис. 20.2) або з контекстного меню (рис. 20.3). Нагадаємо, що контекстне меню в Windows-додатках викликається щигликом правої кнопки миші.

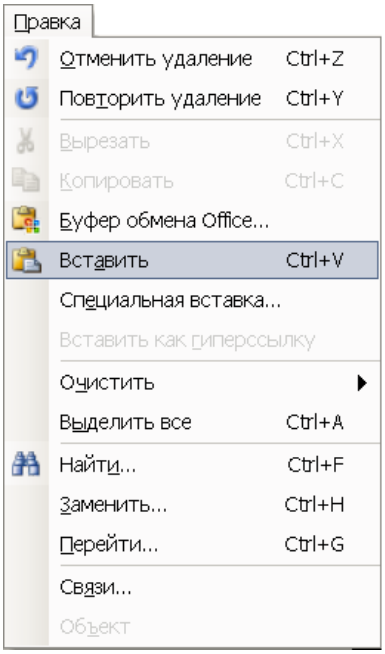


Рис. 20.2. Меню **Правка** Microsoft Word

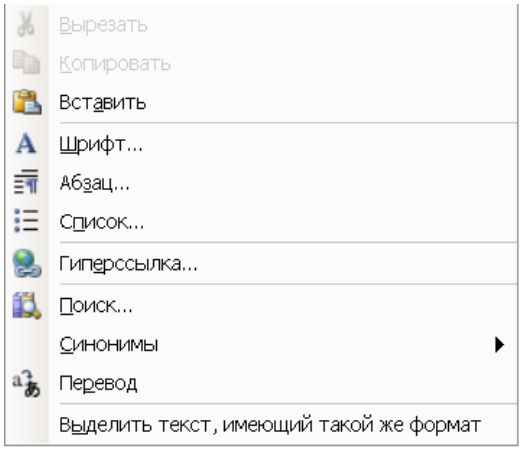


Рис. 20.3. Контекстне меню роботи з текстом

Крім того, для вставки вмісту буфера обміну можна натиснути кнопку  на панелі інструментів Word або комбінацію клавіш <Ctrl+V>.

Після вставки рисунка моделі часто необхідно зробити його обрізку, щоб видалити порожній простір. Це можна зробити за допомогою інструмента «Обрезка», що активізується натисканням кнопки  на панелі інструментів **Настройка изображения**. У випадку відсутності панелі її можна викликати на екран одним з наступних способів: вибрати **Вид → Панели инструментов → Настройка изображения**; **Сервис → Настройка → Панели инструментов** і поставити прапорець напроти відповідної позиції або вибрати команду **Отобразить панель настройки изображения** в контекстному меню зображення моделі (рис. 20.4).

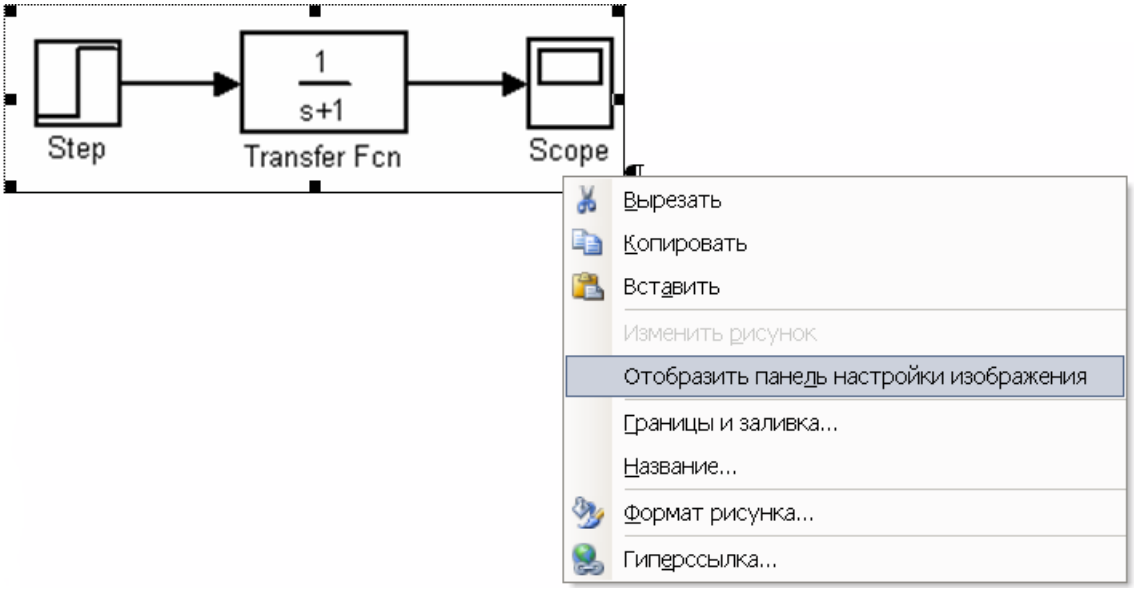


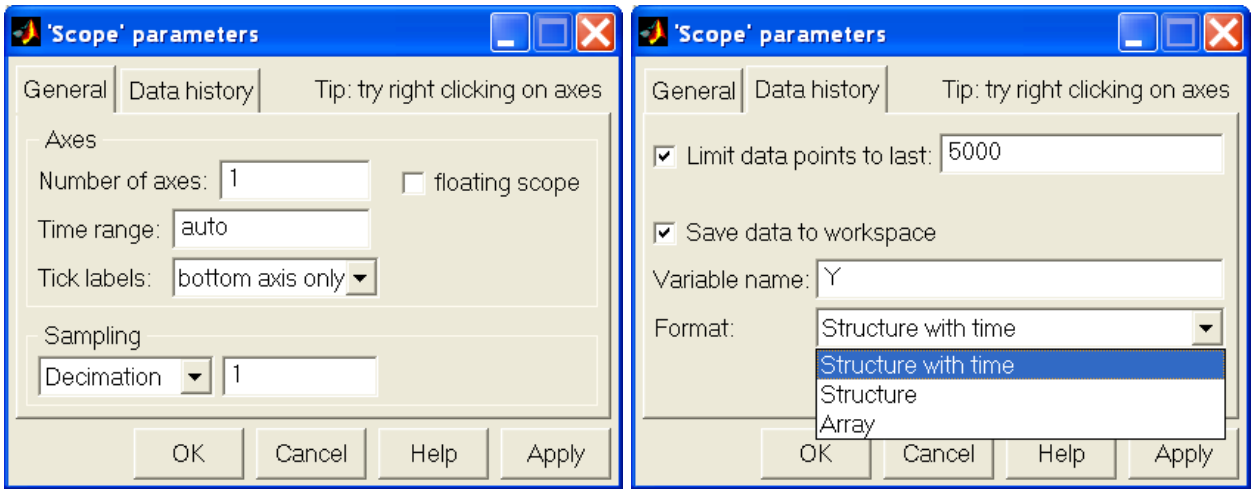
Рис. 20.4. Контекстне меню роботи із зображенням

20.2 Передача графічної інформації в Microsoft Word

На жаль, безпосередньо зберегти графік, побудований в Simulink, як графічний файл, не можна. Немає в Simulink і спеціальної команди, що передає вміст графічного вікна в буфер обміну. Тут на допомогу може прийти описаний вище спосіб копіювання активного вікна за допомогою комбінації клавіш <Alt+PrtScr>. Однак є ще один спосіб переносу вмісту графічного

вікна Simulink в Word. Цей спосіб хоч і більш трудомісткий, але дає користувачеві більше можливостей по редагуванню графіка. Для цього необхідно спочатку передати масив даних, отриманих у результаті моделювання, у робочу область MATLAB (**Workspace**). Зробити це можно таким чином.

Викличемо вікно **Параметри** осцилографа **'Scope' parameters** натисканням кнопки  **(Parameters)** на панелі інструментів діалогового вікна блока Scope (див. рис. 13.7). Вікно, що з'явиться (рис. 20.5), має дві вкладки: **General** (Загальні параметри) і **Data History** (Історія даних) і дозволяє задавати параметри форматування графіків, встановлювати режим плаваючого осцилографа і розміщати масиви даних у робочій області.



а *б*

Рис. 20.5. Діалогове вікно параметрів блока `Scope`:
а – вкладка **General**; *б* - вкладка **Data History**

Для того щоб сформувати масив даних з Simulink у робочій області MATLAB, необхідно зробити активною вкладку **Data History** і поставити прапорець напроти опції **Save data to workspace**. При цьому стануть активними поля **Variable name** і **Format** (рис. 20.5, *б*).

У текстовому полі **Variable name** потрібно вказати ім'я змінної, котра створюється в робочій області (за замовчуванням ScopeData). Цій змінній буде привласнений масив даних, сформований у результаті моделювання. Задамо, для прикладу, у поле **Variable name** ім'я змінної `Y`.

У списку **Format**, що розкривається, представлений набір форматів передачі даних у робочу область (рис. 20.5, б*)*:

- **Structure with time** (Структура з часом);
- **Structure** (Структура);
- **Array** (Масив).

При виборі позиції **Array** в MATLAB формується числовий масив даних, у першому стовпці якого розміщуються значення моментів часу моделювання, а в наступних стовпцях записуються відповідні значення вхідних сигналів блока `Scope`.

У випадку вибору позиції **Structure** у робочій області створюється масив даних, вміст поля `time` якого є порожнім масивом.

Якщо вибрати позицію **Structure with time**, то в робочій області MATLAB формується масив даних, що включає вектор відповідних їм значень моментів часу.

Для передачі даних у робочу область MATLAB можна також використати блок `To Workspace` (рис. 20.6), що також міститься в бібліотеці блоків `Sinks`. Цей блок зручно використовувати в тому випадку, якщо будувати графіки даної функції в Simulunk немає потреби.

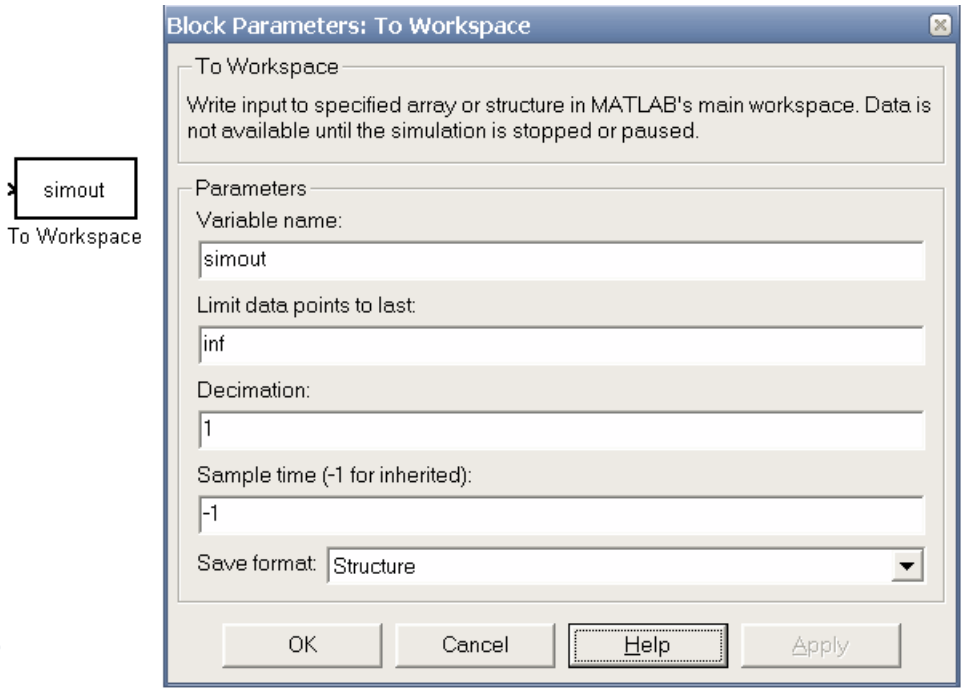


Рис. 20.6. Зображення і вікно параметрів блока `To Workspace`

Блок має наступні параметри (рис. 20.6):

Variable name – ім'я змінної, якій привласнюються дані (за замовчуванням `simout`);

Limit data points to last – максимальна кількість розрахункових точок за часом, що зберігаються, (відлік ведеться від моменту завершення моделювання). У тому випадку, якщо значення цього параметра задано як `inf`, то в робочій області будуть збережені всі дані;

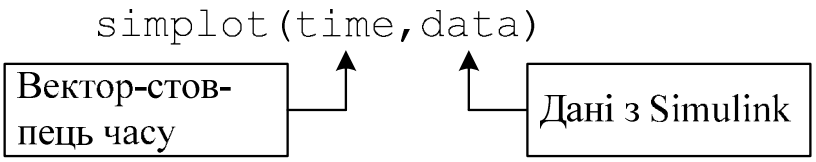
Decimation – кратність запису даних у робочу область.

Sample time – період квантування;

Save format – формат збереження даних. Цей параметр аналогічний параметру **Format** блока `Scope` і також містить три розглянуті вище позиції: **Structure with time**, **Structure** і **Array**.

Зрозуміло, що у робочу область дані можуть надійти лише після того, як ці дані Simulink *прорахував*, тобто якщо користувач оголосив нову змінну або вніс якісь зміни у формат виводу даних, необхідно дані перерахувати, тобто ще раз повторити моделювання.

Дані про результати моделювання в Simulink, що зберігаються в робочій області MATLAB, можна використати в подальшій роботі, у тому числі й будувати графіки. Для цього використовується команда `simplot`, що у загальному випадку має наступний формат:



Аргумент `time` необхідно вказувати в тому випадку, якщо дані в робочій області мають формат **Array** або **Structure**. Якщо ж дані з Simulink мають формат **Structure with time**, аргумент `time` у команді `simplot` вказувати не треба:

```
simplot(data)
```

Команда `simplot` дозволяє побудувати графік за даними, отриманими з Simulink у звичайному графічному вікні MATLAB, тому до нього можна застосовувати всі способи редагування, розглянуті в п. 4.2: змінювати тип, кольори та товщину ліній,

кольори фону, масштаб, накладати і знмати сітку, додавати написи, а також копіювати графік у буфер обміну і зберігати його в окремий файл.

На рис. 20.7 показана побудована за допомогою команди `simplot` перехідна функція системи з ПІД-регулятором (рис. 15.29). За вихідну змінну в полі **Variable name** параметрів блока `Scope` зазначена змінна `Y` (рис. 20.5, *б)*. Відредагуємо цей графік і передамо його в документ Word.

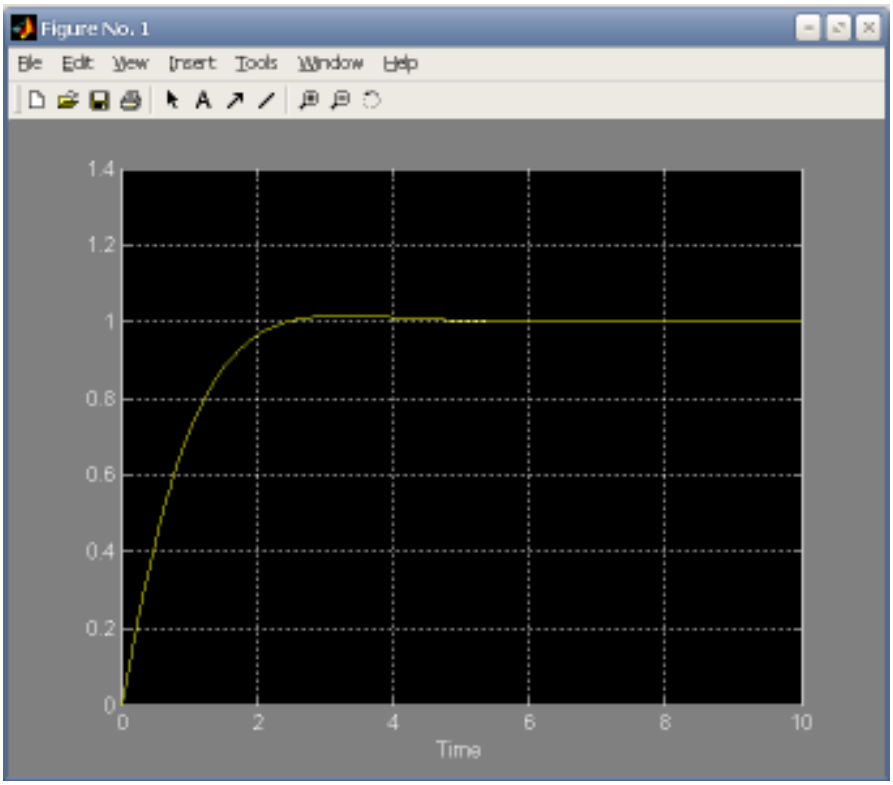


Рис. 20.7. Перехідна функція системи з ПІД-регулятором, побудована командою `simplot`

Одне з головних завдань - позбутися від сірих кольорів графічного вікна і чорного фону самого графіка. Крім того, бажано поміняти колір лінії графіка, тому що прийнятий за замовчуванням жовтий колір погано виден.

Забрати темні фонові кольори простіше всього в таким чином. На панелі інструментів графічного вікна виберемо команду **File → Preferences**. У вікні настроювань, що з'являється, виберемо **Figure Copy Template → Copy Options** (рис. 4.8) і у групі параметрів **Figure background color** (Кольори фону фігури) виберемо позицію

Force white background. У цьому випадку при копіюванні графіка в буфер обміну або при збереженні його як рисунка, фон графічного вікна має білий колір.

Кольори фону, а також кольори і тип лінії графіка можна поміняти, вибравши в меню графічного вікна команду **Edit → Figure Properties…**, **Edit→AxesProperties…** або двічі клацнувши у відповідному місці інструментом **Edit Plot** (кнопка на панелі інструментів). При цьому з'явиться вікно **Property Editor**. Об'єкт для редагування можна вибрати в списку, що розкривається, **Edit Properties for***: figure: 1* для зміни загальних параметрів графічного вікна (рис. 4.9); *axes:* для зміни властивостей вісей (рис. 4.10); *line: `Run #1 – data1`* - для зміни властивостей лінії графіка.

В списку, що розкривається, **Color** на вкладці **Style** вікна **Property Editor - Figure** можна вибрати кольори графічного вікна (рис. 20.8), найкраще білий. Правда, при цьому треба зробити більш темними вісі графіка на вкладках **X** і **Y** вікна **Property Editor – Axes**.

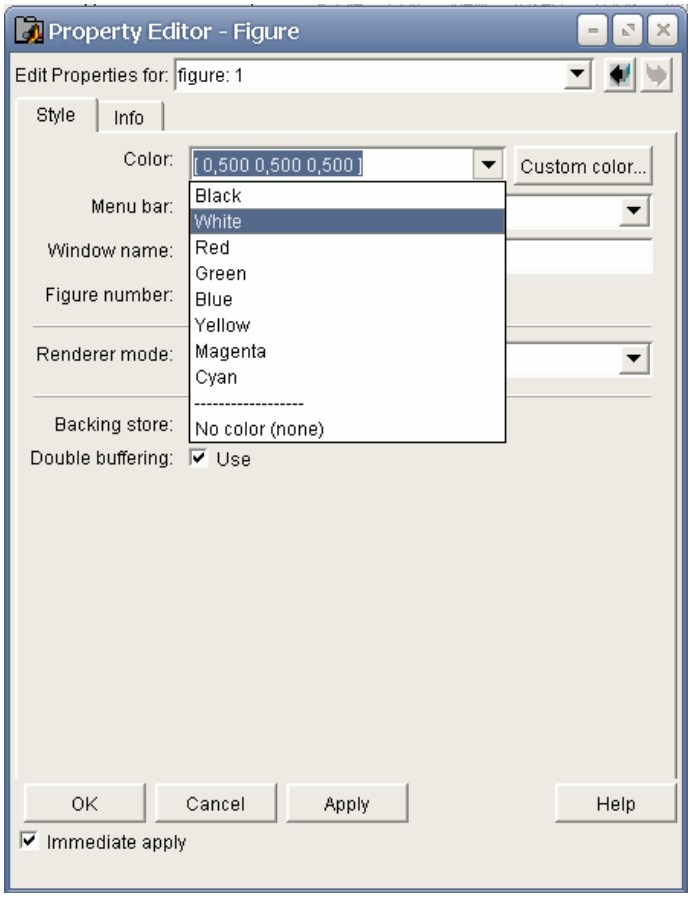


Рис. 20.8. Зміна кольорів графічного вікна

У полі **Background** на вкладці **Style** вікна **Property Editor – Axes** вибирається колір фону графіка (рис. 20.9).

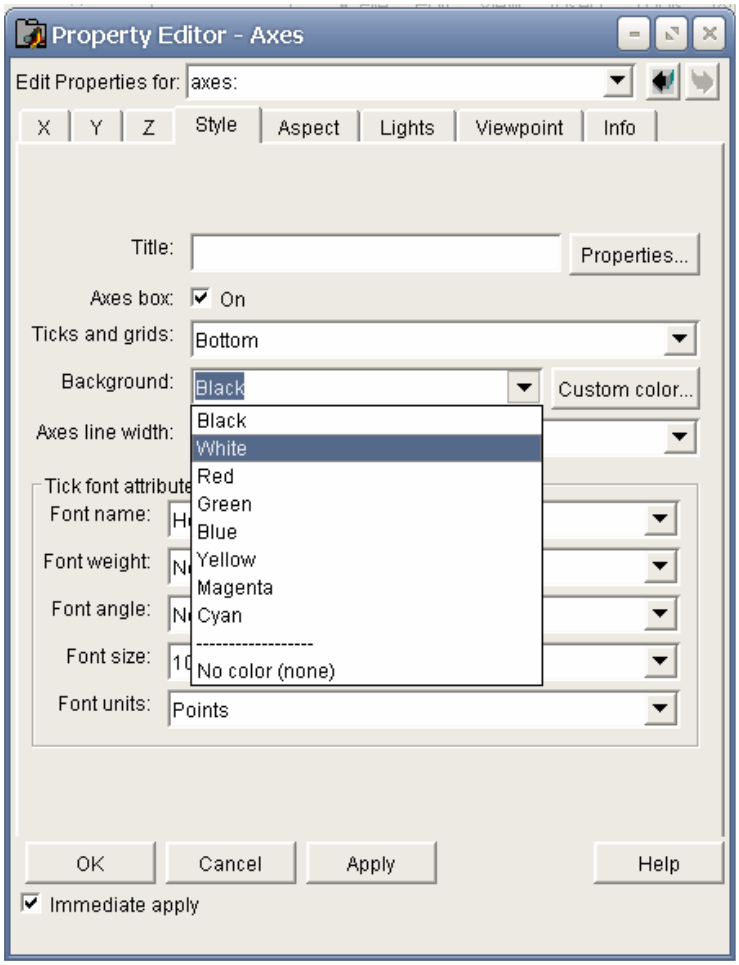


Рис. 20.9. Зміна фону графіка

Нам залишилося змінити колір лінії графіка. Для цього в діалоговому вікні **Property Editor** у списку **Edit Properties for:** вибираємо позицію *line: `Run #1 – data1`*. У вікні, що з'являється, **Property Editor – Line** на вкладці **Style** (рис. 20.10) можна змінювати стиль лінії (**Line style:**), її товщину (**Line width:**), колір (**Line color:**) та ін.

Тепер перенесемо графік в Microsoft Word. Його можна передати в буфер обміну (**Edit → Copy Figure**) і вставити в документ Word будь-яким зручним користувачеві способом. Крім того, вікно із графіком можна зберегти як файл рисунка. Це особливо зручно, якщо графік може знадобитися при роботі з іншими документами або додатками.

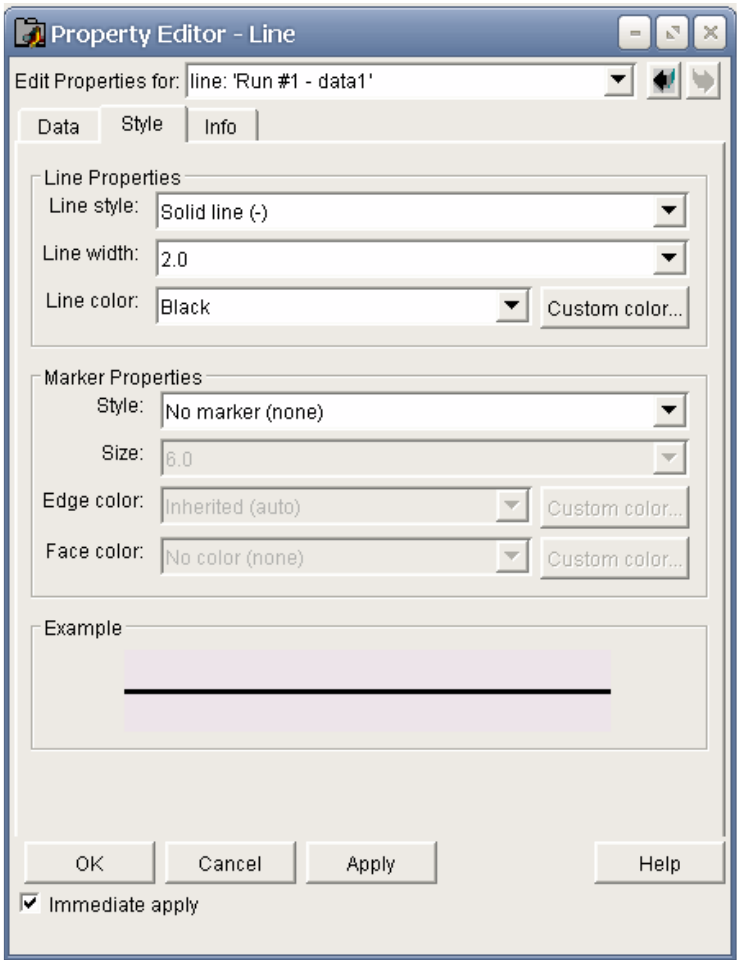


Рис. 20.10. Зміна властивостей лінії графіка

Альтернативном способом зміни властивостей графіків у Simulink є наступний. Якщо у командному рядку MATLAB ввести команди

```
set(0,'ShowHiddenHandles','On')
set(gcf,'menubar','figure')
```

то у вікні активного (виділеного) блоку `Scope` з'явиться додаткове меню, яке є аналогічним меню графічного вікна MATLAB (рис.20.11). Тепер можна описаним вище способом змінювати зовнішній вид графічного вікна, вісей та самих графіків у відповідності зі своїми вимогами.

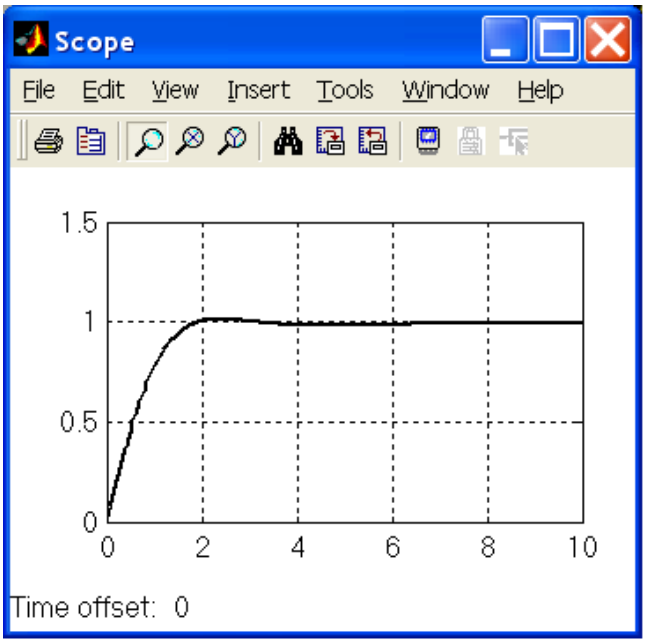


Рис. 20.11. Вікно блоку `Scope` із додатковим меню

Щоб зберегти графік як файл рисунка потрібно в меню **File** вибрати команду **Export…** і в діалоговому вікні, що відкрилося, вказати каталог, у який Ви хочете зберегти файл, ім'я файлу і його тип. Збережений рисунок вставляється в Microsoft Word вибором команди **Вставка→Рисунок→Из файла…** (рис. 20.12).

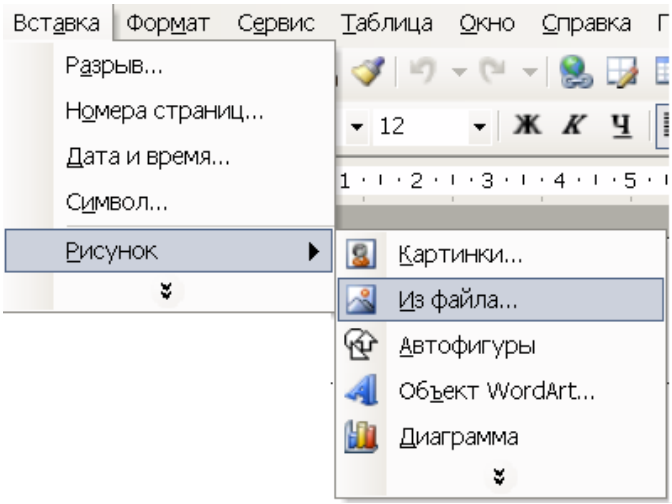


Рис. 20.12. Вставка рисунка в Word з файлу

На рис. 20.13 показаний графік перехідної функції, побудований в MATLAB за допомогою команди `simplot`,

відредагований за допомогою редактора графічного вікна і вставлений у документ Word через буфер обміну.

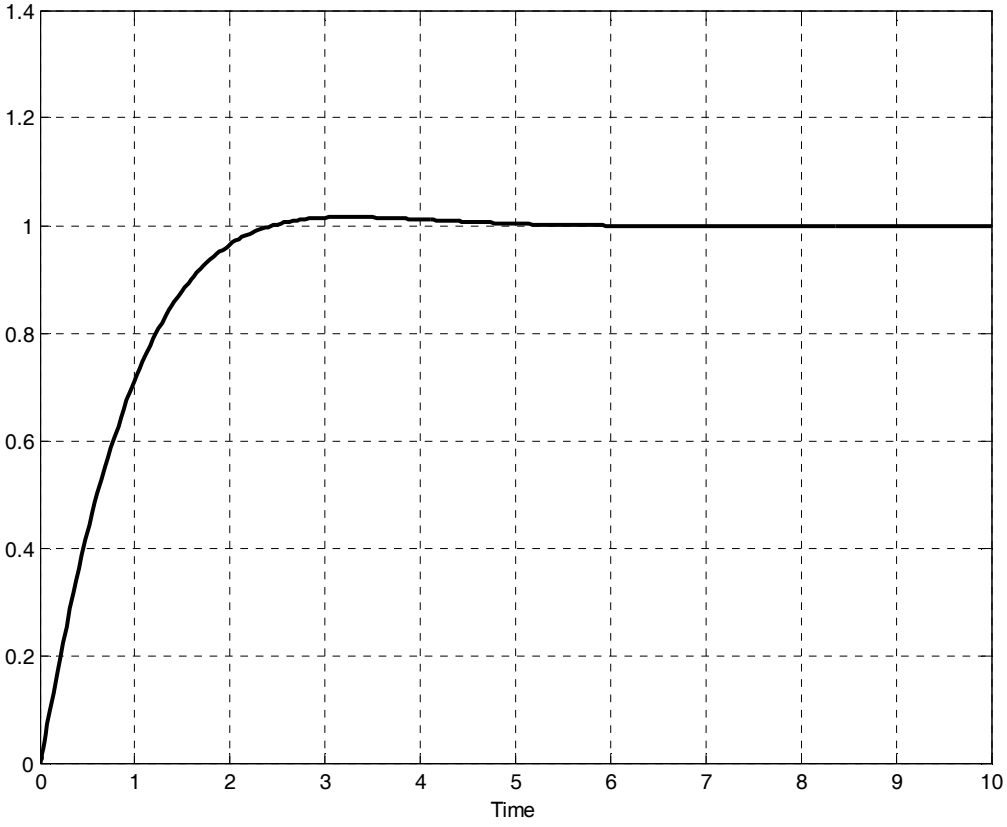


Рис. 20.13. Графік, побудований командою `simplot` і вставлений в Word через буфер обміну

Завдання для самостійної роботи

1. Зробіть у текстовому редакторі Microsoft Word звіт про створення та дослідження системи керування літака з автопілотом, що розглядалася у завданнях до попередньої глави.

а) Скопіюйте модель з ПІ-регулятором у буфер обміну за допомогою команди **Copy model to clipboard** і вставте її в Word.

б) Обріжте вставлений рисунок з моделлю за допомогою інструмента «Обрезка».

в) За допомогою комбінації клавіш <Alt+PrtScr> скопіюйте і вставте в Word довідкову інформацію про модель та її підсистеми.

2. За допомогою команди `simplot` побудуйте графік перехідної функції системи при K_p = 1,5 і K_i = 0,01 та встановіть наступні параметри отриманої фігури

а) фон вікна графіка - білий;

б) колір лінії графіка, його осей та усіх підписів - чорний;

в) вставте відредаговане зображення перехідної функції системи у звіт.

3. Побудуйте реакцію системи на лінійно наростаючий вхідний сигнал.

а) За допомогою блоку `To Workspace` передайте дані про зміну вихідного сигналу у робочу область MATLAB.

б) Повторіть дії пунктів а-в з попереднього завдання.

в) Що Ви можете сказати про порядок астатизму даної системи?

Алфавітний перелік команд MATLAB

ЛІТЕРАТУРА

1. Современные системы управления/Р. Дорф, Р. Бишоп. Пер. с англ. Б.И. Копылова. - М.: Лаборатория Базовых Знаний, 2002. – 832 с.
2. Системы управления с обратной связью/Ч. Филлипс, Р. Харбор. Пер. с англ. Б.И. Копылова. - М.: Лаборатория Базовых Знаний, 2001. – 768 с.
3. Методы классической и современной теории автоматического управления: Учебник в 5-и тт.; 2-е изд., перераб. и доп. Т1: Математические модели, динамические характеристики и анализ систем автоматического управления / под ред. К.А. Пупкова и Н.Д. Егупова. - М.: Издательство МГТУ им. Н.Э. Баумана, 2004. 656 с.
4. Методы классической и современной теории автоматического управления: Учебник в 5-и тт.; 2-е изд., перераб. и доп. Т3: Синтез регуляторов систем автоматического управления / под ред. К.А. Пупкова и Н.Д. Егупова. - М.: Издательство МГТУ им. Н.Э. Баумана, 2004. 616 с.
5. Проектирование систем управления/ Г.К. Гудвин, С.Ф. Гребе, М.Э. Сальгадо. – М.: БИНОМ. Лаборатория базовых знаний, 2004. – 911 с.
6. Теория автоматического регулирования/А.С. Востриков, Г.А. Французова. – М.: Высш. шк., 2004. – 365 с.
7. MATLAB6/6.1/6.5+Simulink 4/5. Основы применения. Полное руководство пользователя/ Дьяконов В.П. М.: СОЛОН-Пресс. – 2002. – 768 с.
8. Медведев В.С., Потемкин В.Г. Control System Toolbox. MATLAB 5 для студентов/Под общ. ред. к.т.н. В.Г. Потемкина. – М.: ДИАЛОГ-МИФИ, 1999. – 287 с.
9. Никульчев Е.В. Практикум по теории управления в среде MATLAB: Учебное пособие. – М.: МГАПИ, 2002. – 88 с.
10. Черных И.В. SIMULINK: среда создания инженерных приложений/Под общ. ред. к.т.н. В.Г. Потемкина. – М.: ДИАЛОГ-МИФИ, 2003. – 496 с.
11. Дэбни Дж. Simulink 4. Секреты мастерства. – М.: Лаборатория знаний. – 2003. - 403 с.

12. Бесекерский В.А., Попов Е.П. Теория систем автоматического управления / В. А. Бесекерский, Е. П. Попов. - Изд. 4-е, перераб. и доп. — СПб.: Профессия, 2003. - 752 с.

13. Теория автоматического управления: Нелинейные системы управления при случайных воздействиях/ Под ред. А.В. Нетушила. Учебник. - М.: Высшая школа, 1983.

14. Поляков К.Ю. Основы теории цифровых систем управления: учеб. пособие; СПбГМТУ. – СПб.: 2006. 161 с.

15. Половко А. М., Бутусов П. Н. MATLAB для студента. - СПб.: БХВ-Петербург, 2005. —320 с.

16. Лазарев Ю. Ф. Начала программирования в среде MatLAB: Учебное пособие. - К.: НТУУ "КПИ", 2003. - 424 с.

17. Simulink Documentation [Електронний ресурс]. - Режим доступу: http://www.mathworks.com/ access/ helpdesk/ help/ toolbox/ simulink.

18. Matlab Documentation.[Електронний ресурс]. - Режим доступу: - http://www.mathworks.com/ access/ helpdesk/help/toolbox/control.

19. Информатика: Базовый курс/С.В. Симонович и др.– СПб.: Питер, 2002. - 640 с.

20. Образовательный математический сайт "Exponenta.ru". [Електронний ресурс]. — Режим доступу: www.exponenta.ru.

21. Getting Started with MATLAB. Version 6.5. The MathWorks, Inc., 2002.