МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ АВТОМОБІЛЬНО-ДОРОЖНІЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робот з дисципліни
«Теорія автоматичного керування»

Харків 2016

МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ АВТОМОБІЛЬНО-ДОРОЖНІЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ

з дисципліни
«Теорія автоматичного керування»
для студентів спеціальностей
6.050202 та 6.050702

Затверджено методичною
радою університету,
протокол № _ від __. __.2016 р.

Харків ХНАДУ 2016

Укладачі: Гурко О.Г.
Кононихін О.С.
Кафедра автоматизації та комп’ютерно-інтегрованих технологій

ВСТУП

Проектування сучасних систем керування неможливе без використання математичного моделювання за допомогою спеціальних пакетів прикладних програм, серед яких одним з найвідоміших і пристосованих для інженерних потреб є середовище MATLAB. Для вирішення задач керування і імітаційного моделювання до складу MATLAB входять декілька пакетів розширення (Toolboxes), зокрема Control Systems Toolbox та Simulink.

Control Systems Toolbox працює з безперервними і дискретними моделями систем у вигляді передаточних функцій, а також з моделями в просторі станів і дозволяє вирішувати класичні задачі: розрахунок динамічних та частотних характеристик; проектування регуляторів частотними і кореневими методами, підтримує системи з запізнюванням тощо.

Могутнім засобом дослідження динамічних систем є середовище візуального моделювання Simulink.

Для найефективнішого використовування MATLAB і, для аналізу та синтезу систем автоматичного керування необхідно уміти працювати зі всіма спеціалізованими пакетами розширення, які вигідно доповнюють один одного. Зрозуміло, використання цих пакетів неможливе без володіння навиками роботи в MATLAB.

Дані методичні вказівки покликані допомогти набути навики роботи в MATLAB і використовувати Control Systems Toolbox для аналізу та синтезу систем автоматичного керування.

Методичні вказівки включають цикл з 6 лабораторних робіт, кожна з яких розрахована на 6 годин: 4 години аудиторних занять і 2 години самостійної підготовки. Хоча кожна з лабораторних робіт вміщує теоретичну частину, в якій висловлюються основні відомості, необхідні для її виконання, автори наполегливо рекомендують перед аудиторними заняттями повторювати лекційний матеріал, присвячений відповідній темі.

Для контролю надбаних знань і умінь, у кінці кожної з лабораторних робіт приводяться вправи, призначені для самостійного виконання. Крім того, викладач на свій розсуд може видати індивідуальні завдання.

ЛАБОРАТОРНА РОБОТА №1

ОСНОВИ MATLAB

Мета роботи: придбання практичних навичок роботи з пакетом MATLAB: вводу та виводу змінних, виконання простих математичних операцій, налаштування робочого вікна MATLAB.

1.1 Загальні положення

MATLAB (скорочено від MATRIX LABORATORY) - це інтерактивне середовище для наукових та інженерних обчислень. Система MATLAB з'явилася в 1984 р. і швидко завоювала популярність у фахівців у різних галузях науки й техніки, оскільки вона дозволяє застосовувати до будь-яких даних, що задані у вигляді векторів і матриць, різноманітні чисельні алгоритми. Зручна мова програмування, у якій завдяки матричній орієнтації системи значно зменшилася необхідність у циклах, ще більше розширила сферу застосування MATLAB.

До складу MATLAB входить основна програма (ядро) і спеціалізовані **пакети прикладних програм** (too1boxes), що складаються з так званих *m-файлів, які* розширюють функціональні можливості основної програми. Один із цих пакетів, *Control System Toolbox,* у сполученні з основною програмою дає можливість використовувати MATLAB для аналізу та синтезу систем керування.

При роботі у середовищі MATLAB користувач взаємодіє з комп'ютером за допомогою чотирьох основних об'єктів: інструкцій та змінних, матриць, графічних зображень і скриптів. MATLAB інтерпретує й обробляє вхідні дані у вигляді одного або декількох цих об'єктів.

1.2 Інструкції та змінні

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

У загальному вигляді інструкції представляються в такий спосіб:

```
>> змінна=вираз
```

Командний рядок в MATLAB позначається двома спрямованими вправо стрілками « >> », а у якості оператору присвоювання використовується звичний знак рівності «=».

У виразах використовуються звичайні символи математичних операцій, які наведені у таблиці 1.1. Порядок виконання арифметичних дій можна змінити за допомогою круглих дужок. У круглих дужках записуються аргументи функцій. Найбільш часто зустрічаються функції MATLAB, що наведені у таблиці 1.2.

Таблиця 1.1 - **Символи математичних операцій**

+	Додавання
-	Вирахування
*	Множення
/	Ділення
^	Зведення в ступінь
==	

Таблиця 1.2 – **Функції MATLAB**

`sin(x)`	Синус	`log(x)`	Натуральний логарифм
`asin(x)`	Арксинус	`log10(x)`	Десятковий логарифм
`cos(x)`	Косинус	`log2(x)`	Логарифм з основою 2
`acos(x)`	Арккосинус	`sqrt(x)`	Квадратний корінь
`tan(x)`	Тангенс	`abs(x)`	Абсолютне значення числа
`atan(x)`	Арктангенс	`angle(x)`	Аргумент комплексного числа
`cot(x)`	Котангенс	`real(x)`	Дійсна частина комплексного числа
`acot(x)`	Арккотангенс	`imag(x)`	Мнима частина комплексного числа
`exp(x)`	Експонентна функція	`nthroot(x,n)`	Корінь *n*-го ступеня з *x*

Нижче наведений приклад розрахунку значення змінної *x*:

```
>> x=2+5   <-------- Натиснути < Enter >
x =
      7
>>    (MATLAB готовий до наступної інструкції)
```

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

Якщо після інструкції поставити крапку з комою (`;`), то вивід значення змінної `x` на екран припиняється. Це зручно, коли виконуються якісь проміжні розрахунки, вивід результатів яких на екран не представляє інтересу. Дізнатися значення змінної можна просто ввівши її ім'я:

```
>> x   <-------- Натиснути < Enter >
     7
>>    (MATLAB готовий до наступної інструкції)
```

Помітимо, що якщо у виразі відсутні ім'я змінної та символ «=», то результату автоматично присвоюється ім'я `ans`:

```
>> sin(x) <-------- Натиснути < Enter >
ans =
    0.6570
```

Імена змінних повинні починатися з букви, за якою може бути будь-яка кількість букв або цифр, включаючи символ підкреслення.

MATLAB розрізняє **верхній і нижній регістри,** тому змінні `M` та `m` будуть мати різний сенс.

В MATLAB є декілька змінних із заздалегідь закріпленими за ними іменами. Це `pi`, `Inf`, `NaN`, `i` та `j`.

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

```
>> z = 0/0
Warning: Divide by zero. (Попередження: ділення на нуль)

z =
   NaN
```

`Inf` відповідає $+\infty$, а `pi` — числу π. Змінні *i* та *j* позначають уявну одиницю й використовуються при арифметичних операціях з комплексними числами.

Уникайте застосовувати ці змінні без зайвої необхідності. Звичайно, їх можна використати й для інших цілей. Наприклад, якщо ви звикли позначати уявну одиницю через `j`, то змінну `i` можете зарезервувати для імені цілого числа. Однак, щоб уникнути непорозумінь, намагайтеся не застосовувати без необхідності ці змінні, оскільки й без того існує досить багато інших імен.

1.3 Формати виводу змінних

Змінні `x, z` і `ans` автоматично запам'ятовуються в **робочій області** та можуть бути використані протягом усього сеансу роботи з MATLAB. За допомогою функції `who` можна вивести на екран список всіх змінних, що зберігаються в робочій області:

```
>> who
Your variables are: (Ваші змінні: )
ans  x    z
```

Оскільки MATLAB розрізняє регістри, то функції `who` і `WHO` сприймаються по-різному.

Функція `who` є вбудованою, і набравши `who`, ви одержите список змінних у робочій області. Однак, набравши `WHO` (на верхньому регістрі), ви одержите повідомлення про помилку. Це стосується до всіх функцій MATLAB.

```
>> WHO
??? Capitalized internal function WHO; Caps Lock may be on.
```

(Набрана внутрішня функція WHO; можливо включена клавіша Caps Lock.)

```
>> Who
??? Capitalized internal function Who; Caps Lock may be on.
```

Функція `whos` виводить на екран список змінних у робочій області разом з додатковою інформацією про їхній тип, розмірність і пам'ять, що вони займають:

```
>> whos
  Name      Size                Bytes  Class
  ans       1x1                     8  double array
  x         1x1                     8  double array
  z         1x1                     8  double array
Grand total is 3 elements using 24 bytes
```

Як бачимо, всі 3 змінні в робочій області займають 24 байт. Цю ж інформацію можна одержати, переглянувши вікно (browser)

Workspace (робоча область), що окремо винесена на рисунку 1.1. Якщо у Вас на екрані цього вікна немає, поставте прапорець напроти позиції **Workspace** у меню **View**.

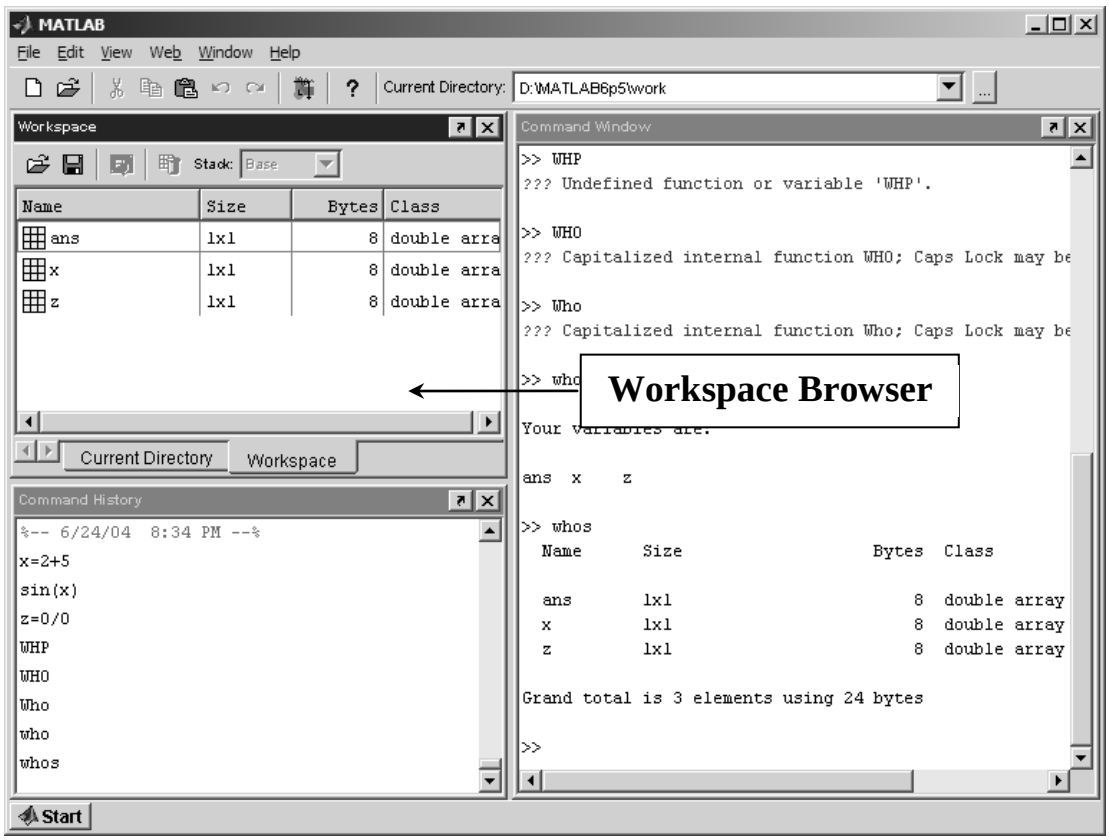


Рисунок 1.1 - Workspace Browser

Змінні можна видалити з робочої області за допомогою функції `clear`. Власне функція `clear` видаляє з робочої області всі дані (змінні й функції); `clear variables` видаляє всі змінні; `clear name1 name2...` видаляє змінні `name1, name2` і т.д. Наприклад,

```
>> clear z
>> who
Your variables are:
ans  x
```

Всі обчислення в MATLAB виконуються з **подвійною точністю.** Однак вивід чисел на екран може здійснюватися в різних форматах. За замовчуванням нецілі числа виводяться із чотирма розрядами після десяткової крапки, але за допомогою функції `format`

можна змінити формат виводу чисел. Основні формати виводу чисел наведені в таблиці 1.3. Якщо задано якийсь певний формат, то він зберігає силу доти, поки не буде змінений.

Таблиця 3 - **Основні формати виводу чисел**

`format` або `format short`	5-ти розрядне число з фіксованою крапкою, використається за замовчуванням
`format long`	15-ти розрядне число з фіксованою крапкою
`format short e`	5-ти розрядне число із плаваючою крапкою
`format long e`	15-ти розрядне число із плаваючою крапкою
`format rational`	представлення у вигляді раціонального дробу

Нагадаємо, насамперед, що формат з фіксованою крапкою має на увазі, що крапка відокремлює цілую частину від дробової. Якщо, наприклад, змінна x = 125,266, то за замовчуванням вона відобразитися у вигляді:

```
>> x=125.266
x =
  125.2660
```

Зверніть увагу, що ми задали нове значення змінної x (125,266 замість 7); воно запам'ятовується, поки ми не задамо яке-небудь інше значення. Приймемо тепер x = 0,08416, у цьому форматі воно буде представлено на екрані так:

```
>> x=0.08416
x =
    0.0842
```

Зберігаються тільки чотири знаки після крапки, причому здійснюється округлення убік меншої помилки. Якщо б x дорівнювало 0,08412, на екрані воно було б представлено у вигляді x = 0,0841.

Формат *e* із плаваючою крапкою уведений для того, щоб, наприклад, при необхідності виводу на екран значення змінної x = 0,0000003522 не набирати багато нулів. Якщо інструкція має вигляд

```
>> x=0.000003522;
>> format short e;x
x =
  3.5220e-006
```

У такий спосіб символіка `e-006` еквівалентна запису 10^6. Потренуємось в освоєнні інших форматів.

```
>> format short; pi
ans =
    3.1416
```
← 4 знаки після крапки

```
>> format long; pi
ans =
   3.141592653589793
```
← 15 знаків після крапки

```
>> format long e; pi
ans =
   3.141592653589793e+00
```
← 15 знаків після крапки

```
>> format short; x=1/2
x =
    0.5000
>> format rational; x
x =
     1/2
```
← раціональний дріб

Переглянути значення змінної в будь-якому форматі можна у вікні **Workspace**. Досить двічі клацнути лівою кнопкою миші на змінній (рисунок 1.2) як з'явиться вікно (рисунок 1.3) редактора **Array Editor** (Редактор матриць). У цьому вікні можна переглянути значення змінної в будь-якому форматі, а також присвоїти їй нове значення. Якщо число в комірці не поміщається, комірку можна розсунути за допомогою миші.

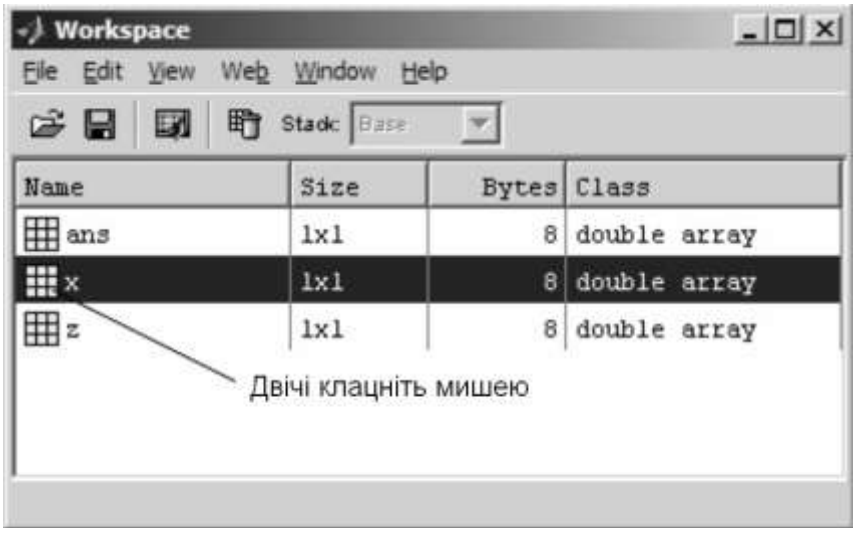


Рисунок 1.2 - Відкриття вікна **Array Editor**

Рисунок 1.3 - Вікно **Array Editor**

Крім розглянутих, MATLAB має ще безліч вбудованих функцій. Повний їхній перелік можна знайти в інструкції для користувача MATLAB або звернутися для цього до помічника **On-line Help.**

1.4. Операції рядкового редагування

При роботі з MATLAB у командному режимі діє найпростіший рядковий редактор. Його команди перераховані в таблиці 1.4.

Таблиця 1.4 - **Команди рядкового редактора MATLAB**

Комбінація клавіш	**Призначення**
→	Переміщення курсору вправо на один символ
←	Переміщення курсору вліво на один символ
Ctrl + →	Переміщення курсору вправо на одне слово
Ctrl + ←	Переміщення курсору вліво на одне слово
Home	Переміщення курсору в початок рядка
End	Переміщення курсору в кінець рядка
↑ і ↓	Перелистування попередніх команд нагору або вниз для підстановки в рядок вводу
Del	Стирання символу праворуч від курсору
←	Стирання символу ліворуч від курсору
Ctrl + k	Стирання до кінця рядка
Esc	Очищення рядка вводу
Ins	Вмикання/вимикання режиму вставки
PgUp	Перегортання сторінки сесії вгору
PgUn	Перегортання сторінки сесії вниз

Зверніть особливу увагу на застосування клавіш ↑ і ↓. Вони використовуються для підстановки після маркера рядка вводу >> раніше введених рядків, наприклад для їхнього виправлення, дублювання або доповнення й запозичені з більше ранніх версій MATLAB. Для цих цілей можна також використовувати вікно **Command History**, що за замовчуванням розташовано в лівому нижньому кутку вікна MATLAB (рисунок 1.4). У цьому випадку необхідну команду досить перетягнути мишкою в командний рядок.

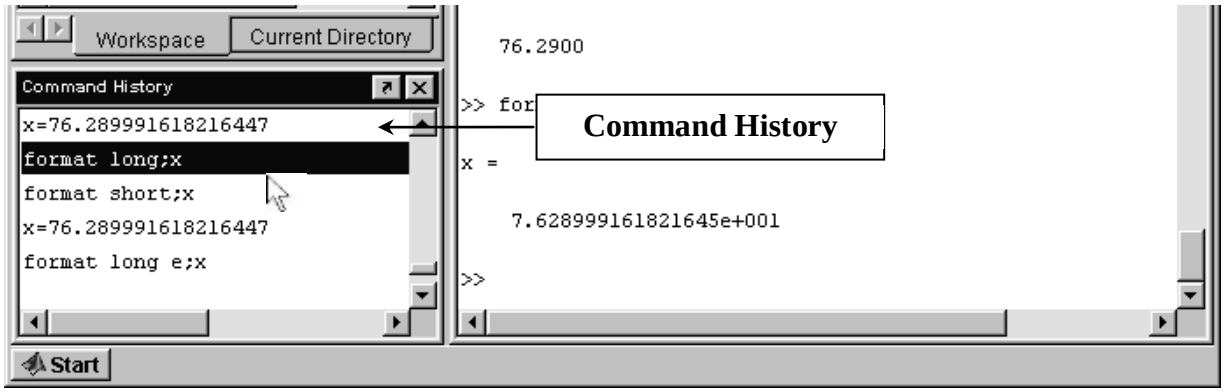


Рисунок 1.4 - Вікно **Command History**

1.5 Завдання на лабораторну роботу

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*) $\lg(-150)$;

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

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

3. Виведіть 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.

5. *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.

Зміст звіту.

1. Символи математичних операцій та основні функції MATLAB.
2. Основні формати виводу чисел з чисельними прикладами.
3. Лістинг команд для виконання завдань на лабораторну роботу.

ЛАБОРАТОРНА РОБОТА №2

ВЕКТОРИ ТА МАТРИЦІ В MATLAB

Мета роботи: навчитись задавати матриці та вектори у MATLAB та виконувати основні операції з матрицями.

2.1 Загальні положення

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

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

З вікон **Workspace** або **Array Editor** (рисунок 2.1) добре видно, що змінна *x* і непойменована змінна, що дорівнює квадрату *x* та одержала ім'я `ans`, трактуються як матриці розміром 1×1.

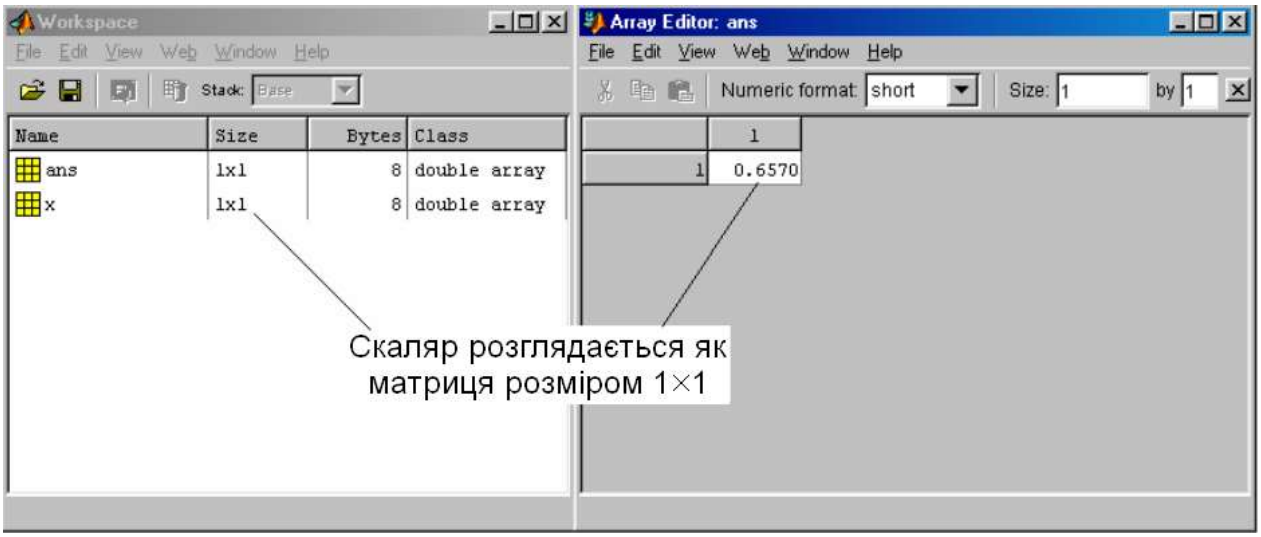


Рисунок 2.1 - Вікна **Workspace** і **Array Editor**

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.

2.2 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
```

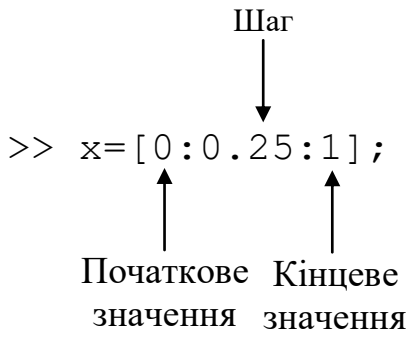
Елементи матриці можна вводити у вигляді арифметичних виразів, що містять будь-які доступні у MATLAB функції. Наприклад,

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

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
```

2.3 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 середньоквадратичним 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.

2.4 Операції з матрицями

До основних матричних операцій відносяться операції, що представлені в таблиці 2.1.

Таблиця 2.1 - **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

$$C(i,j)=\sum_{r=1}^{m}\left(a_{ir}\cdot b_{rj}\right),\quad i=1,n;\ \ j=1,k\,. \tag{2.1}$$

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**:

$$\mathbf{A}^{-1}\cdot\mathbf{A}=\mathbf{A}\cdot\mathbf{A}^{-1}=\mathbf{E}. \tag{2.2}$$

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 $\mathbf{B}^{-1}$:

$$\mathbf{C}\cdot\mathbf{B}^{-1} = \mathbf{A}\cdot\mathbf{B}\cdot\mathbf{B}^{-1}. \quad (2.3)$$

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

$$\mathbf{A} = \mathbf{C}\cdot\mathbf{B}^{-1}, \quad (2.4)$$

або з використанням символу правого ділення:

$$\mathbf{A} = \mathbf{C}/\mathbf{B}. \quad (2.5)$$

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 $\mathbf{A}^{-1}$:

$$\mathbf{A}^{-1}\cdot\mathbf{C} = \mathbf{A}^{-1}\cdot\mathbf{A}\cdot\mathbf{B}. \quad (2.6)$$

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

$$\mathbf{B} = \mathbf{A}^{-1}\cdot\mathbf{C}, \quad (2.7)$$

або з використанням символу лівого ділення

$$\mathbf{B} = \mathbf{A}/\mathbf{C}. \qquad (2.8)$$

Ліве ділення матриці **C** на квадратну матрицю **A**, тобто **A\C**, визначає матрицю **B** таку, що **A**·**B** = **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:

$$\begin{cases} a_{11} \cdot x_1 + a_{12} \cdot x_2 = b_1 \ ; \\ a_{21} \cdot x_1 + a_{22} \cdot x_2 = b_2 \ . \end{cases} \qquad (2.9)$$

або в матричній формі:

$$\mathbf{A} \cdot x = \mathbf{B}. \qquad (2.10)$$

де

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

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:

$$\left(a_{11}-a_{21}\cdot\frac{a_{11}}{a_{21}}\right)\cdot x_1+\left(a_{12}-a_{22}\cdot\frac{a_{11}}{a_{21}}\right)\cdot x_2=b_1-b_2\cdot\frac{a_{11}}{a_{21}}. \quad (2.11)$$

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:

$$\begin{cases} a_{11}\cdot x_1+a_{12}\cdot x_2=b_1\ ; \\ 0\ \cdot x_1+\left(a_{12}-a_{22}\cdot\dfrac{a_{11}}{a_{21}}\right)\cdot x_2=b_1-b_2\cdot\dfrac{a_{11}}{a_{21}}\ . \end{cases} \quad (2.12)$$

Із другого рівняння цієї системи одержимо:

$$x_2=\frac{b_1-b_2\cdot\dfrac{a_{11}}{a_{21}}}{a_{12}-a_{22}\cdot\dfrac{a_{11}}{a_{21}}}, \quad (2.13)$$

а з першого рівняння:

$$x_1=\frac{b_1-a_{12}\cdot x_2}{a_{11}}=\left(b_1-a_{12}\cdot\frac{b_1-b_2\cdot\dfrac{a_{11}}{a_{21}}}{a_{12}-a_{22}\cdot\dfrac{a_{11}}{a_{21}}}\right)\cdot\frac{1}{a_{11}}. \quad (2.14)$$

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.

2.5 Визначення характеристик матриць

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·(-0,8246) + 2·(0,5658) = -0,3723·(-0.8246),
0,307 = 0,307

2) 3·(-0,8246) + 4·(0,5658) = -0,3723·(0,5658),
0,2106 = 0,2106

2.6 Завдання для лабораторної роботи

1. Розгляньте дві матриці:

$$\mathrm{A}=\begin{pmatrix}-1 & 5\\ \pi & \sqrt{2}\end{pmatrix},\ \mathbf{B}=\begin{pmatrix}-1+0,5j & 3\pi\\ 10 & 7\end{pmatrix}.$$

За допомогою MATLAB виконайте наступні операції:

а) $\mathbf{A}+\mathbf{B}$; *в)* $\mathbf{A}\cdot\mathbf{B}$; *д)* $\mathbf{A}^{\mathrm{T}}$;

б) $\mathbf{B}^{-1}$; *г)* $\mathbf{A}^2$; *е)* $\mathbf{A}^2+\mathbf{B}^2-\mathbf{AB}$.

2. Вирішите систему лінійних алгебраїчних рівнянь, записавши їх у матричній формі:

а)

$$\begin{cases}2x_1+3x_2+4x_3=20,\\ -x_1-x_2+3x_3=10,\\ x_1+x_2-x_3=2.\end{cases}$$

в)

$$\begin{cases}-9x_1+15x_2+14x_3=20,\\ 4{,}5x_1-5x_2+10{,}5x_3=10,\\ -4{,}5x_1+10x_2-3{,}5x_3=2.\end{cases}$$

б)

$$\begin{cases}x_1-2x_2+3x_3=-1,\\ -2x_1+3x_2-x_3=2,\\ -x_1+x_2+x_3=0.\end{cases}$$

г)

$$\begin{cases}3x_1+2x_2-x_3=4,\\ x_1-3x_2+2x_3=5,\\ 2x_1+x_2-3x_3=3.\end{cases}$$

3. Побудуйте наступні матриці:

а) одиничну (4×4) матрицю;

б) нульову матрицю;

в) матрицю розміром 5×4 з випадкових чисел, рівномірно розподілених у діапазоні від 0 до 1.

4. Компоненти вектора **x** приймають значення від 0 до 20 з шагом 2. За допомогою операторів циклу `while…end` сформуйте квадратну матрицю **A**, головна діагональ якої заповнена компонентами вектора **x**, а всі інші елементи дорівнюють 0.

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

6. 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.

7. 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.

Зміст звіту.

1. Методи вводу матриць у MATLAB.
2. Команди та приклади створення матриць зі специфічними властивостями.
3. Оператори здійснення операцій з матрицями
4. Команди та приклади визначення характеристик матриць.
5. Лістинг команд для виконання завдань на лабораторну роботу.

ЛАБОРАТОРНА РОБОТА № 3

ПОБУДОВА ГРАФІКІВ В MATLAB

Мета роботи: навчитися будувати та редагувати графіки засобами MATLAB.

3.1 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*, оскількі 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.

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 рисунку 3.1 представлене вікно Figure No. 1, а на рисунку 3.2 види вікон: **Workspace, Command History** і **Command Window**.

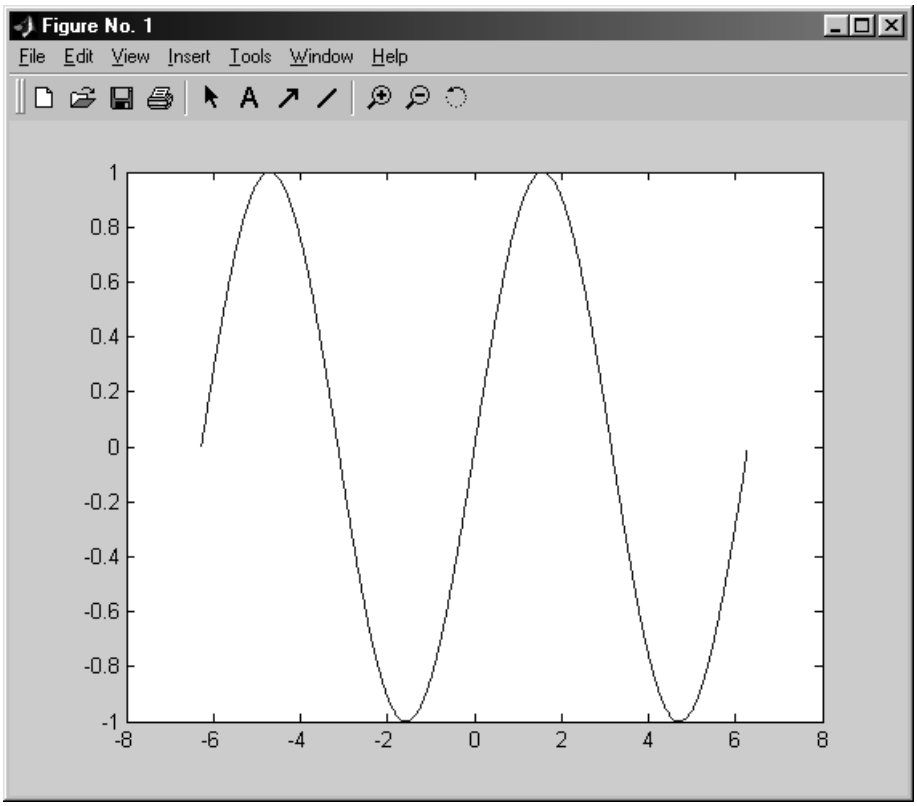


Рисунок 3.1 - Графічне вікно Figure No. 1

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 но-

вого графічного вікна.

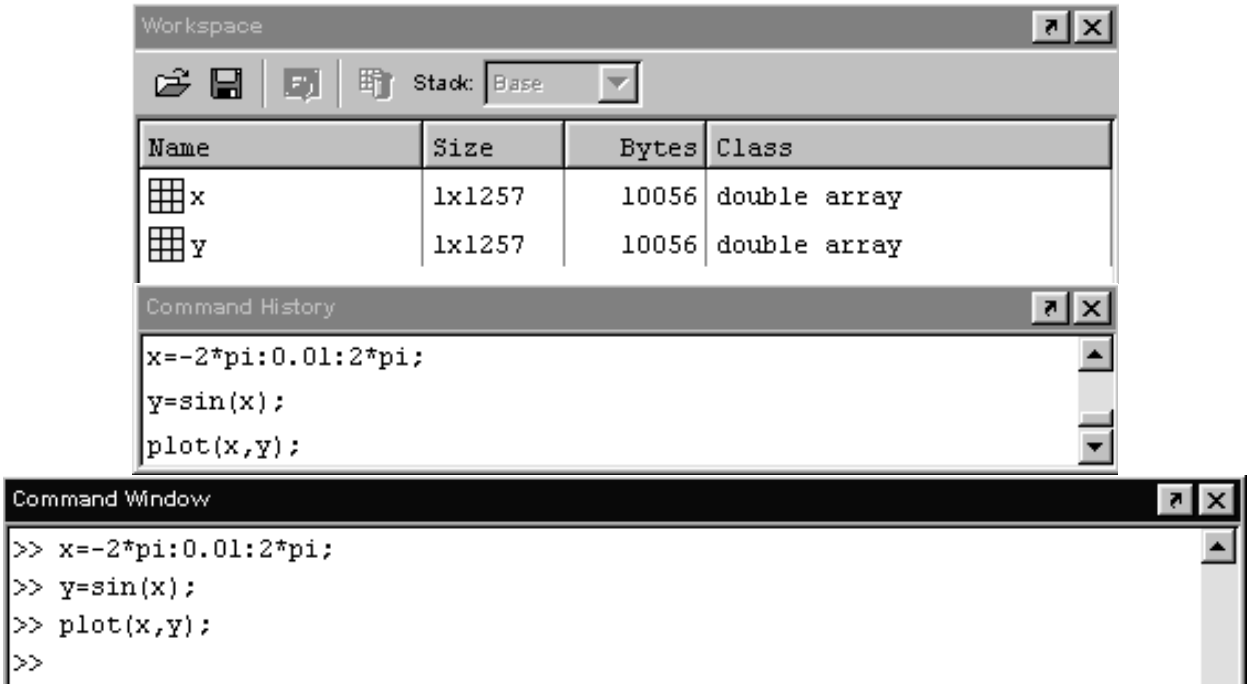


Рисунок 3.2 - Вікна **Workspace**, **Command History** і **Command Window** після виконання команд

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 рисунку 3.3 представлене графічне вікно, що відкривається в результаті виконання наведеної вище програми.

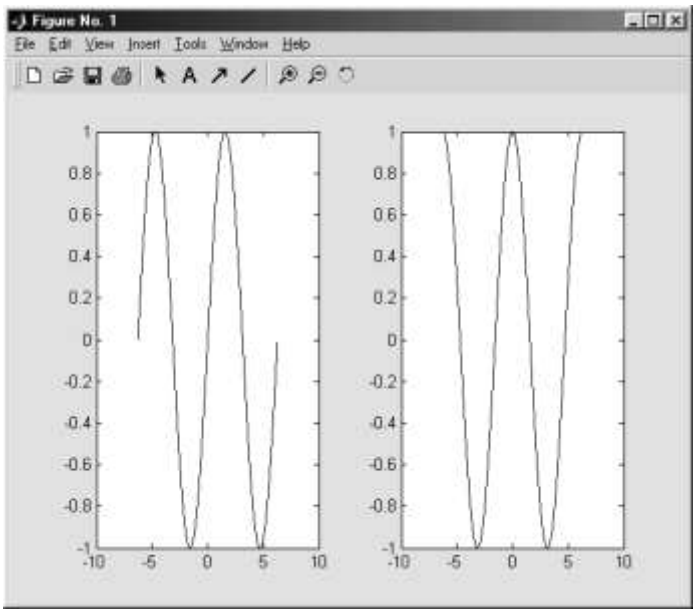


Рисунок 3.3 - Вид графічного вікна Figure No. 1
після застосуванням команди `subplot`

3.2 Основи форматування графіків

Для заdдання формату графіків (визначення типу ліній, встановлення звичайного, напівлогарифмічного або логарифмічного масштабу), а також їхнього оформлення (додавання заголовків і пояснюючого тексту, позначення осей) існують групи графічних функцій, деякі з яких уже зустрічалися в прикладах. Так для позначення осей використовуються функції `xlabel` та `ylabel`, а функція `grid on` включає координатну сітку. За допомогою спеціальних рядкових констант можна задавати кольори й тип лінії графіка. Деякі із цих рядкових констант наведені в таблиці 3.1.

Таблиця 3.1 - **Рядкові константи для форматування графіків**

Кольори лінії		Тип крапки		Тип лінії	
Y	Жовтий	.	Крапка	-	Суцільна
R	Червоний	O	Окружність	:	Подвійний пунктир
B	Синій	X	Хрест	-.	Штрих-пунктир
G	Зелений	+	Плюс	-	Штрихова

Ви можете вводити до графіків спеціальні й математичні символи, використовуючи набір символів TeX (TeX - це торгівельна марка Американського Математичного Суспільства). Деякі із символів TeX наведені в таблиці 3.2. Всім послідовностям літер у TeX повинен передувати символ « \ ».

Таблиця 3.2 – **Спеціальні й математичні символи TeX**

Текст MATLAB	Символ	Текст MATLAB	Символ	Текст MATLAB	Символ
\alpha	α	\nu	ν	\infty	∞
\beta	β	\pi	π	\pm	$\pm$
\gamma	γ	\rho	ρ	\leq	$\leq$
\delta	δ	\sigma	σ	\geq	$\geq$
\epsilon	ε	\tau	τ	\approx	$\approx$
\zeta	ζ	\phi	φ	\in	$\in$
\theta	θ	\omega	ω	\div	$\div$
\lambda	λ	\Sigma	Σ	\neq	$\neq$
\mu	μ	\Omega	Ω	\oslash	$\oslash$

Крім того, можна змінити накреслення символів за допомогою наступних модифікаторів:

- `\bf` - напівжирний шрифт
- `\it` - курсив
- `\rm` - звичайний шрифт
- `\fontname` - визначає назву сімейства шрифтів
- `\fontsize` - визначає розмір шрифту

Для позначення нижніх і верхніх індексів використовуються відповідно символи «_» та «^».

Однак засоби графіки нових версій MATLAB (починаючи з 6.0) дозволяють виконувати всі ці операції безпосередньо в графічних вікнах. Так, наприклад, у меню **Edit** вікна графіки поряд із стандартними операціями роботи з буфером є наступні команди:

- Copy Figure - копіювання фігури (графіка) у буфер;
- Copy Options - копіювання опцій графіка;
- Figure Properties - вивід вікна властивостей графіка;
- Axes Properties - вивід вікна властивостей осей графіка;
- Current Object Properties -вивід вікна властивостей поточного об'єкта.

Для виводу властивостей графіків, їхніх осей і поточних об'єктів використовується вікно властивостей графіків з відповідними вкладками. За допомогою панелей інструментів, призначення кнопок яких цілком очевидне, можна відкривати порожнє вікно графіки, завантажувати у вікно графік і виводити його до друку, міняти режим редагування графіка, наносити на графіки напис, стрілки й лінії, змінювати масштаб графіка й обертати його мишею. Розглянемо деякі з цих дій більш докладно.

Для редагування властивостей якого-небудь об'єкта (графіка або його вікна) потрібно цей об'єкт спочатку виділити. Для цього треба натиснути кнопку **Edit Plot** (Редагувати графік) у панелі інструментів вікна графіка й, клацнувши по потрібному об'єкті лівою кнопкою миші виділити його. Подвійним натисканням лівої кнопки миші викликається вікно форматування виділеного об'єкта. Це ж вікно можна викликати з контекстного меню (натисканням правої кнопки миші).

Наприклад, указавши в режимі редагування мишею на лінію графіка (і двічі швидко клацнувши лівою клавішею), можна побачи-

ти вікно форматування ліній графіка, що показане на рисунку 3.4 ліворуч. Виділений графік показано праворуч. Зверніть увагу на появу на лінії графіка ряду чорних квадратиків - вони використовуються для вказівки курсором миші саме на лінію графіка, а не на інші об'єкти.

У цьому вікні відкрита головна для операцій форматування вкладка - **Style** (Стиль). Вона встановлює стиль відображення лінії, тобто її вид (наприклад, суцільна лінія або пунктирна), ширину й кольори, а також параметри маркерів, що відзначають опорні точки графіків.

Кнопка **Apply** (Застосувати) дозволяє застосувати зроблені установки до графіка до закриття вікна діалогу. Кнопка **ОК** підтверджує зроблені установки та закриває вікно діалогу. Призначення інших кнопок очевидне.

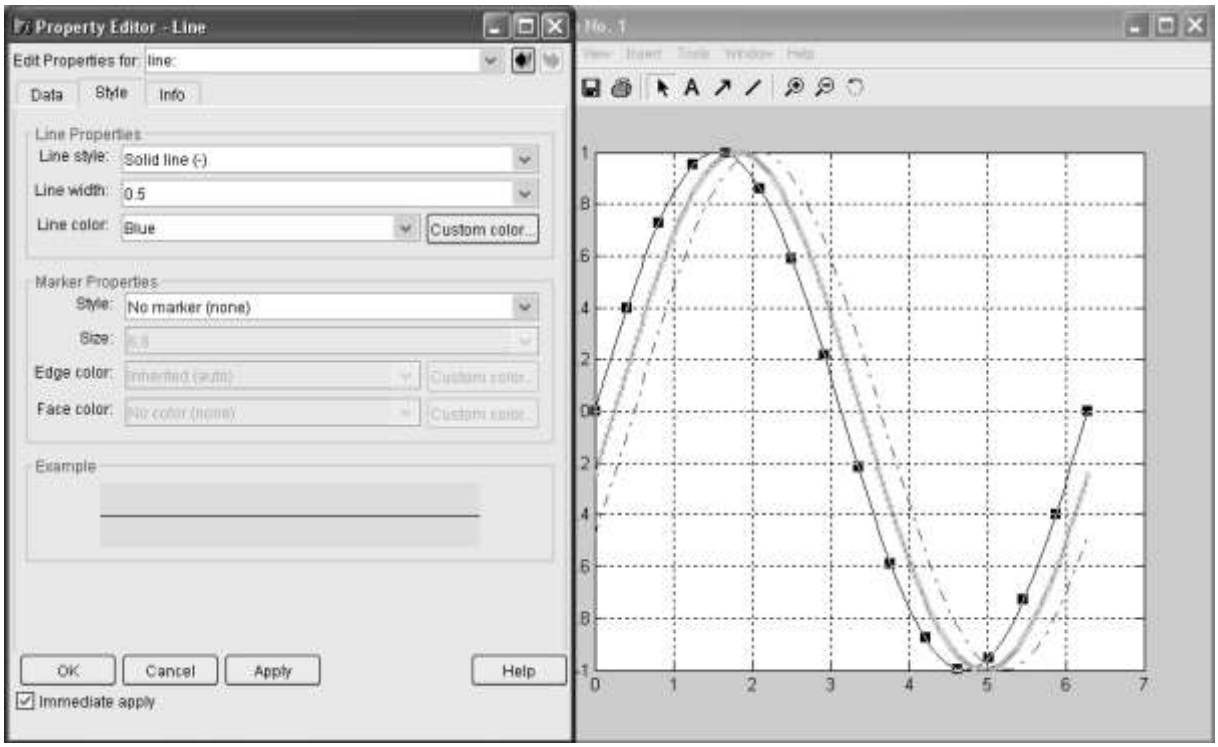


Рисунок 3.4 - Вікно графіка (праворуч) і вікно форматування ліній (ліворуч)

Аналогічним чином виконується форматування й інших об'єктів графіків. Наприклад, указавши курсором миші на осі графіків (на них теж є мітки у вигляді чорних квадратиків) і двічі клацнувши лівою клавішею миші, можна побачити появу вікна **Property Editor**

(Редактор властивостей, Графічний редактор властивостей), настроєного на форматування осей (рисунок 3.5).

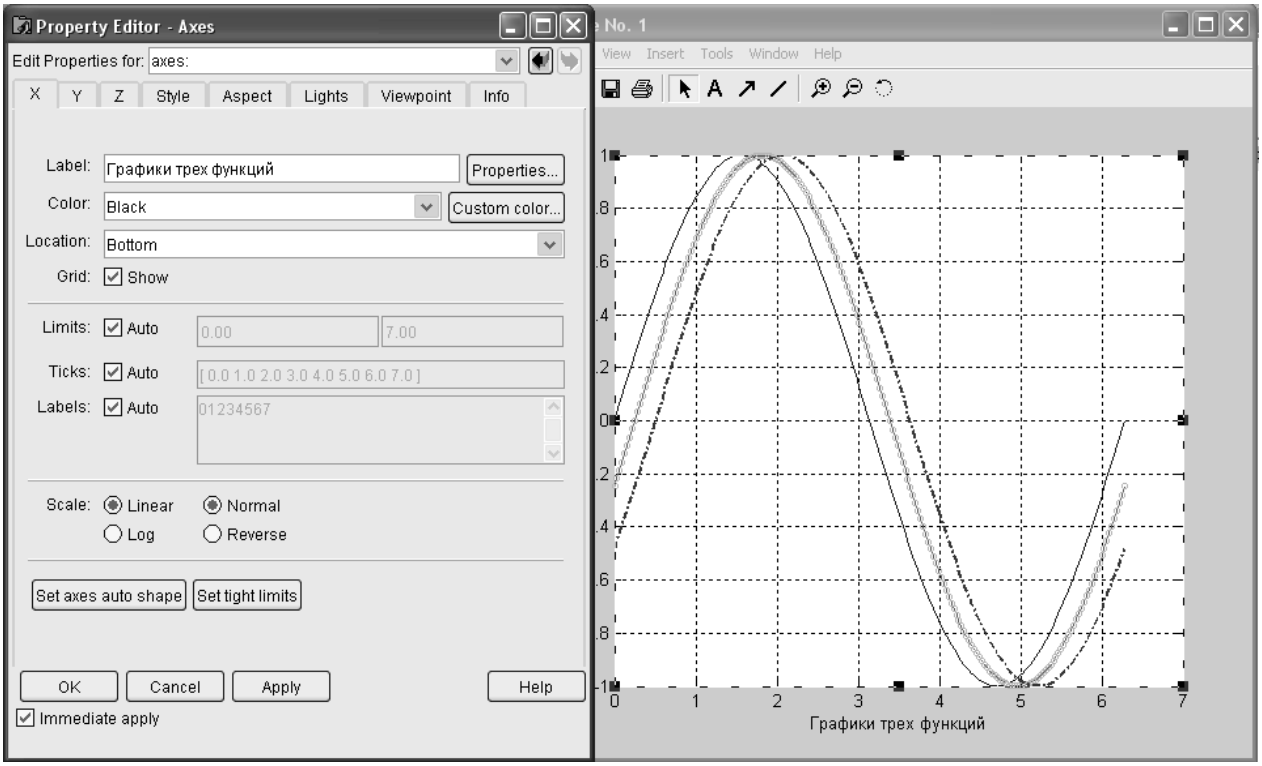


Рисунок 3.5 - Приклад форматування осей графіка

Вікно графічного редактора властивостей графіка має безліч вкладок, настроювання яких досить очевидні. Наприклад, ви можете задати лінійний або логарифмічний масштаб осей (вкладка **Scale** (Масштаб), відкрита на рисунку 3.5), нормальний або інверсний напрямок осей (X, Y, а у випадку тривимірних графіків й Z), показ сітки (параметр **Grid Show**), змінити стиль осей і кольори фону (вкладка **Style** (Стиль)), нанести біля осей напис (вкладка **Label** (Ярлик)) та інше. Нагадаємо, що всі описані вище дії можна виконати, вибравши меню **Edit**.

Додатково на графік можна нанести напис за допомогою кнопки **Insert Text** A. Місце напису фіксується натисканням миші. На рисунку 3.6 показаний відформатований графік із текстовим блоком. Там же показане контекстне меню правої клавіші миші, що пояснює вибір розміру символів напису (і інші можливості цього меню). Отриманий в такий спосіб напис можна виділити й перенести мишею в будь-яке інше місце.

Додатково можна змінити розміри графіка (див. меню **Tools** (Інструменти) і його команди **Zoom In** (Збільшити) і **Zoom Out** (Зменшити)), почати обертання графіка мишею (команда **Rotate** 3D), додати відрізок прямої або інший графічний примітив (підменю **Add**) і підключити до графіка *легенду* — пояснення у вигляді відрізків ліній з довідковими написами, розташоване всередині графіка або біля нього. Оскільки наш графік містить три криві, то легенда являє собою позначення цих трьох ліній у правому верхньому куті рисунка (рисунок 3.6). Кожна лінія має ті ж кольори, що й на графіку (і той же стиль). Можливий також вивід шкали кольорів.

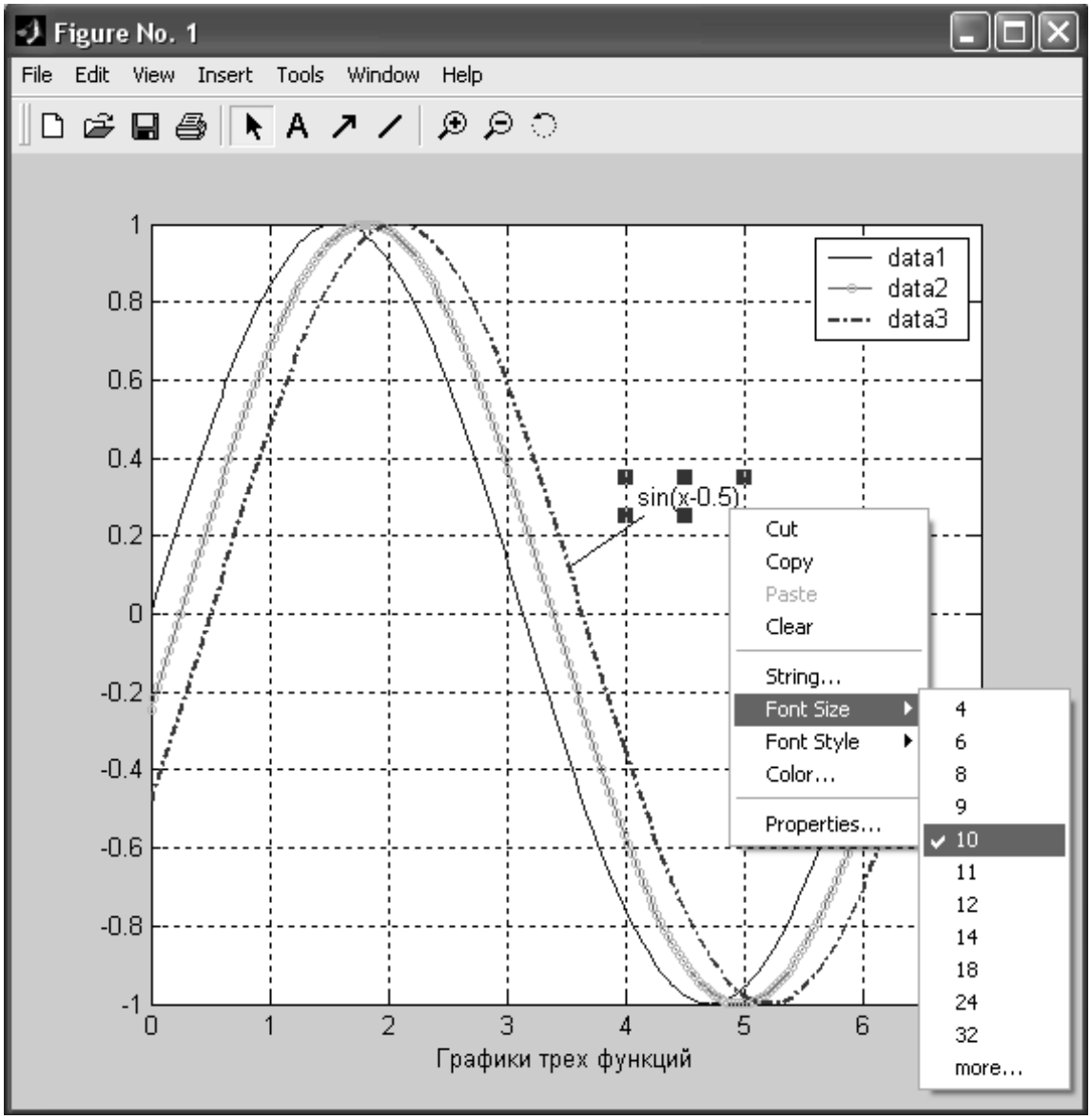


Рисунок 3.6 - Нанесення напису на графік

3.3 Скрипти та m-файли

Дотепер наше спілкування з MATLAB відбувалося на рівні командних рядків. Ми вводили інструкції й функції, MATLAB інтерпретував їх і виконував відповідні дії. Такий режим гарний у тих випадках, коли сеанс роботи в середовищі MATLAB є коротким і не містить операцій, що повторюються. Однак дійсна міць MATLAB стосовно до аналізу й синтезу систем керування виявляється в можливості виконання послідовності команд, записаних у вигляді файлу. Такі файли називаються **m-файлами,** тому що їхні імена мають вигляд **ім'я_файлу.m**. Одним із видів m-файлу є **скрипт.**

Скрипти - це звичайні текстові файли, які створюються за допомогою текстового редактора. На додаток до m-файлів, що поставляються разом з MATLAB і пакетами прикладних програм, ви можете розробити власні скрипти для вирішення конкретних завдань.

Для підготовки, редагування й налагодження m-файлів служить спеціальний редактор, який можна викликати вибравши **NEW** → **m-file** з меню **FILE** або просто нажавши кнопку 🗋 на панелі інструментів. Редактор m-файлів має своє меню та панель інструментів (рисунок 3.7).

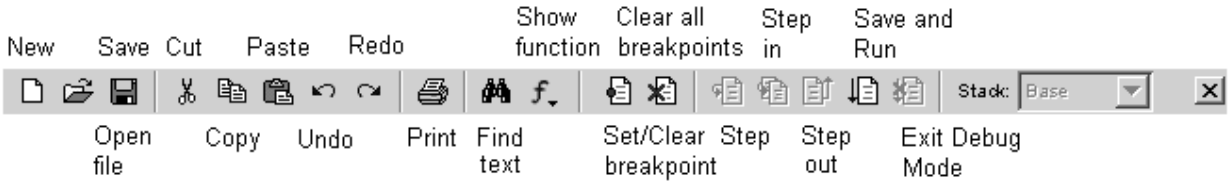


Рисунок 3.7 - Панель інструментів редактора

Призначення кнопок панелі інструментів редактора наступне:

- New - створення нового m-файлу;
- Open - вивід вікна завантаження файлу;
- Save - запис файлу на диск;
- Print - друк змісту поточного вікна редактора;
- Cut - вирізання виділеного фрагмента й перенесення його в буфер;
- Copy - копіювання виділеного об'єкта в буфер;
- Paste - розміщення фрагмента з буфера в позиції текстового курсору;

- Undo - скасування попередньої операції;
- Redo - повторення 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 кроку трасування;
- Step In - покрокове трасування із заходом у m-файли, що викликаються;
- Step Out - покрокове трасування без заходу у m-файли, що викликаються;
- Save and Run - запуск і збереження;
- Exit Debug Mode - вихід із режиму налагодження.

Редактор m-файлів не тільки спрощує введення великих програм і дозволяє зберігати їх на диску. Він також є отладчиком і піддає текст m-файлу ретельній синтаксичній перевірці, у ході якої виявляються та відсіваються багато помилок користувача. Для зручності синтаксичної перевірки програми користувачем, текст m-файлу виділяється різними кольорами:

- синім - ключові слова мови програмування;
- чорним - оператори, константи й змінні;
- зеленим - коментарі після знака %;
- коричневим - символьні змінні (в апострофах);
- червоним - синтаксичні помилки.

Редактор має й інші важливі засоби налаштування - він дозволяє встановлювати в тексті файлу спеціальні мітки, іменовані точками переривання (breakpoints). При їхньому досягненні обчислення припиняються, і користувач може оцінити проміжні результати обчислень (наприклад, значення змінних), перевірити правильність виконання циклів тощо. Для зручності роботи з редактором рядка програми в ньому нумеруються. Редактор є багатовіконним. Вікно кожної програми оформлюється як вкладка.

Редактор дозволяє легко переглядати значення змінних. Для цього досить підвести до імені змінної курсор миші й затримати його - з'явиться спливаюча підказка з ім'ям змінної та її значенням.

Розглянемо тепер приклад створення найпростішого m-файлу. Припустимо, що нам необхідно побудувати графік функції $y(t) =$

sinαt, де α - параметр, що необхідно змінювати. Тоді в редакторі ми записуємо скрипт і привласнюємо йому, скажемо, ім'я **plotdata.m**. Потім вводимо в командний рядок значення α = 10. Для виконання скрипта потрібно ввести його ім'я **plotdata** у командний рядок або натиснути кнопку **Save and Run** на панелі інструментів редактора. При цьому для побудови графіка скрипт використовує останнє значення α з робочої області. Після отримання результату ви можете ввести інше значення α у командний рядок (наприклад α = 50) і виконати скрипт ще раз, щоб отримати новий графік.

Скрипт повинен бути добре забезпечений **коментарями,** що починаються із символу %. Забезпечте скрипт коментарями**, що** містять деякі відомості про його призначення

```
%Скрипт побудови графіка функції y=sin(alpha*t).
%Значення alpha повинно бути задано у робочій
% області перед запуском скрипту на виконання.
t=[0:0.01:1];
y=sin(alpha*t);
plot(t,y)
xlabel(Час, c')
ylabel('y(t)=sin(\alpha t)')
grid on
```

Щоб вивести коментарі на екран, використайте функцію `help`, як показано нижче.

```
>> help plotdata
 Скрипт побудови  графіка функції y=sin(alpha*t).
 Значення alpha повинно бути задано у робочій
 області перед запуском скрипту на виконання.
```

3.4 Завдання для лабораторної 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 – суцільна 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. Побудуйте графіки функції $y(x) = e^{-0,5x}\sin\omega x$, де $\omega = 10$ рад/с та $0 \le x \le 10$. Сформуйте вектор x за допомогою двокрапки із кроком 0,1.

6. Розробіть скрипт MATLAB для побудови графіка функції $y(x) = (4/\pi)\cdot\cos(\omega x) + (4/9\pi)\cdot\cos(3\omega x)$.

Включіть до скрипту коментар із описом завдання та переконайтеся, що функція `help` виводить цей коментар на екран. Перевірте виконання скрипту при $\omega = 1$, 3 та 10 рад/с. Позначте вісь x як «*Time* (*sec)*», а вісь y як «$y(x)$».

7. Розгляньте функцію $y(x) = 10+5e^{-x}\cos(\omega x+0,5)$. Розробіть скрипт, що дозволяє побудувати в одних осях графіки $y(x)$ для трьох значень $\omega = 1$, 3 та 10 рад/с при $0 \le x \le 5$ с. Графічне зображення повинне мати наступні атрибути:

Заголовок	$y(x) = 10+5\exp(-x)\cos(\omega x+0,5)$
Позначення осі x	*Time (sec)*
Позначення осі y	$y(x)$
Тип лінії	$\omega = 1$: суцільна лінія
	$\omega = 3$: штрихова лінія
	$\omega = 10$: пунктирна лінія
Сітка	Нанесена

8. На рисунку 3.8 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

$$y(t) = \frac{y_0}{\sqrt{1-\zeta^2}} \cdot e^{-\zeta\omega_0 t} \sin(\omega_0\sqrt{1-\zeta^2}\,t + \arccos\zeta, \qquad (3.1)$$

де y_0 – початкове відхилення ω_0 – власна частота коливань системи, $\omega_0 = \sqrt{c/M}$, ζ - безрозмірний коефіцієнт загасання (демпфування).

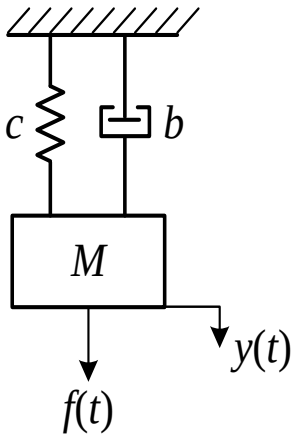


Рисунок 3.8 - Система маса-пружина з демпфуванням

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)* $c = 5$; $M = 1$, $b = 1$.

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 (3.1) є векторами, то для їх поелементного множення необхідно використовувати операцію множення з точкою (.*).

9. 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 «x», ширина лінії – 2;

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

Зміст звіту.

1. Команди та приклади побудови графіків.

2. Таблиці з символами для форматування графіків та текстових надписів.

3. Лістинг команд та скриптів для виконання завдань на лабораторну роботу.

4. Результати виконання завдань.

ЛАБОРАТОРНА РОБОТА №4

ПРЕДСТАВЛЕННЯ МОДЕЛЕЙ СИСТЕМ КЕРУВАННЯ У ПАКЕТІ MATLAB

Мета роботи: придбати практичні навички з завдання та перетворення моделей лінійних систем автоматичного керування, а також з побудови перехідних та частотних характеристик систем керування.

4.1 Загальні положення

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

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

- за допомогою передатних 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 використаються такі функції MATLAB як `roots`, `tf`, `series`, `parallel`, `feedback`, `pole`, `zero`, `poly`, `conv`, `polyval`, `minreal`, `pzmap`, `step` та інші.

4.2 Завдання передаточних функцій у MATLAB

Однією з найбільш поширених форм представлення математичної моделі системи керування є передаточні функції. 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(рисунок 4.1)

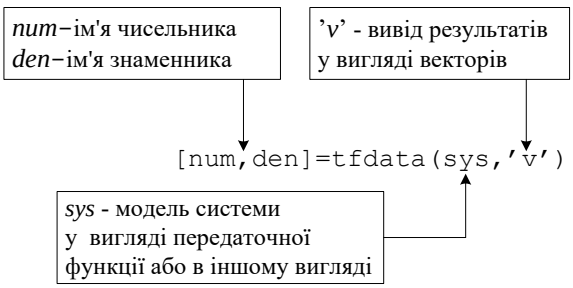


Рисунок 4.1 – Синтаксис команди `tfdata`

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

4.2 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);
```

4.3 Завдання частотної передаточної функції

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.
```

4.4 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 (рисунок 4.2).

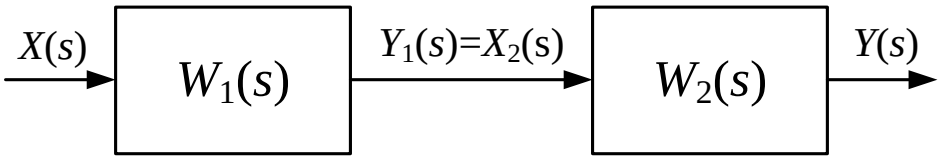


Рисунок 4.2 - Послідовне з'єднання ланок

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 (рисунок 4.3).

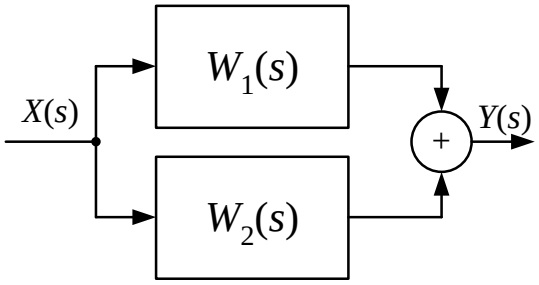


Рисунок 4.3 - Паралельне з'єднання ланок

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 виразом

$$W(s) = \frac{W_1(s)}{1 \pm W_1(s)W_{зз}(s)}, \qquad (4.1)$$

де знак «+» ставится у випадку негативного зворотного зв'язку (рисунок 4.4), 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.

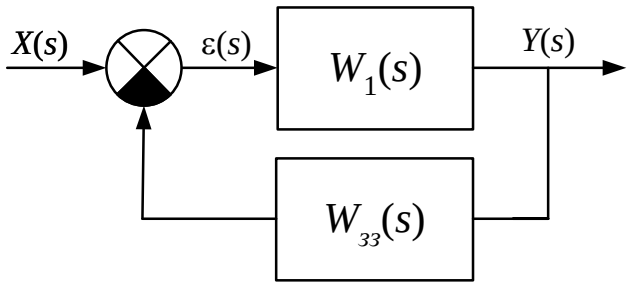


Рисунок 4.4 - Система керування зі зворотним зв'язком

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 рисунку 4.5 дорівнюють

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

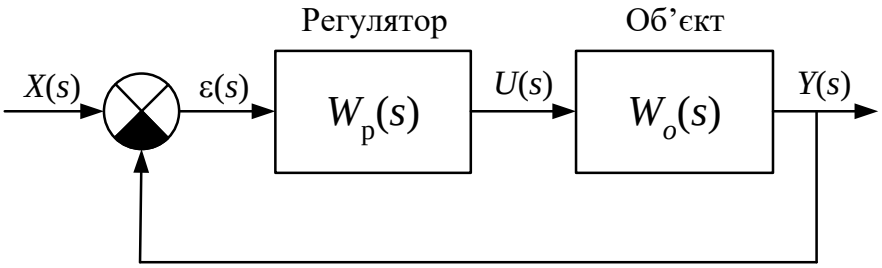


Рисунок 4.5 - Структурна схема зі зворотним зв'язком

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.

4.5. Побудова перехідної функції системи

Після завдання моделі системи керування звичайно необхідно провести її аналіз. Для цього потрібно побудувати часові (перехідну й імпульсну перехідну функції) і частотні (амплітудно-частотну, фазово-частотну, амплітудно-фазову) характеристики, визначити розташування полюсів і нулів та ін. Як будувати імпульсну перехідну функцію системи, її частотні характеристики, картину розташування полюсів і нулів ми розглянемо більш докладно в наступних лабораторних роботах, при визначенні стійкості та якості роботи систем. Зараз же зупинимося на побудові перехідної та імпульсної перехідної функцій, як найбільш наочних, і, тому, найчастіше використовуваний характеристик.

Реакцію на східчастий сигнал можна досліджувати за допомогою функції `step`. Якщо єдиною метою досліджень є отримання графіка перехідної функції $h(t)$, то команду `step` можна використати без зазначення аргументів у лівій частині інструкції. Графік буде отриманий автоматично із зазначенням змінних за осями координат (рисунок 4.6):

```
>> W=tf(1,[0.2 1]);
>> step(W)
```

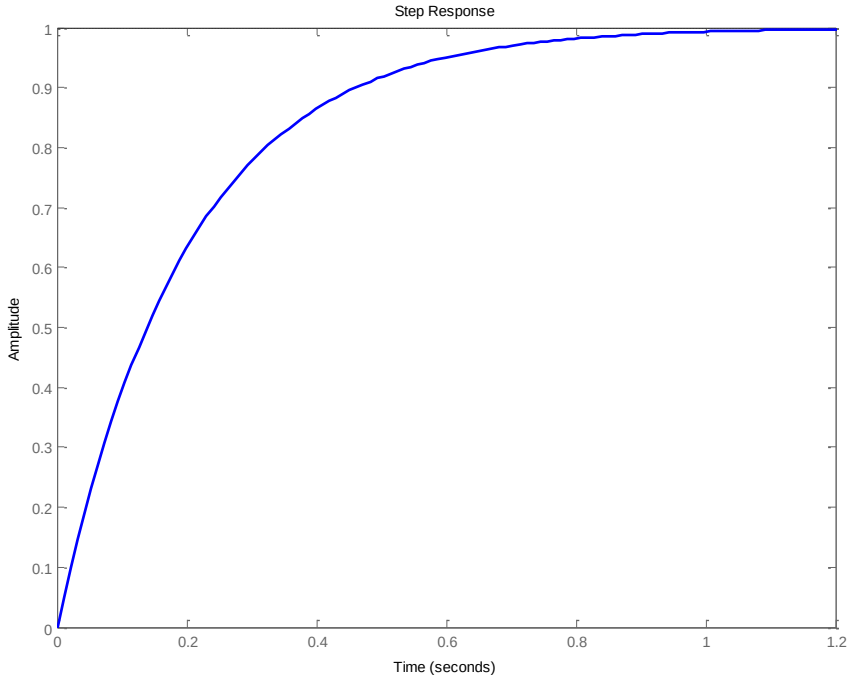


Рисунок 4.6 – Результат застосування функції `step`

Якщо ж $h(t)$ необхідно для якихось інших цілей, крім побудови графіка, то функцію `step` треба використати із зазначенням всіх аргументів у лівій частині інструкції, після чого висновок графіка здійснюється за допомогою функції `plot`(рисунок 4.7).

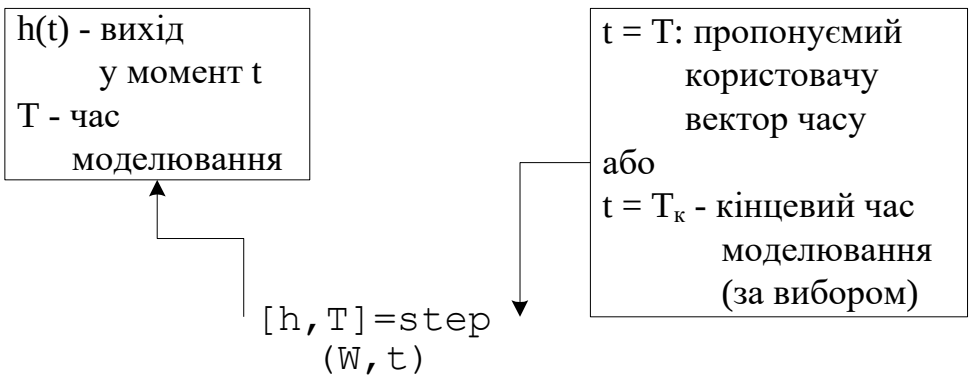
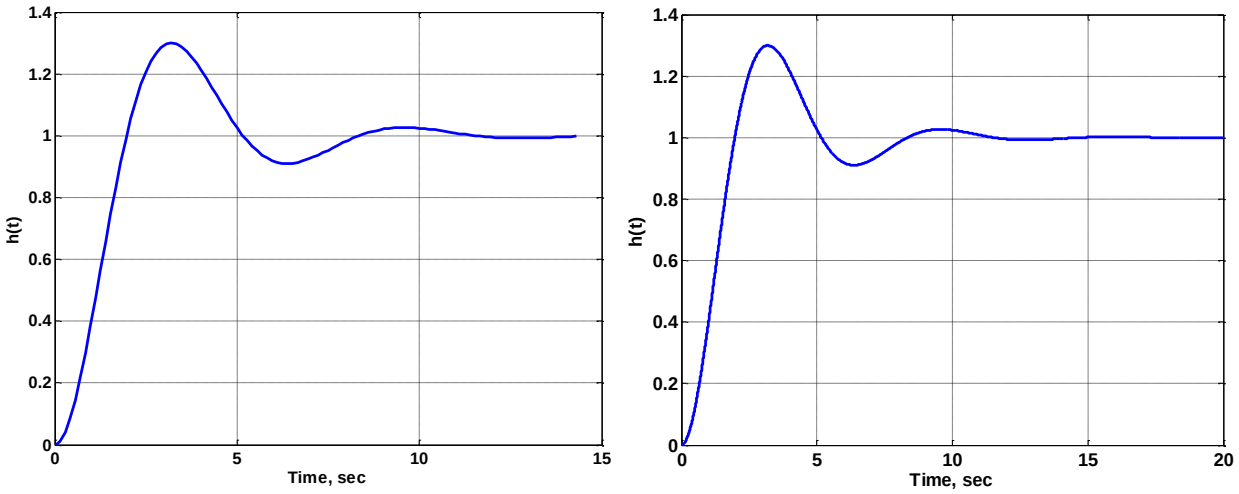


Рисунок 4.7 – Синтаксис функції `plot`

Змінна *t* визначається як вектор рядок, що складається з моментів часу, у які ми бажаємо одержати значення вихідної змінної *h*(*t*).

Нижче представлений скрипт, що використовує розглянутий формат функції `step`. Перехідна характеристика, побудована за допомогою цього скрипту, представлена на рисунку 4.8,*а*.



а) *б)*

Рисунок 4.8 – Перехідна функція: *а)* із $t_к$ заданою автоматично;
б*) із* $t_к$ заданою вручну

```
W=tf(1,[0.9 0.68 1]);
[h,t]=step(W);
plot(t,h),grid
xlabel('Time, sec')
ylabel('h(t)')
```

Зверніть увагу, що MATLAB автоматично пропонує час перехідного процесу. У цьому легко переконатися, якщо вивести значення вектора *t* на екран.

У програмі можна також задати значення $t = t_к$, так що перехідна функція буде обчислена на інтервалі від $t = 0$ до $t = t_к$, і крім того вказати число точок у цьому інтервалі. Наприклад, якщо в зазначеному вище скрипті після завдання передаточної функції записати:

```
t=[0:0.005:20];
[h,t]=step(W,t);
```

то отримаємо графік, представлений на рисунку 4.8, *б*.

4.6 Завдання для самостійної роботи

1. Розгляньте систему зі зворотнім зв'язком, що зображена на рисунку 4.4. Прийнявши

$$W_p(s) = \frac{1}{s+1} \text{ і } W_o(s) = \frac{s+2}{s+3}$$

виконайте наступне.

а) За допомогою MATLAB обчисліть передаточну функцію замкнутої системи.

б) Визначте реакцію системи на одиничний східчастий вплив і переконайтеся, що кінцеве значення вихідної змінної дорівнює 2/5.

2. На рисунку 4.9 надана структурна схема двигуна постійного струму, що керується по ланцюгу якоря.

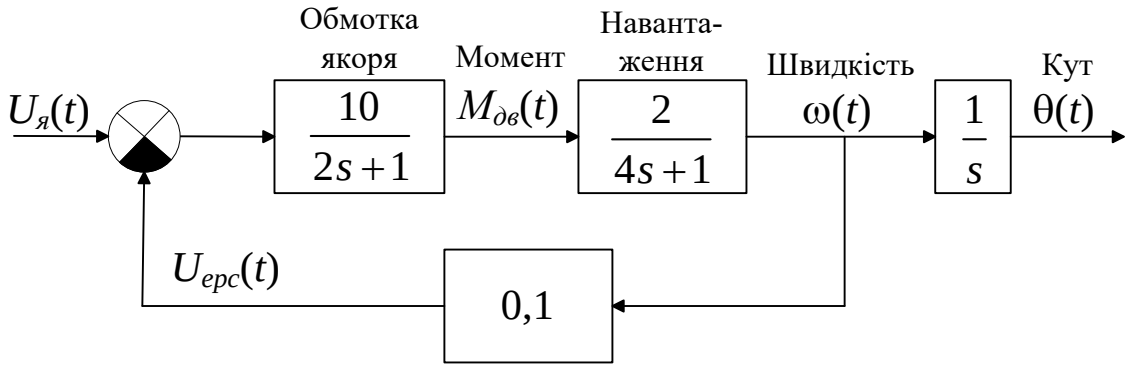


Рисунок 4.9 - Структурна схема двигуна постійного струму, що керується за ланцюгом якоря

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-формі.

3. На рисунку 4.10 зображена замкнена система керування.

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

б) Побудуйте перехідну функцію системи.

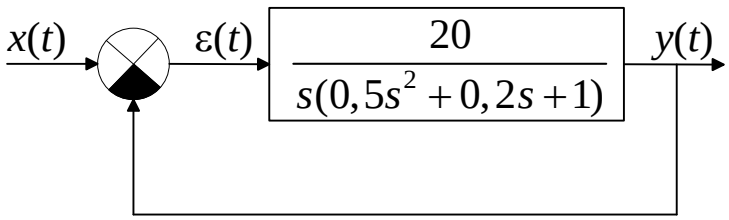


Рисунок 4.10 - Структурна схема замкненої системи керування

4. 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?

5. Дано диференціальне рівняння $\ddot{y} + 4\dot{y} + 4y = u$, де $y(0) = \dot{y}(0) = 0$ та $u(t)$ є одинична східчаста функція. Отримайте рішення цього рівняння аналітично та перевірте результат за допомогою MATLAB, одночасно побудувавши графік перехідної функції.

6. Розгляньте механічну систему, зображену на рисунку 4.11, де входом є $f(t)$, а виходом - $y(t)$.

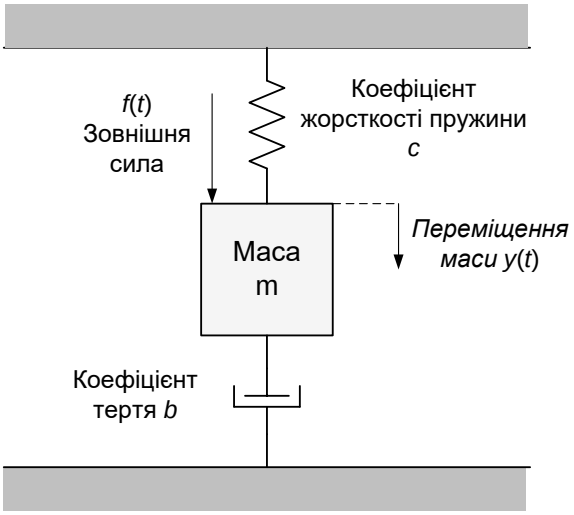


Рисунок 4.11 – Механічна система «маса-пружина» с демпфуванням

Визначте передаточну функцію, що зв'язує $f(t)$ та $y(t)$, а також за допомогою MATLAB отримайте графік реакції системи на одиничний східчастий вплив. Параметри системи: $m = 10$, $c = 1$ і $b = 0{,}5$. Переконайтеся, що максимальне значення вихідної змінної дорівнює 1,8.

7. Система керування положенням супутника по одній координаті може бути представлена структурною схемою, зображеною на рисунку 4.4. При цьому вхідним сигналом системи є задане положення супутника $\theta_з(t)$, вихідним сигналом – дійсне положення $\theta(t)$, а

$$W_p(s) = \frac{k(s+a)}{s+b} \text{ і } W_o(s) = \frac{1}{Js^2}.$$

Змінні k, a та b – параметри регулятора, а J – момент інерції супутника. Прийміть наступні значення: $J = 10{,}8 \cdot 10^8$, $k = 10{,}8 \cdot 10^8$, $a = 1$ і $b = 8$.

а) Напишіть програму MATLAB, що обчислює передаточну функцію замкнутої системи $W(s) = \theta(s)/ \theta_з(s)$.

б) Обчисліть та побудуйте графік реакції системи на східчасту зміну вхідного сигналу величиною 10°.

в) Точне значення моменту інерції супутника в загальному випадку невідомо й може повільно мінятися в часі. Зрівняйте реакції системи у випадках, коли J зменшується на 20% та на 50%. Параметри регулятора при цьому залишаються незмінними. Проаналізуйте отримані результати.

ЛАБОРАТОРНА РОБОТА № 5

АНАЛІЗ СТІЙКОСТІ СИСТЕМ КЕРУВАННЯ В MATLAB

Мета роботи: засвоєння методів аналізу стійкості лінійної безперервної системи за допомогою MATLAB.

5.1. Умова стійкості системи

Основне питання аналізу системи керування – це питання її стійкості. У більшості випадків нестійка система непрацездатна. Стійкою є така система, яка після короткочасного впливу, що обурює, повертається до початкового стану.

В якості такого обурюючого впливу зручно брати δ-функцію та оцінювати стійкість за імпульсною перехідною характеристикою. Тоді, аналітичне визначення стійкості можна записати в такий спосіб:

$$\lim_{t\to\infty} w(t) = 0. \qquad (5.1)$$

Нагадаємо, що для стійкості лінійної ланки необхідно та достатньо, щоб речовинні частини всіх корнів характеристичного рівняння A(s) = 0 були негативними, тобто всі полюси передаточної функції лежали ліворуч від мнимої осі комплексної площини. Якщо хоча б один речовинний корінь або пара комплексних сполучених коренів перебуває праворуч від мнимої осі, то система є нестійкою. Якщо є нульовий корінь або пара чисто мнимих коренів, то система вважається нейтральною (що перебуває на границі стійкості). Отже, мнима вісь комплексної площини є границею стійкості.

Існують правила або критерії, які дозволяють не вирішуючи характеристичного рівняння визначити, чи перебувають всі його корені в лівій напівплощині. Критерії бувають алгебраїчні (Рауса, Гурвиця), і частотні (Найквиста, Михайлова).

5.2 Аналіз полюсів передаточної функції

MATLAB має широкі можливості для аналізу стійкості систем автоматичного керування. Розглянемо як MATLAB дозволяє досліджувати стійкість систем, що описуються передаточними функціями. Оскільки передаточна функція є відношенням 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 - як у чисельнику, так і в знаменнику.

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

```
>> 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 включається* при вводі полінома *p(s)*.

Якщо **p** є вектор-рядок, що складається з коефіцієнтів $p(s)$ у порядку зменшення ступенів, то функція *roots*(**p**) визначає вектор-стовпець, що містить корені цього полінома. Нехай, наприклад, характеристичній поліном має вигляд $p(s) = s^3 + 3s^2 + 4$. Тоді, його корені можна обчислити так:

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

Як бачимо, запропонована система буде нестійкою, так як її характеристичний поліном має позитивні корені.

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

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

Перемножування поліномів здійснюється за допомогою функції *conv*. Припустимо, що ми хочемо отримати поліном *n*(s*)* у розгорнутій формі, 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 отримуємо поліном $n(s) = 3s^3 + 14s^2 + 9s + 4$.

Для обчислення значення полінома при заданому значенні змінної використається функція *polyval*.

```
>> v=polyval(n,-5)
v =
   -66
```

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

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

```
>> W=tf([1 0.2], [1 1 2 23])
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 (рисунок 5.2):

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

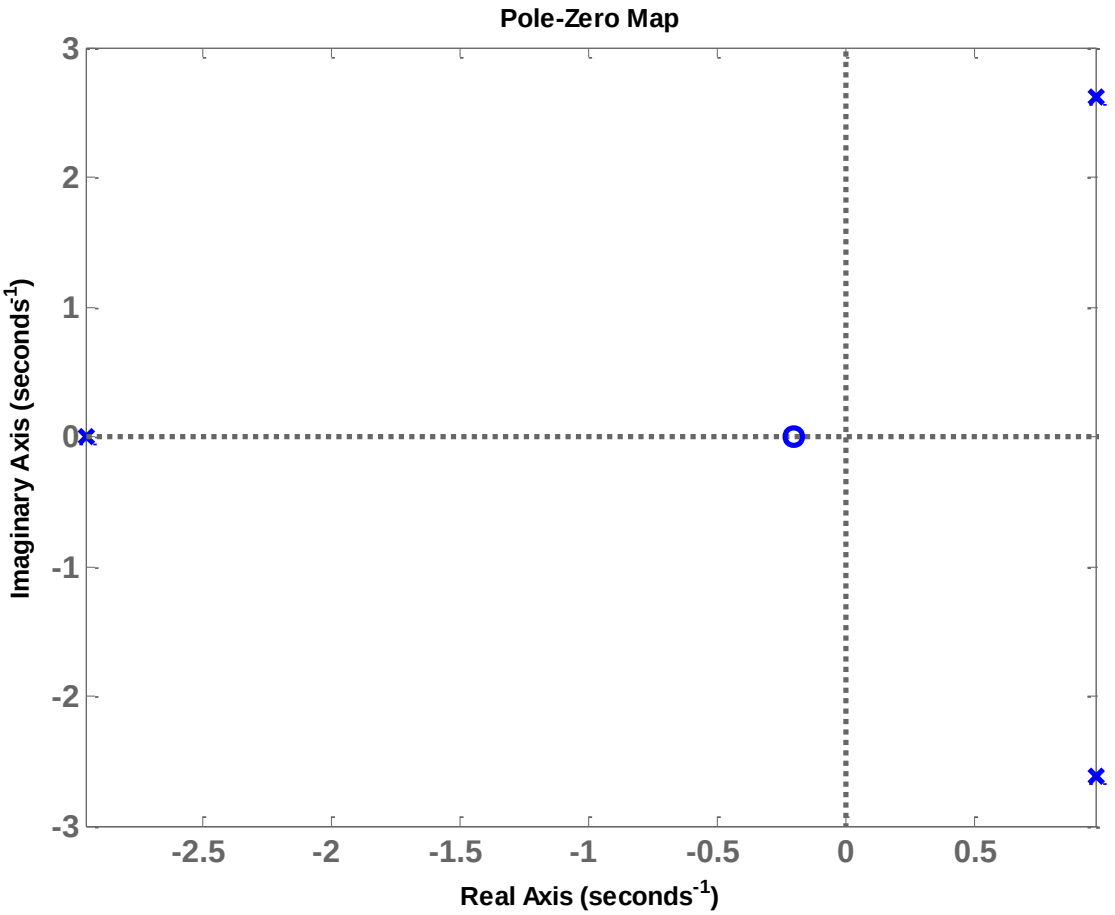


Рисунок 5.2 – Результат виконання команди `pzmap`

Корені характеристичного полінома можна знайти, використовуючи команду `damp(sys)`. Якщо ввести її у командному вікні, то після виконання отримаємо три стовпчики з цифрами. Перший (`Eigenvalue`) містить власні значення або корені характеристичного полінома, другий (`Damping`) коефіцієнти демпфірування, третій - (`Freq. (rad/s)`) власні частоти (рад/с). Наприклад, для веденої раніше передаточної функції:

```
>> damp(W)
      Eigenvalue          Damping       Frequency

 9.78e-001 + 2.61e+000i  -3.51e-001    2.79e+000
 9.78e-001 - 2.61e+000i  -3.51e-001    2.79e+000
-2.96e+000                1.00e+000    2.96e+000

(Frequencies expressed in rad/seconds)
```

5.3 Аналіз стійкості за критерієм Найквиста

Згідно з критерієм стійкості Найквиста, стійкість замкнутої системи можна визначити за амплітудно-фазовою частотною характеристикою (АФЧХ, або діаграма Найквиста) розімкнутої системи. АФЧХ (рисунок 5.3) будується за допомогою команди `nyquist`. Побудуємо, наприклад, діаграму Найквиста для наступної системи другого порядку:

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

Для цього потрібно ввести наступні команди:

```
>> W=tf([2 5 1],[1 2 3]);
>> nyquist(W)
```

АФЧХ не повинна охоплювати критичну точку з координатами (-1, *j*0), яка на діаграмі (на рисунок 5.3) позначена «+». Для зручності можна змінити масштаб осей. 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 та Y_i – відповідні мінімальні й максимальні значення осей.

Викликавши контекстне меню (натисканням правої кнопки миші), та вибравши підменю **Characteristics** можна визначити запаси стійкості системи за амплітудою та фазою. Підвівши курсор до характерних точок можна отримати чисельні значення для запасів стійкості (рисунок 5.3). Зверніть також увагу на те, що MATLAB сам дає відповідь, чи стійка замкнена система (Closed Loop Stable? Yes).

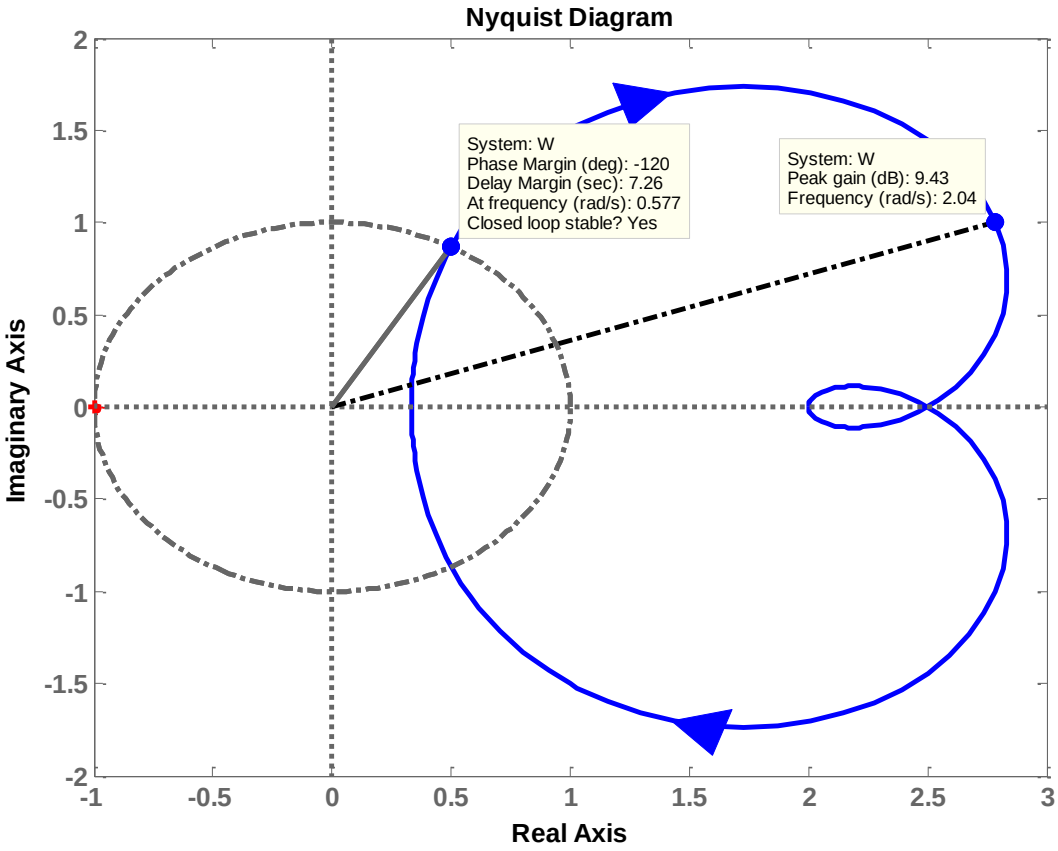


Рисунок 5.3 - Визначення стійкості за критерієм Найквиста

5.4 Аналіз стійкості за діаграмами Боде

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

Для побудови ЛАЧХ та ЛФЧХ використовують команду `margin`:

```
margin(W)
```

У результаті виконання цієї команди з'явиться вікно з логарифмічними частотними характеристиками (рисунок 5.4).

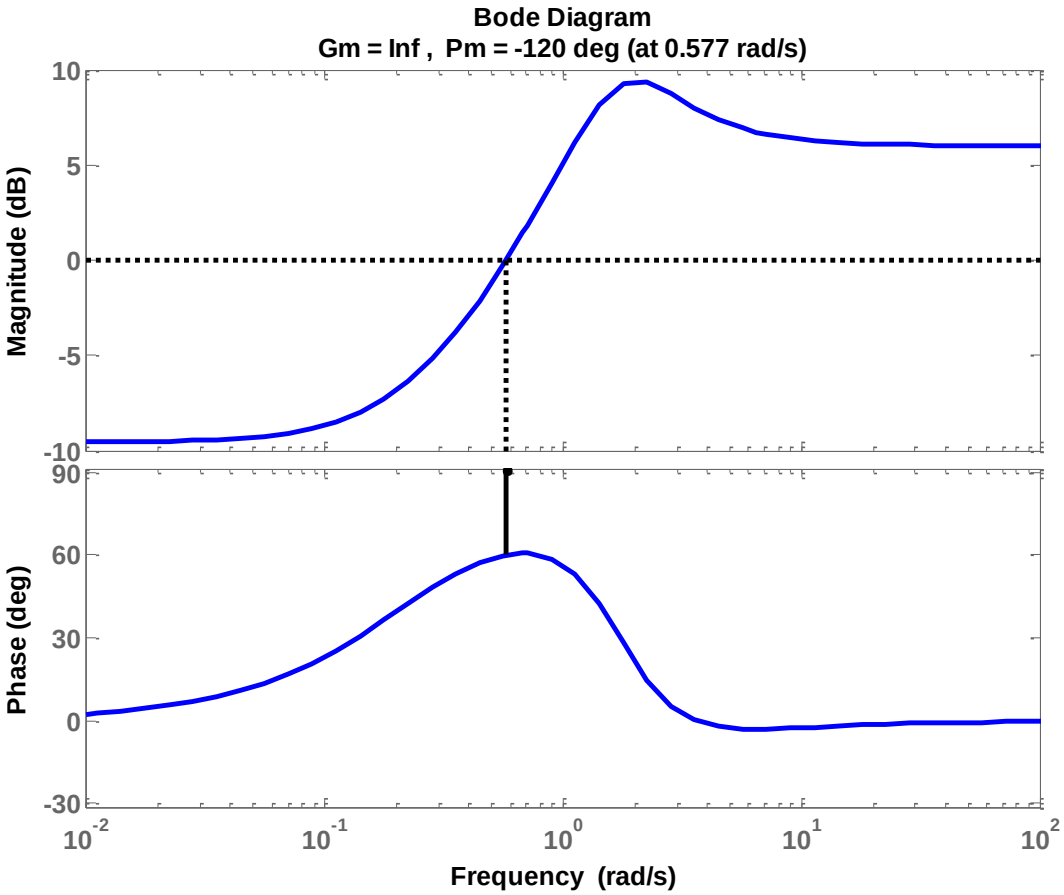


Рисунок 5.4 - Визначення стійкості за логарифмічними частотними характеристиками

За ЛАЧХ і ЛФЧХ можна визначити запаси стійкості за амплітудою та за фазою. Величина запасів за амплітудою (`Gm`) та за фазою (`Pm`) зазначена у верхній частині вікна.

5.5 Завдання до лабораторної 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. Розгляньте два поліноми:

$$p(s) = s^2 + 2s + 1, \quad q(s) = s + 1.$$

За допомогою MATLAB обчисліть наступне:

а) $p(s)\cdot q(s)$;

б) корінь $p(s)$ і$\cdot q(s)$;

в) корінь $p(s)$ і$\cdot q(s)$ при $s = 2$ і $s = -3$.

3. За допомогою критерію Гурвиця визначте число коренів у лівій напівплощині, у правій напівплощині та на мнимій осі для наступних характеристичних рівнянь:

а) $q(s) = s^3 + 3s^2 + 5s + 7 = 0$; *б)* $q(s) = s^4 + 3s^2 + 4s + 10 = 0$;

в) $q(s) = s^4 + 2s^2 + 1 = 0$. *г)* $q(s) = s^3 + 2s^2 + 1 = 0$.

Перевірте результати, обчисливши корені рівнянь за допомогою MATLAB.

4. Розгляньте систему з одиничним негативним зворотним зв'язком, що в розімкнутому стані має передаточну функцію

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

За допомогою MATLAB знайдіть корені характеристичного рівняння замкнутої системи при K = 1, 2 та 5. При яких значеннях K замкнена система стійка?

5. На рисунку 5.5 надана структурна схема системи керування частотою обертів двигуна внутрішнього згоряння. Постійна часу T_1 обумовлена обмеженнями на упорскування пального в карбюратор і пропускну здатність трубопроводу. Двигун має постійну часу $T_{дв}$ = = J/b = 3 с, де J – момент інерції, b – коефіцієнт тертя вала. Постійна часу датчика швидкості $T_{дс}$ = 0,4 с, а T_1 = 1 с.

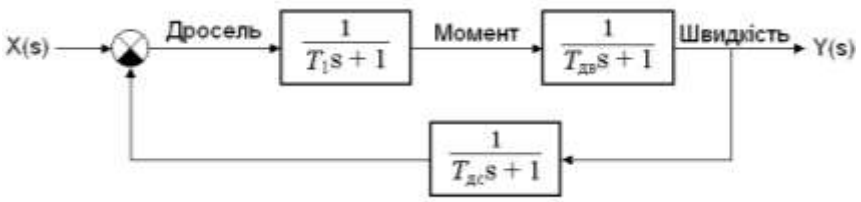


Рисунок 5.5 - Система керування частотою обертів двигуна внутрішнього згоряння

Визначте стійкість цієї системи:

а) за допомогою команди `pole`;

б) за допомогою команди `pzmap`;

в) за допомогою критерію Найквиста;

г) за допомогою діаграм Боде.

Зміст звіту.

1. Команди, що використовуються для аналізу стійкості системи керування та їх призначення.

2. Лістинг команд та скриптів для виконання завдань на лабораторну роботу.

3. Результати виконання завдань з відповідними ***висновками***.

ЛАБОРАТОРНА РОБОТА № 6

АНАЛІЗ ЯКОСТІ СИСТЕМ КЕРУВАННЯ В MATLAB

Мета роботи: засвоєння методів аналізу якості лінійної безперервної системи за допомогою MATLAB. Навчитися використовувати **LTIViewer** для аналізу систем керування.

6.1 Основні положення

Стійкість - головна, але не єдина вимога до системи керування. Стійку систему оцінюють додатково, користуючись спеціальними показниками якості. Найчастіше ці показники визначаються за перехідною функцією (рисунок 6.1). Основними показниками якості є наступні:

- час регулювання $t_р$ – визначає швидкодію системи (час, за який вихідна величина входить в «коридор» $\pm\,\delta$ = 2..5% від $h_{уст}$);

- максимальне перерегулювання

$$\sigma = \frac{\Delta h}{h_{уст}} \cdot 100\%. \qquad (6.1)$$

Максимальне перерегулювання характеризує запас стійкості системи, тобто ступінь видалення системи від коливальної границі стійкості;

- t_{max} – час настання першого максимуму;

- число перерегулювань N - кількість коливань вихідної величини системи протягом часу перехідного процесу;

- стала похибка $\varepsilon_{уст} = 1 - h_{уст}$.

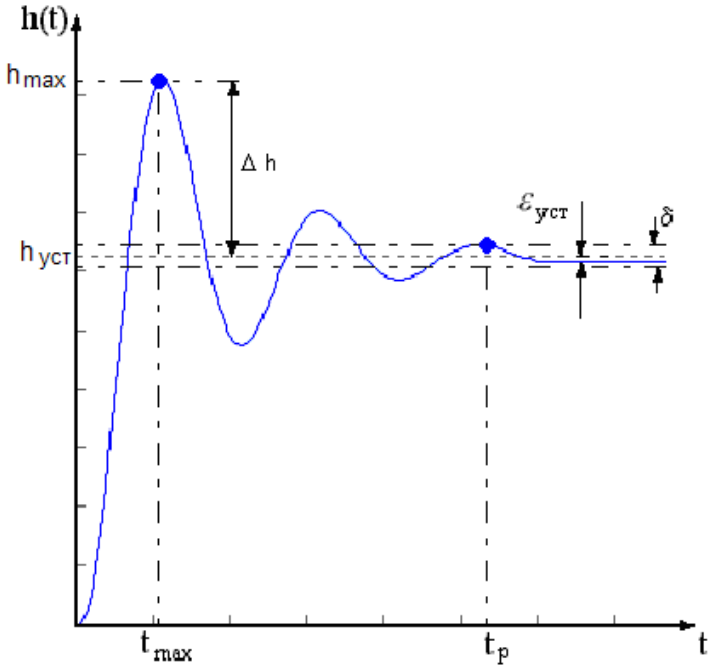


Рисунок 6.1 - Визначення якості за перехідною функцією

Системи, що мають ненульову сталу похибку при одиничному вхідному сигналі називаються **статичними системами**. У цьому випадку сталу похибку називають *статичною похибкою* $\varepsilon_{ст}(t)$.

Якщо система автоматичного керування має нульову сталу похибку $\varepsilon_{уст} = 0$ (якщо вхідним сигналом є одинична функція), то вона називається **астатичною.** Астатичні властивості системі надає інтегруюча ланка.

Оскільки астатичні системи мають $\varepsilon_{уст} = 0$, то їх оцінюють при більш складних для відпрацьовування вхідних сигналах, наприклад, лінійно наростаючому.

Астатизмом порядку ν називається властивість системи з постійною сталою похибкою відпрацьовувати вхідний сигнал виду

$$x(t) = C_0 + C_1 t + C_2 t^2 + \ldots + C_\nu t^\nu \ldots \qquad (6.2)$$

Таким чином, система з астатизмом порядку ν відпрацьовує без похибки усе більш прості сигнали ($t^{\nu-1}$, $t^{\nu-2}$…), а із зростаючою

похибкою більш складні ($t^{\nu+1}$, $t^{\nu+2}$…) сигнали.

Одна з найважливіших характеристик лінійної системи - коефіцієнт підсилення в сталому режимі або статичний коефіцієнт підсилення (static gain, DC-gain). Його можна визначити як стале значення сигналу виходу при постійному вхідному сигналі, рівному одиниці. Розмірність цієї величини дорівнює відношенню розмірностей сигналів виходу і виходу. Розглянемо диференціальне рівняння

$$\ddot{y} + 2\dot{y} + 3y = 4\dot{u} + 5u.$$

Вважаючи всі похідні (у сталому режимі) рівними нулю, одержуємо

$$3y = 5u \quad \Rightarrow \quad y = \frac{5}{3}u.$$

Таким чином, Статичний коефіцієнт підсилення дорівнює $k_s = 5/3$.

Якщо задана передаточна функції, для обчислення k_s треба підставити в неї $s = 0$, оскільки змінна s відповідає оператору диференціювання. Розглянутому вище рівнянню відповідає передаточна функція

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

Звідси

$$k_s = \lim_{s \to 0} W(s) = \frac{5}{3}.$$

Якщо система містить інтегруючу ланку (передаточна функція має полюс в точці $s = 0$), ця межа дорівнює нескінченності, тобто, при постійному сигналі вихід нескінченно збільшується або зменшується, не досягаючи сталого режиму.

Розглянемо тепер, як такий потужний засіб як MATLAB дозволяє оцінити якість роботи систем автоматичного керування.

6.2 Аналіз якості за перехідними характеристиками

Як уже згадувалося, якість системи 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 за реакцією на типовий вхідний сигнал.

Розглянемо систему другого порядку, зображену на рисунку 6.2.

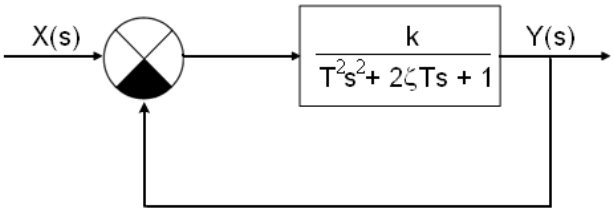


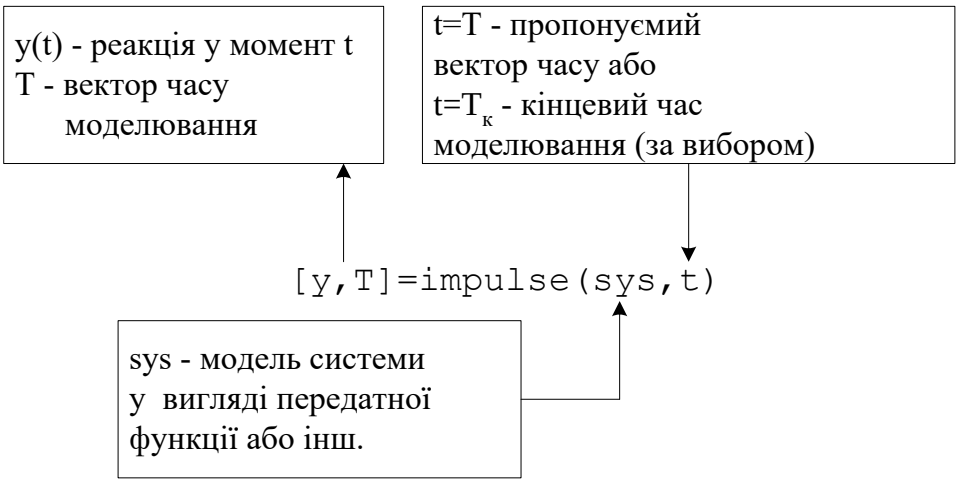
Рисунок 6.2 - Одноконтурна система другого порядку

Побудуємо реакцію системи на найпоширеніші типові сигнали - одиничний східчастий сигнал і одиничний імпульс та проаналізуємо якість роботи системи на рисунку 6.2. Відповідний скрипт наведений нижче, а характеристики, побудовані з його допомогою представлені на рисунках 6.3 та 6.4. Змінюючи значення змінної zeta можна визначити її вплив на поводження системи.

```
zeta=0.4; % Коефіцієнт демпфування
num=[1]; % Чисельник передаточної функції
den=[1 2*zeta 1]; % Знаменник передаточної функції
W=tf(num,den); % Передаточна функція
step(W), grid % Побудова перехідної характеристики
ylabel('h(t)') % Назва осі y
title('step response' ) % Назва графіка
legend('\zeta = 0.4') % Виведення легенди
figure(2)  % Відкриваємо друге графічне вікно
impulse(W), grid %Побудова імпульсної перехідної
              % характеристики
ylabel('\itw(t)')
title('impulse response')
legend('\zeta = 0.4')
```

Тут ми використали нову для нас команду MATLAB `impulse`, що дозволяє будувати імпульсну перехідну функцію системи. Фор-

мат функції `impulse` наступний:



Повернемось тепер до графіків на рисунках 6.3 та 6.4. Викликавши контекстне меню та поставивши в підменю **Characteristics** прапорці напроти відповідних показників якості, можна отримати наглядне подання про якість системи. Якщо підвести курсор миші до якої-небудь характерної точки, то на екрані відображається числове значення відповідного показника якості (рисунок 6.3). Отримане вікно можна зафіксувати натисканням миші та потім переміщувати в довільному напрямку або редагувати за допомогою контекстного меню.

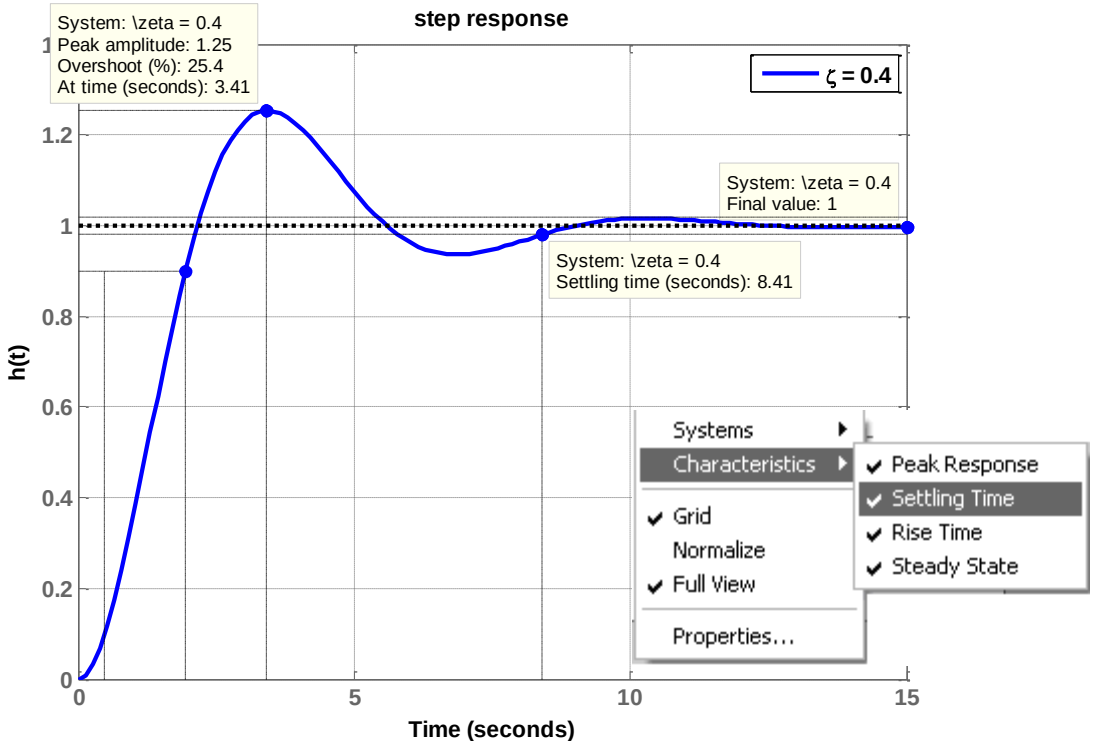


Рисунок 6.3 - Визначення показників якості за перехідною функцією

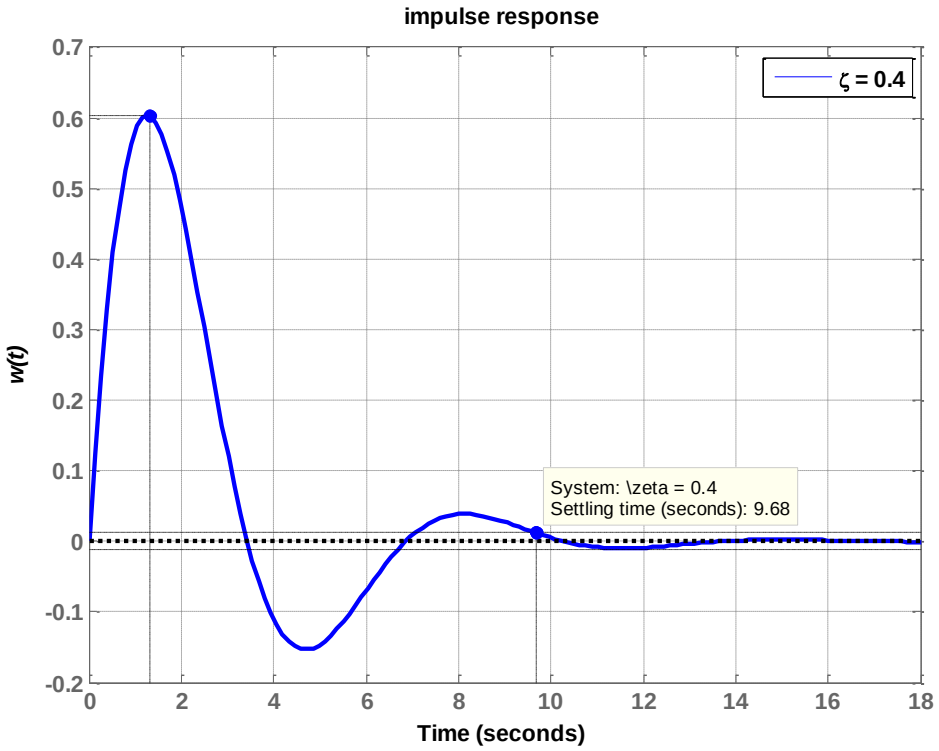


Рисунок 6.4 - Визначення показників якості за імпульсною перехідною функцією

Щоб знайти статичний коефіцієнт підсилення моделі `W` в MATLAB, використовується команда

```
>> k = dcgain (W)
k =
     1
```

6.3 Реакція системи на сигнал довільної форми

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 даному випадку формат функції `lsim` наведений на рисунку 6.5:

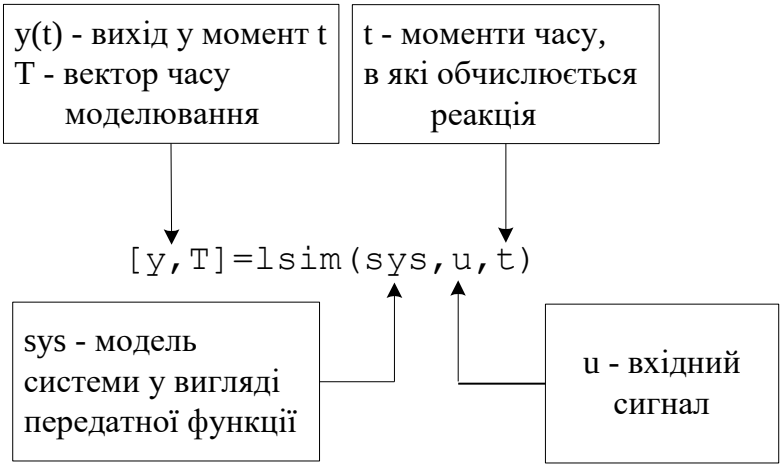


Рисунок 6.5 – Формат функції `lsim`

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

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

Розглянемо як приклад систему керування кермовим механізмом мобільного робота (рисунок 6.6).

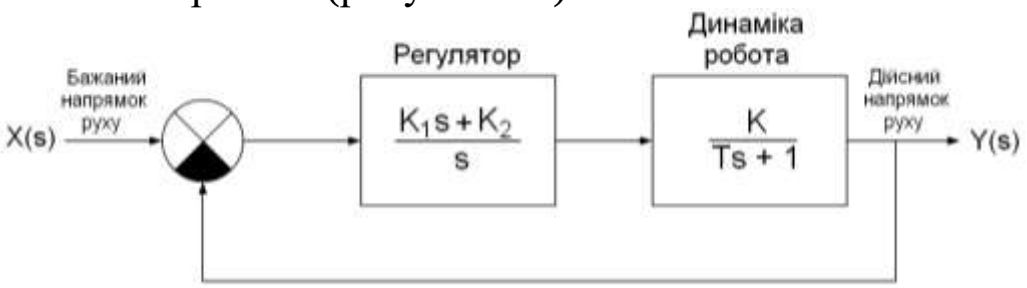


Рисунок 6.6 - Структурна схема системи керування кермовим механізмом мобільного робота

Якщо $K_2 = 0$, то стала похибка системи при одиничному східчастому вхідному сигналі дорівнює

$$\varepsilon_{уст} = \frac{1}{1 + KK_1}. \qquad (6.3)$$

Якщо $K_2 \neq 0$, то ми маємо систему з астатизмом першого порядку, тобто стала похибка при одиничному вхідному сигналі дорівнює нулю.

При лінійному вхідному сигналі

$$\varepsilon_{уст} = \frac{1}{K_2 K}. \qquad (6.4)$$

На рисунку 6.6 зображена реакція системи на періодичний сигнал трикутної форми, отриманий за допомогою функції `lsim`:

```
%Обчислення реакції системи керування
%мобільним роботом на сигнал трикутної форми
W₀=tf([10 20],[1 10 0]);
W=feedback(W₀,1); %Передаточна ф-я замкненої системи
t=[0:0.1:8.2]'; %Час моделювання
v1=[0:0.1:2]';v2=[2:-0.1:-2]'; % Вхідний сигнал
v3=[-2:0.1:0]'; u=[v1;v2;v3]; % трикутної форми
[y,T]=lsim(W,u,t);
plot(T,y,t,u,'--'),
xlabel('Time (sec)'),
ylabel('\theta (radians)'), grid
```

Як видно з рисунка 6.7, на перехідній характеристиці чітко простежується поява сталої похибки, що може не мати істотного значення, якщо K_2K досить велике. Помітимо, що хоча в сталому режимі похибка відмінна від нуля, але вихідний сигнал змінюється із заданою швидкістю.

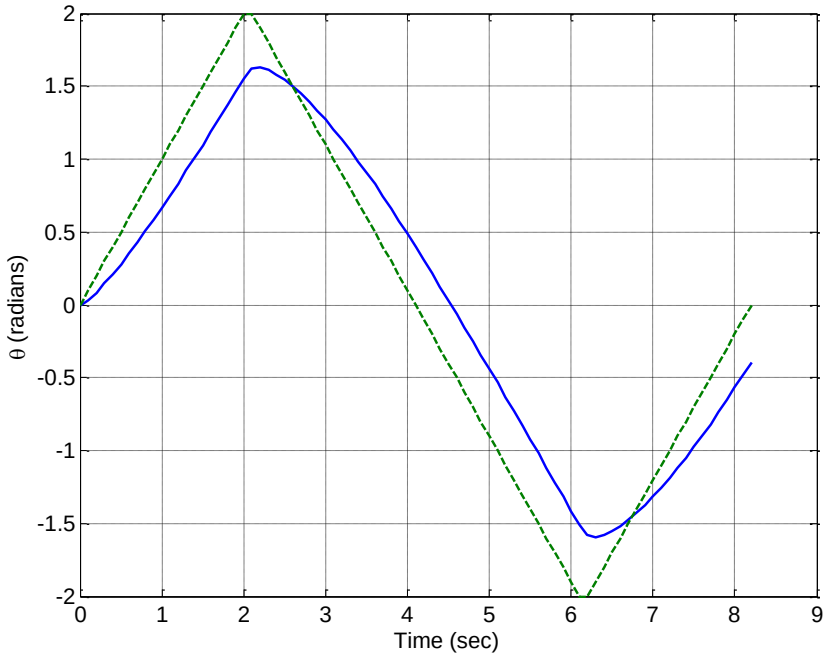


Рисунок 6.7 - Реакція системи керування кермовим механізмом мобільного робота на сигнал трикутної форми

Очевидно, що для зменшення сталої похибки бажано збільшувати K_2K. Однак збільшення K_2K приводить до одночасного зменшення коефіцієнта загасання ζ і, отже, до більш коливального характеру реакції системи на східчастий вхідний вплив. Тому необхідно знайти розумний компроміс між значеннями параметрів K_2K і ζ.

6.4 Графічний інтерфейс LTI Viewer

Практично всі розглянуті вище характеристики можна побудувати, використовуючи графічний 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 (рисунок 6.8) натискаємо **File→Import**..., вибираємо потрібну нам передаточну функцію, наприклад `W_ob` і натискаємо **OK**.

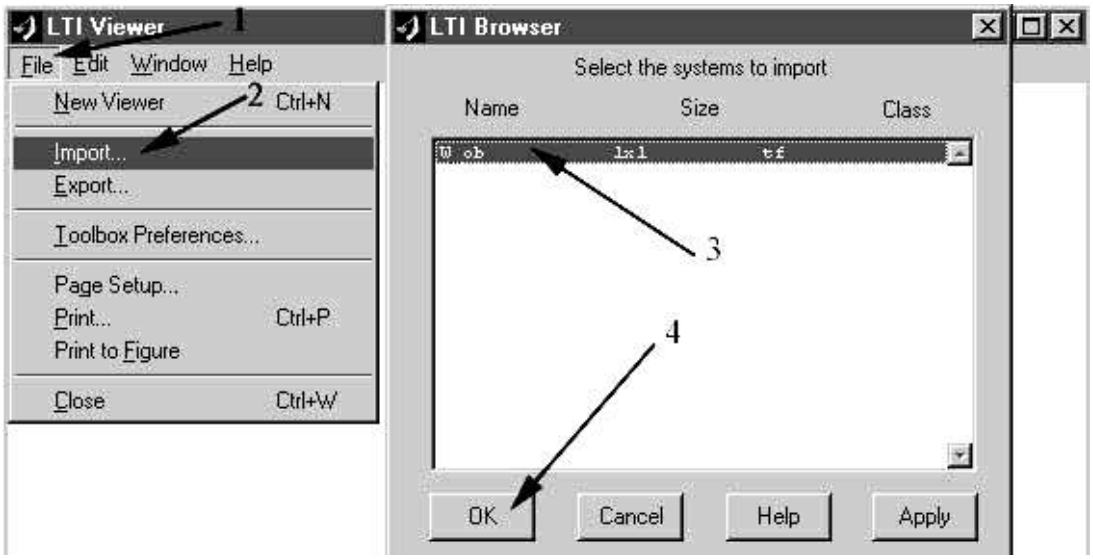


Рисунок 6.8 - Вибір передаточної функції в LTI Viewer

Після цього автоматично будується реакція `W_ob` на одиничний східчастий вплив. Натисканням правої кнопки миші в області графіка викликається меню (рисунок 6.9).

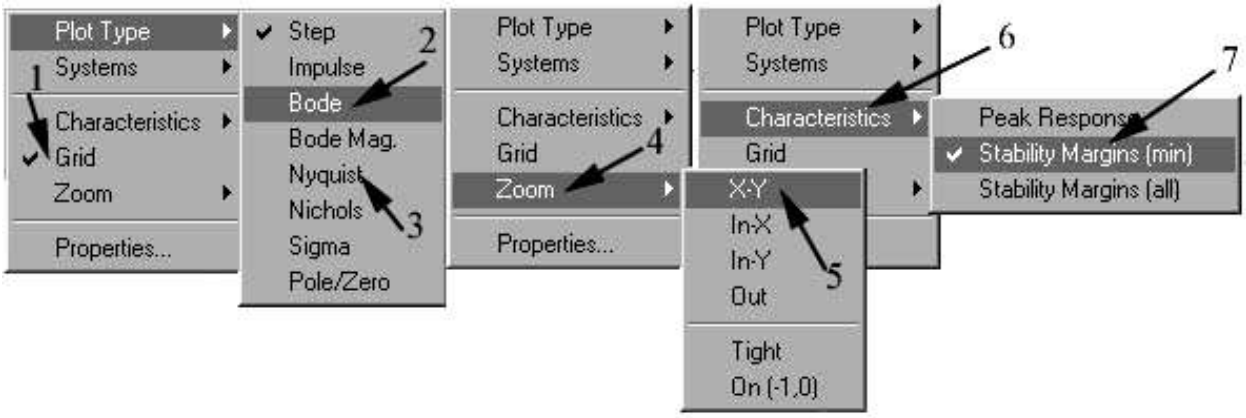


Рисунок 6.9 - Меню LTI Viewer

На отриманий графік можна накласти сітку - галочка навпроти **Grid** (1). Щоб побудувати логарифмічні частотні характеристики **Bode** (2) або АФЧХ **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) та обводимо мишею при натиснутій лівій кнопці бажану область (наприклад для АФХ область у районі (-1, *j*0)). Для ЛФЧХ й АФЧХ можна визначити запаси за амплітудою й фазою. Для цього вибираємо **Characteristics** (6) і ставимо галочку напроти **Stability Margins** (7). Якщо є запаси за амплітудою та фазою, то з'являються точки, наводячи на них курсор миші можна переглянути їхні чисельні значення.

У принципі, інтерфейс LTI Viewer'а простий та очевидний і робота з ним не повинна викликати істотних труднощів.

6.5 Завдання до лабораторної роботи

1. Система в розімкнутому стані має передаточну функцію

$$W(s) = \frac{s+7}{s^2(s+10)}.$$

Вважаючи, що система охоплена одиничним негативним зворотним зв'язком:

а) за допомогою команди `dcgain` знайдить статичний коефіцієнт підсилення;

б) отримайте реакцію системи на сигнал $X(s) = 1/s^2$. Розгляньте інтервал часу $0 \le t \le 25$ с. Чому дорівнює стала похибка?

2. Передаточна функція прямого ланцюга в системі з одиничним негативним зворотним зв'язком має вигляд:

$$W(s) = \frac{50}{s(s+10)}.$$

За допомогою команди `step` побудуйте графік перехідної функції замкнутої системи та визначте значення максимального перерегулювання σ, часу першого максимуму t_{max} і часу регулювання $t_р$. Доповніть графік відповідними чисельними значеннями σ, t_{max} та $t_р$.

3. На рисунку 6.10 зображена спрощена структурна схема системи керування швидкістю автомобіля. Положення дроселя регулюється соленоїдом, двигун представлений аперіодичною ланкою 1-го порядку з постійною часу 1 с, автомобіль і навантаження також описані передатною функцією 1-го порядку з постійною часу 3 с. У якості обурення прийнятий ухил дороги. Швидкість повинна регулюватися в діапазоні від 70 до 120 км/ч. У системі використаний ПІ-регулятор із передавальною функцією $W_p(s) = K_p + K_i/s$.

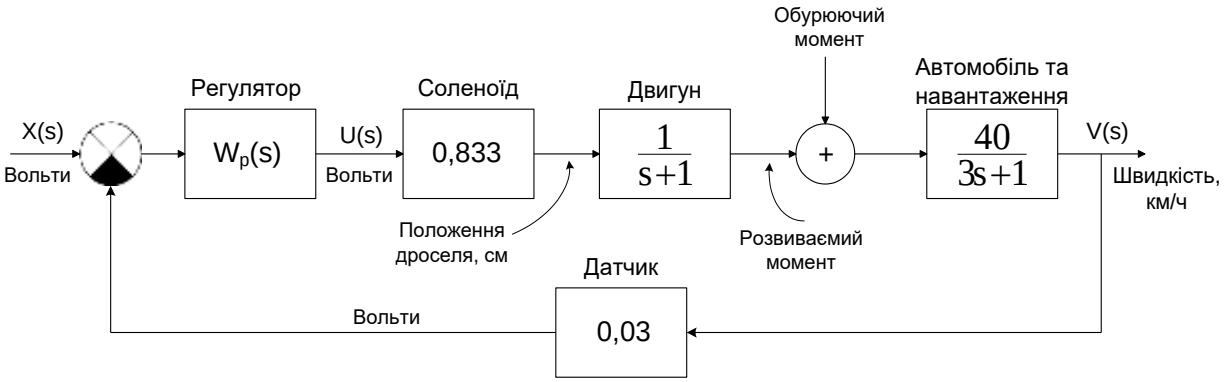


Рисунок 6.10 - Система керування швидкістю автомобіля

а) Визначте діапазони значень K_p і K_i, при яких система стійка.

б) Припустіть, що значення K_i встановлено в діапазоні, відповідному стійкості, знайденому в п. (*а*), а K_p обрано дещо менше ніж нижня межа діапазону для цього параметра. Отже, система є нестійкою. Якби ви при цьому керували автомобілем і намагалися впливати на швидкість, то як би він реагував на ваші дії? Постарайтеся відповісти якомога точніше, без розрахунків.

в) За допомогою **LTI Viewer** промоделюйте систему за умов п. (*б*) і перевірте ваші очікування.

Зміст звіту.

1. Команди, що використовуються для аналізу побудови перехідної, імпульсної перехідної характеристик та реакції на довільний сигнал.

2. Лістинг команд та скриптів для виконання завдань на лабораторну роботу.

3. Результати виконання завдань з відповідними ***висновками***.

СПИСОК ЛІТЕРАТУРИ

1. Теория автоматического управления: Учебно-методическое пособие /[Гурко А.Г., Еременко И.Ф., Кортнева В.С. и др]. - Харьков, ХНАДУ, 2009. – 216 с.

2. Гурко О.Г. Аналіз та синтез систем автоматичного управління у MATLAB: Навчальний посібник /О.Г. Гурко, І.Ф. Єрьоменко. - Харків, ХНАДУ, 2012. – 284 с.

3. Дорф Р. Современные системы управления / Р. Дорф, Р. Бишоп. Пер. с англ. Б.И. Копылова. - М.: Лаборатория Базовых Знаний, 2002. – 832 с.

4. Филлипс Ч. Системы управления с обратной связью / Ч. Филлипс, Р. Харбор. Пер. с англ. Б.И. Копылова. - М.: Лаборатория Базовых Знаний, 2001. – 616 с.

5. Востриков А.С. Теория автоматического регулирования/А.С. Востриков, Г.А. Французова. – М.: Высш. шк., 2004. – 365 с.

6. Дэбни Дж. Simulink 4. Секреты мастерства / Дж. Дэбни. – М.: Лаборатория знаний, 2003. – 403 с.

7. Образовательный математический сайт Exponenta.ru – Режим доступа: http://www.exponenta.ru

ЗМІСТ

Вступ..... 3
Лабораторна робота №1. Основи MATLAB..... 4
Лабораторна робота №2. Вектори та матриці в MATLAB..... 13
Лабораторна робота №3. Побудова графіків в MATLAB……..... 28
Лабораторна робота №4. Представлення моделей систем керування у пакеті MATLAB …… 42
Лабораторна робота №5. Аналіз стійкості систем керування в MATLAB..... 56
Лабораторна робота №6. Аналіз якості систем керування в MATLAB…… 64
Список літератури..... 76

Навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робот
з дисципліни
«Теорія автоматичного керування »
для студентів зі спеціальностей
6.050202 та 6.050702

Укладачі: ГУРКО Олександр Геннадійович
КОНОНИХІН Олександр Сергійович

Відповідальний за випуск Л.І.Нефьодов

Редактор _____

Підписано до друку_____ Формат 60х84 1/16.
Умовн. друк арк. _____ Обл. – вид. Арк._
Замовлення №_____ Тираж ____ прим. Ціна договірна

ХНАДУ, 61002, Харків, вул. Петровського, 25

Свідоцтво державного комітету інформаційної політики, телебачення та радіомовлення України про внесення суб'єкта видавничої справи до державного реєстру видавців, виготівників і розповсюджувачів видавничої продукції, серія № ДК № 407

Підготовлено і надруковано видавництвом
Харківського національного автомобільно-дорожнього університету