МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

„КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

СПЕЦІАЛЬНІ РОЗДІЛИ ТЕОРІЇ АВТОМАТИЧНОГО КЕРУВАННЯ

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт для студентів спеціальності
„Автоматизоване управління технологічними процесами”

Київ

НТУУ “КПІ”

2012

Спеціальні розділи теорії автоматичного керування: Метод. вказівки до викон. лабораторних робіт для студ. спец. „Автоматизоване управління технологічними процесами” / Уклад.: Кубрак А.І., Ковалюк Д.О. – К. : НТУУ ”КПІ“, 2012. – 43с.

Гриф надано Вченою радою ІХФ
(Протокол № 6 від 31 травня 2012 р.)

Навчальне видання

СПЕЦІАЛЬНІ РОЗДІЛИ ТЕОРІЇ АВТОМАТИЧНОГО КЕРУВАННЯ

Методичні вказівки до виконання лабораторних робіт для студентів спеціальності „Автоматизоване управління технологічними процесами”

Укладачі: Кубрак Анатолій Іванович, канд. техн. наук, проф
Ковалюк Дмитро Олександрович, канд. техн. наук, старший викладач

Відповідальний редактор А.І. Жученко, докт. техн. наук, проф.

Рецензент О.Л. Сокольський, канд. техн. наук, доц.

Авторська редакція

ЗМІСТ

ВСТУП

Моделювання систем керування є важливою задачею, що виникає як на етапі розробки систем так і безпосередньо в процесі їх експлуатації. Результатом даної задачі є отримання характеристик системи керування чи розрахунок параметрів регулятора для забезпечення заданих показників технологічного процесу.

Враховуючи високу складність програмної реалізації моделювання систем керування «з нуля», в методичних вказівках пропонується використовувати математичний пакет Matlab (інструментарій Control System Toolbox) для розв’язання цієї задачі.

В методичних вказівках охоплено всі етапи розрахунку систем керування – отримання математичної моделі об’єкту, створення замкнених та розімкнених систем, оптимізація параметрів регулятора, дослідження характеристик системи керування.

Лабораторна робота №1

ДОСЛІДЖЕННЯ МОЖЛИВОСТЕЙ ПРОГРАМУВАННЯ АЛГОРИТМІВ РОЗВ'ЯЗАННЯ ЗАДАЧ ЧИСЛОВОГО АНАЛІЗУ В СЕРЕДОВИЩІ MATLAB

Мета роботи – дослідити можливості програмування в Matlab на прикладі задачі побудови перехідної характеристики замкненої системи

Теоретичні відомості

Matlab надає широкі можливості для розв'язання інженерних та дослідницьких задач. Це реалізується за рахунок використання вбудованих функцій. Проте вбудовані функції є лише засобом для досягнення мети, а самому дослідникові необхідно використати ці засоби в своєму алгоритмі дослідження. Організація взаємодії між компонентами Matlab та реалізація самих алгоритмів дослідження може бути здійснена лише програмним шляхом. З огляду на це розглянемо базові моменти програмування в Matlab.

Будь-яка програма є не що інше ніж виконання певних операцій над даними для отримання результату. Основними складовими програм в Matlab (як і в інших мовах програмування) є:

1. Типи даних та операції над ними
2. Оператори організації обчислювальних процесів
3. Можливість оформлення частини коду у вигляді підпрограм для повторного використання.

Розглянемо вказані складові детальніше.

Типи даних. На рис. 1.1. наведена класифікація типів даних, визначених у MATLAB. Зазначимо, що в даних методичних вказівках всі приклади реалізовані в Matlab R2010a.

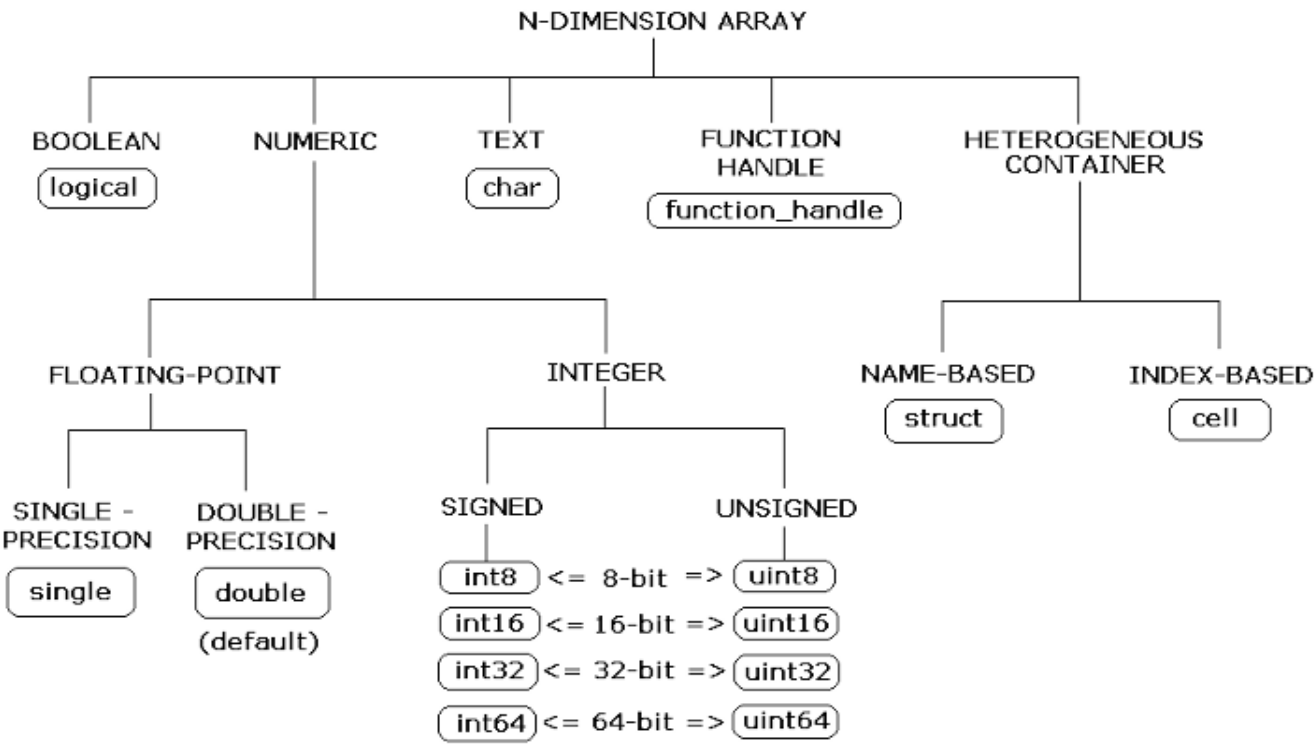


Рис.1.1. Класифікація типів даних в Matlab

В Matlab існує 15 основних типів даних (класів), кожен з яких представляється у вигляді матриці або масиву мінімального розміру 0-by-0 і за необхідності розширюється до n-мірного масиву довільного розміру.

При написанні програм в Matlab не вимагається явне оголошення типів даних, зазначення типу використовується переважно для підвищення ефективності (швидкодії) виконання програм. З огляду на це, стисло розглянемо основні типи даних. Числові типи в Matlab включають цілий тип integer із знаком або без (signed and unsigned) та типи даних з плаваючою крапкою (single, double). Крім того Matlab забезпечує підтримку логічного (boolean) та символьного (text) типів. Основні елементарні функції для операції над типами даних наведені в наступній таблиці

Таблиця 1.1. Елементарні функції

Дія	Назва вбудованої функції
тригонометричні, аргумент в радіанах	cos, cot, csc, sec, sin, tan
тригонометричні, аргумент в градусах	cosd, cotd, cscd, seed, sind, tand
зворотні тригонометричні, результат в радіанах	acos, acot, acsc, asec, asin, atan, atan2
зворотні тригонометричні, результат в градусах	acosd, acotd, acscd, asecd, asind, atand
гіперболічні	cosh, coth, csch, sech, sinh, tanh
зворотні гіперболічні	acosh, acoth, acsch, asech, asinh, atanh
степені, логарифми, корені,	exp, expml, log, loglp, iog2, loglo, nextpow2, pow2, reallog, realsqrt, sqrt
округлення	ceil, fix, floor, round
найбільший спільний дільник	gcd
найменше спільне кратне	lem
модуль числа	abs
знак числа	sign
остача від ділення з урахуванням знака діленого	mod
остача від ділення	rem
розкладання числа на прості множники	factor
обчислення факторіалу	factorial
дробово-раціональна апроксимація дійсного числа	rats
генерація простих чисел, що не перевищують аргументу	primes

Оператори організації обчислювальних процесів є синтаксичними конструкціями, які дозволяють керувати ходом виконання програми, і наведені в наступній таблиці.

Таблиця 1.2. Оператори організації обчислювальних процесів

№	Оператор	Опис
1	variable = expression	Оператор присвоєння. Обчислює значення виразу expression і заносить результати обчислень у змінну variable
2	if умова 1 оператори 1 [elseif умова 2 оператори 2 elseif умова 3	Умовний оператор. Виконує лише оператори, розміщені в секції для якої справджується задана умова. Якщо задовольняється умова 1, то виконується група оператори 1, якщо задовольняється

	оператори 3 else оператори] end	умова 2, то виконуються оператори 2 і т.д.. Якщо всі зазначені умови не задовольняються, то виконуються оператори, розташовані між else та end
3	switch expression case value1 оператори 1 case value2 оператори 2 [otherwise оператори] end	Перемикач по значенню виразу. Якщо значення змінної expression збігається з значенням value, то виконується група операторів для цього value. Якщо значення expression не збігається з жодним value, то виконуються оператори, розташовані між otherwis та end
4	for variable = counter_begin: [step:] counter_end оператори end	Цикл з лічильником. Виконує блок операторів поки початкове значення змінної variable - counter_begin var не досягне кінцевого значення counter_end. Крок, з яким змінюється значення змінної – 1 за замовчуванням, або [step:], за явного опису.
5	while умова оператори end	Цикл з передумовою. Виконує блок операторів, до тих пір, поки справджується умова
6	try оператори 1 catch оператори 2 end	Блок обробки виняткових ситуацій. Якщо при виконанні операторів 1 виникає помилка, то управління передається групі операторів 2. В іншому випадку група оператори 2 не виконується
7	break	Переривання циклів або вихід з керуючих конструкцій

Оформлення частини коду у вигляді підпрограм – є одним з фундаментальних принципів програмування, який дозволяє багаторазово застосовувати певні дії (описані один раз) до різних наборів вхідних даних. В Matlab підпрограми реалізуються за допомогою функцій.

Функції в Матлаб оголошуються наступним чином:

```
function [out1, out2, ...] = myfun(in1, in2, ...)

тіло функції
```

Дана інструкція оголошує функцію `myfun`, її вхідні (`in1, in2, ...`) та вихідні `[out1, out2, ...]` параметри.

Оголошення та код функції розміщуються в окремому текстовому файлі, що має ім’я аналогічне назві функції (`myfun` для даного випадку) та розширення - `.m`.

Функції самі по собі не використовуються, а мають бути викликані з «головної» програми. Виклик функції здійснюється за таким же синтаксисом як і її опис, але на вхід подаються вже реальні значення параметрів. Змінні всередині функції мають локальну область видимості, тобто не впливають на значення змінних головної програми з такими ж іменами.

Порядок виконання роботи

1. Написати функцію для розрахунку вихідного сигналу регулятора, що реалізує лінійні закони регулювання.
2. Задати об’єкт керування у вигляді системи однорідних диференційних рівнянь згідно варіанту.
3. Написати програму для отримання перехідної характеристики замкненої системи керування.
4. Дослідити перехідні характерники залежно від настроювання параметрів регулятора та закону регулювання.

Зміст звіту

1. Назва, мета, порядок виконання роботи.
2. Математична модель об’єкта у вигляді системи диференційних рівнянь.
3. Блок-схема головної програми та функції для розрахунку виходу регулятора.
4. Лістинг програми з результатами виконання.
5. Висновки за результатами досліджень.

Контрольні запитання

1. Лінійні закони керування.
2. Метод Ейлера для розв’язання диференційних рівнянь.
3. Структурні схеми замкненої і розімкненої системи керування.
4. Критерії якості системи керування.
5. Реалізація функцій в Matlab.
6. . Оператори організації обчислювальних процесів в Matlab.

Приклад виконання роботи

Функція для розрахунку вихідного сигналу регулятора, що реалізує лінійні закони регулювання

```
function [Y_current] = eil_su(A_coef, B_coef, M_X0, v0, Y_prev)

    % eil_su - функція для розвязання системи др. першого порядку
методом Ейлера
    % A_coef - коефіцієнти виходів,
    % B_coef - коефіцієнти входів,
    % M_X0 - масив значень вхідних сигналів об’єкта станом на
початок кроку,
    % Y_current - масив параметрів стану об’єкта після кроку
інтегрування
    % Y_prev - масив параметрів стану об’єкта на початок кроку
інтегрування
    % v0 - крок інтегрування

    inputs_count = size(B_coef, 2);     %кількість входів обєкту
    outputs_count = size(A_coef, 2);    %кількість виходів обєкту

    for z = 1 : outputs_count
        sum = 0;
        for i = 1 : outputs_count
            sum = sum + A_coef(z,i) * Y_prev(i);
        end;
        for i = 1 : inputs_count
            sum = sum + B_coef(z,i) * M_X0(i);
        end;
        Y_current(z) = Y_prev(z) + v0*sum;
    end;
```

Програма для отримання перехідної характеристики замкненої системи керування

```
A(1,1) = -0.2;
A(1,2) = 0;
A(1,3) = -0.02;
A(2,1) = 0.1;
A(2,2) = -0.1;
A(2,3) = 0;
A(3,1) = 0;
A(3,2) = 1;
A(3,3) = 0;

B(1,1) = 0.02;
B(2,1) = 0;
B(3,1) = 0;

Nzr = 4;
Kreg = 0.1;
Ti = 10000;
Tv = 0;

D = 200;
L = 100;
ks = 10;
Ninp = 1;
Nout = 3;

Dt = D / L;
V0 = Dt / ks;

for s = 1:size(B, 2)
    Mx(s) = 0;
end
for s = 1:size(A, 2)
    My(s) = 0;
end
Hts(1) = 0; Eps = 0; Integral = 0; Y = 0; X = 0;

Zavdannya = 1;

for i = 1 : L
    for s = 1 : ks
        [Eps, Integral, X] = step_reg(Nzr, Kreg, Ti, Tv, Y,
Zavdannya, V0, Eps, Integral, X);
        Mx(Ninp) = X;
        My = eil_su(A, B, Mx, V0, My);
        Y = My(Nout);
    end;
    Hts(i) = Y;
    dt(i) = Dt * i;
end;
plot(dt, Hts);
```

Лабораторна робота 2

ДОСЛІДЖЕННЯ СПОСОБІВ ПОБУДОВИ ГРАФІКІВ, ВІЗУАЛІЗАЦІЇ ЗОБРАЖЕНЬ ПОВЕРХОНЬ В СЕРЕДОВИЩІ MATLAB

Мета роботи – ознайомитися із засобами побудови графічних зображень в Matlab.

Теоретичні відомості

Для побудови графічних зображень в Matlab присутні наступні функції:

PLOT - графік в лінійному масштабі ¶

LOGLOG - графік у логарифмічному масштабі

SEMILOGX, SEMILOGY - графік у напівлогарифмічному масштабі

POLAR - графік в полярних координатах ¶

PLOT3 - побудова ліній і точок в тривимірному просторі ¶

MESHGRID - формування двовимірних масивів X і Y¶

MESH, MESHC, MESHZ - тривимірна сітчаста поверхня¶

SURF, SURFC - затінена сітчаста поверхня¶

SURFL - затінена поверхню з підсвічуванням¶

CONTOURC - формування масиву опису ліній рівня¶

CONTOUR - зображення ліній рівня для тривимірної поверхні¶

CONTOUR3 - зображення тривимірних ліній рівня

Розглянемо, найбільш поширені:
1. `Plot` – двовимірний графік.

Синтаксис:

plot (y)

plot (x, y)

plot (x, y, s)

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

Опис:

Команда *plot (y)* будує графік елементів одновимірного масиву *y* залежно від номера елемента; якщо елементи масиву *y* комплексні, то будується графік *plot(real(y), imag(y)).* Якщо *Y* - двовимірний дійсний масив, то будуються графіки для стовпців; у разі комплексних елементів їх уявні частини ігноруються.

Команда *plot(x,y)* відповідає побудові звичайної функції, коли одновимірний масив *x* відповідає значенням аргументу, а одновимірний масив *y* - значенням функції.

Команда *plot(x,y,s)* дозволяє виділити графік функції, вказавши спосіб відображення лінії, спосіб відображення точок, колір ліній і точок за допомогою строкової змінної s.

2. `contour` – побудова ізоліній

синтаксис:

contour(Z) contour(x, y, Z)

contour(Z, n) contour(x, y, Z, n)

contour(Z, v) contour(x, y, Z, v)

[C, h] = contour (...)

Опис:

Команда *contour (Z)* малює двовимірні лінії рівня для масиву даних *Z*, що визначають поверхню в тривимірному просторі без урахування діапазону зміни координат x і y.

Команда *contour (x, y, Z)*, де x і y - вектори, малює лінії рівня для масиву даних Z з урахуванням діапазону зміни координат x і y.

Команди *contour (Z, n), contour (x, y, Z, n)* малює n ліній рівня для масиву даних Z; за замовчуванням, n дорівнює 10.

Команди *contour (Z, v), contour (x, y, Z, v)* малюють лінії рівня для заданих значень, які вказані у векторі v.

Функція [C, h] = contour (...) повертає масив C і вектор-стовпець дескрипторів h графічних об'єктів line для кожної лінії рівня.

3. `SURF` - Затінена сітчаста поверхня

синтаксис:
surf (X, Y, Z, C)
surf (x, y, Z, C)
surf (Z, C)
surf (X, Y, Z)
surf (x, y, Z)
surf (Z)

Опис:
Команда *surf (X, Y, Z, C)* виводить на екран сітчасту поверхню для значень масиву Z, визначених на множині значень масивів X і Y. Колір визначається масивом C.

Порядок виконання роботи

1. Для функції однієї змінної згідно номеру варіанту побудувати графіки у декартовій та полярній системах координат.
2. Для функції двох змінних згідно номеру варіанту побудувати каркасну поверхню та графік ізоліній.
3. Виконати побудову декількох графіків в одній графічній області.
4. Виконати форматування графіків: підпис значень, осей, легенда.

Зміст звіту

1. Назва, мета, порядок виконання роботи.
2. Аналітичний вираз досліджуваної функції та діапазон вхідних змінних.
3. Лістинг програми з результатами виконання.
4. Висновки за результатами досліджень.

Контрольні запитання

1. Вбудовані функції Matlab для візуалізації зображень.
2. Побудова декількох графіків в одній графічній області.
3. Функції для побудови поверхонь.
4. Параметри функцій для побудови поверхонь та ізоліній.

Приклад виконання роботи

Побудувати ізолінії для функції, що обчислює значення модуля полінома при комплексному аргументів.

```
function [Re, Im] = horner_complex(A_coef, Re_arg, Im_arg)
    % horner_complex - функція обчислення значення поліному
    % від комплексного аргументу за схемою Хорнера
    % A_coef - коефіцієнти поліному,
    % Re_arg - дійсна частина агрументу,
    % Im_arg - уявна частина аргументу,
    % Re - дійсна частина значення поліному,
    % Im - уявна частина значення поліному

    n = size(A_coef, 2);
    Re = A_coef(n);
    Im = 0;

    for z = n-1 : -1 : 1
        R1 = Re * Re_arg - Im * Im_arg + A_coef(z);
        Im = Re * Im_arg + Im * Re_arg;
        Re = R1;
    end;
```

```
Xmin = -2;
Xmax = 2;
Ymin = -2;
Ymax = 2;
L = 100;
H = 100;
Hiz = 1000;

A = [1 0 0 0 0 1];
dx = (Xmax-Xmin)/L;
dy = (Ymax-Ymin)/H;

for z = 1 : L+1
    x = Xmin + (z-1) * dx;
    for s = 1 : H+1
        y = Ymin + (s-1) * dy;
        [Re_val, Im_val] = horner_complex(A, x, y);
        polinom_modul(z, s) = sqrt(Re_val^2 + Im_val^2);
        Y1(z, s) = y; X1(z, s) = x;
    end
end
contour(X1, Y1, polinom_modul);

zmin = floor(min(polinom_modul(:)));
zmax = ceil(max(polinom_modul(:)));
zinc = (zmax - zmin) / Hiz;
zlevs = zmin:zinc:zmax;
contour(X1, Y1, polinom_modul, zlevs);

contour(X1, Y1, polinom_modul, 1000);

[C,h] = contour(X1, Y1, polinom_modul);
set(h,'LevelStepMode', 'manual')
set(h,'LevelStep', 0.2)
```

Дана програма обчислює значення функції в певному інтервалі і будує ізолінії чотирма різними способами:

1. contour(X1, Y1, polinom_modul) - будується 10 ліній рівня (за замовчуванням);
2. contour(X1, Y1, polinom_modul, zlevs) – на вхід подається масив значень функції, для яких треба побудувати ізолінії;
3. contour(X1, Y1, polinom_modul, 1000) – вказується кількість ліній рівня, що повинні відображатися;
4. set(h,'LevelStep', 0.2) – встановлюється крок для значень функції, для яких будуть побудовані ізолінії

Результати виконання:

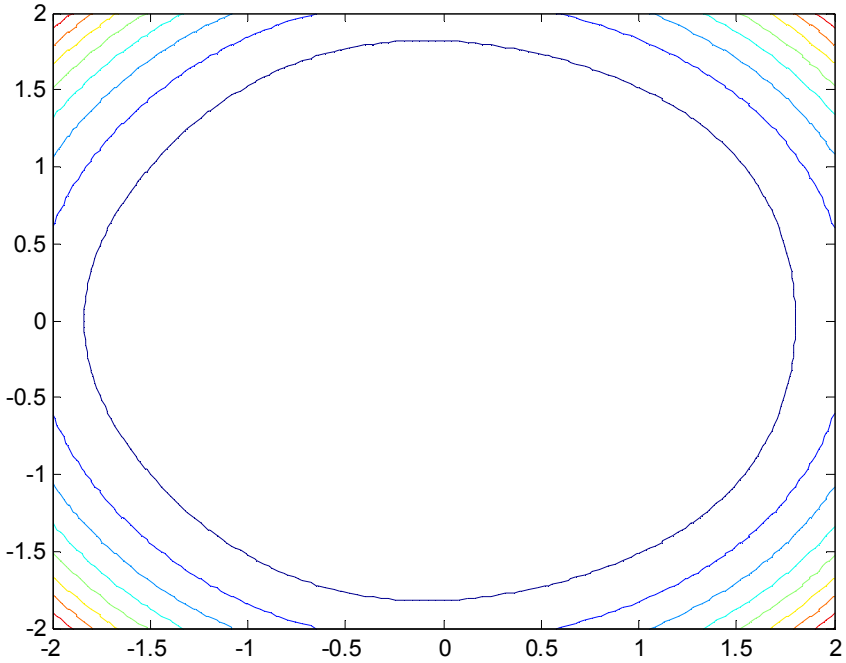


Рис 2.1. Ізолінії для `contour(X1, Y1, polinom_modul)`

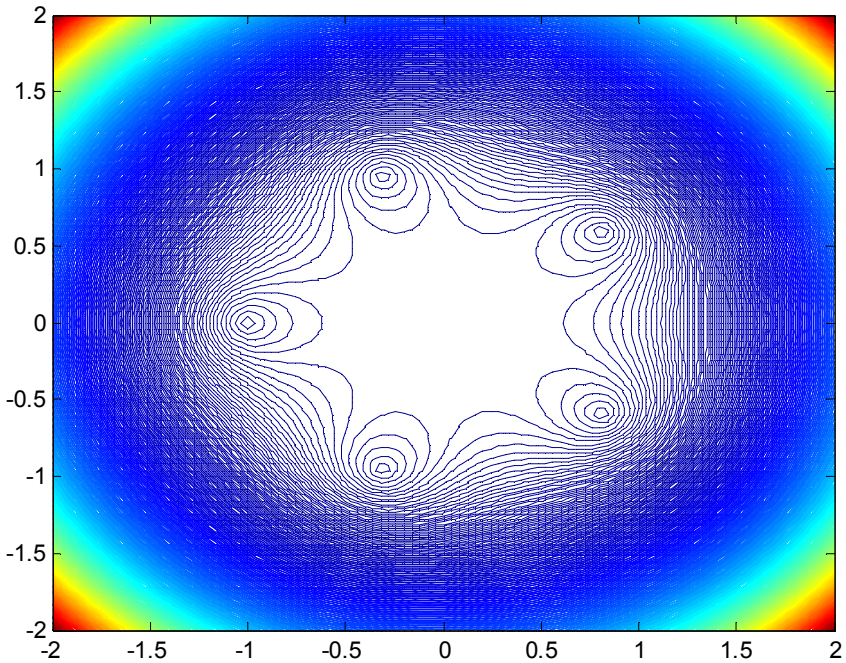


Рис 2.2. Ізолінії для `contour(X1, Y1, polinom_modul, zlevs)` та `contour(X1, Y1, polinom_modul, 1000)`

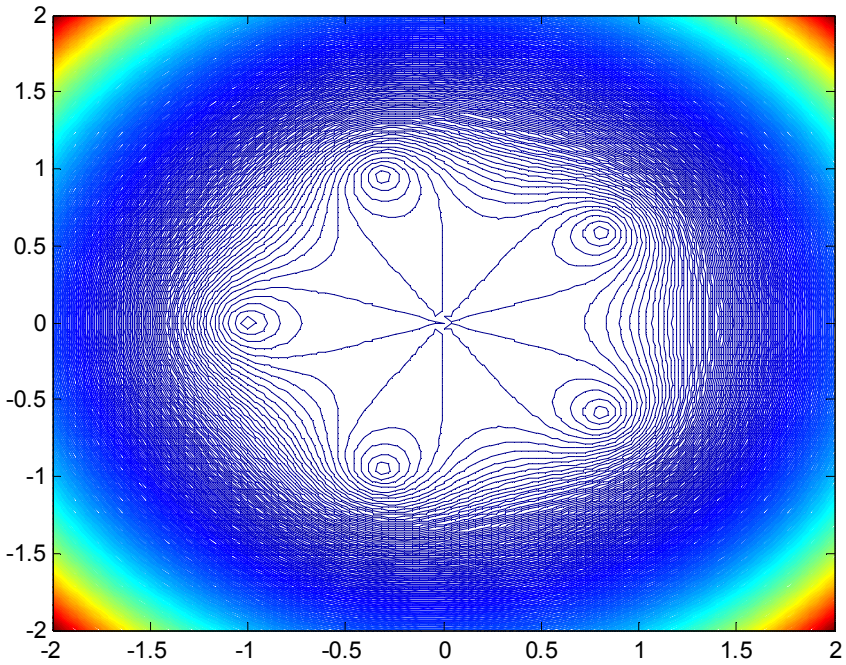


Рис 2.3. Ізолінії з параметром кроку `set(h,'LevelStep', 0.2)`

Лабораторна робота 3
ПЕРЕХІДНІ ХАРАКТЕРИСТИКИ ТА АФХ ДЛЯ ЛІНІЙНИХ СИСТЕМ

Мета роботи – дослідити основні можливості інструментарію Control System Toolbox на прикладі систем з лінійним законом регулювання

Теоретичні відомості

Оскільки Matlab використовується для широкого кола задач (обробка сигналів, фінансове моделювання, біологія, системи керування), тому його структура базується на певних інструментаріях (toolboxes) - колекціях спеціально-призначених функцій MATLAB. Дані пакети розширяють середовище MATLAB для розв'язання окремих класів задач у відповідній області.

Для дослідження систем автоматичного керування призначений пакет прикладних програм – Control System Toolbox. Можливості даного пакету є надзвичайно вагомими, але призначені для аналізу і синтезу *лінійних систем з постійними параметрами*.

Control System Toolbox дозволяє представити неперервну чи дискретну систему в наступному вигляді:

- передатна функція (Transfer functions)
- модель у просторі станів (State-space models)
- у вигляді реакції на сигнал відповідної частоти (Frequency response data (FRD) models)

Досліджувана система може мати один вхід – один вихід, тобто відноситись до класу SISO-систем – single-input/single-output. Система може також мати декілька пар вхід – вихід, тобто відноситись до класу MIMO-систем – multiple-input/multiple-output.

В даній роботі можливості Control System Toolbox будуть досліджені для одноконтурних замкнених систем автоматичного керування, у яких об'єкт задається передатною функцією:

$$W(p)=\frac{b_m p^m+\ldots+b_1 p+b_0}{a_n p^n+\ldots+a_1 p+a_0}e^{-p\tau}. \qquad (3.1)$$

1. *Створення моделі.*

Розглянемо як створити неперервну модель об'єкту керування, заданого передатною функцією

$$W(p)=\frac{b_m p^m+\ldots+b_1 p^1+b_0}{a_n p^n+\ldots+a_1 p^1+a_0} \qquad (3.2)$$

Для створення SISO моделі передатної функції використовується наступна команда:

```
W = tf(num,den)
```

де `num` і `den` – впорядкованими за спаданням степені коефіцієнтів поліномів $B(p)$ та $A(p)$ відповідно; `W` – створена модель.

Нехай маємо наступні коефіцієнти передатної функції: $b_0=1$, $a_1=10$, $a_0=1$. Тоді передатну функцію задамо наступними командами:

```
num = [1];                  %numerator
den = [10 1];               %denominator
Wob = tf(num, den);
```

Розглянемо тепер реалізацію транспортного запізнювання $e^{-p\tau}$. Це найпростіший тип запізнювання - на вході каналу керування. Для його реалізації використовуються властивість InputDelay. Для створеної моделі Wob встановлення транспортного запізнювання буде виконано наступними командами:

```
tau = 4;                    %input_delay
Wob.ioDelay = tau;
```

Об'єднуємо тепер все разом для створення передатної функції (3.1):

```
num = [1];              %numerator
den = [10 1];           %denominator
tau = 4;                %input_delay

Wob = tf(num, den);
Wob.ioDelay = tau;
Wob.variable = 'p'      %transfer function variable
```

Остання інструкція за допомогою властивості variable (специфічної для об'єкту tf) встановлює значення змінної Лапласа – p (більш вживане позначення в пострадянських країнах). Це суто технічний момент, який впливає лише на відображення результату. За замовчуванням змінна Лапласа позначається в Matlab як s. Якщо ввести вказані інструкції в командному вікні Matlab, то отримаємо наступний результат:

```
Transfer function:
                1
exp(-4*p)  * --------
             10 p + 1
```

2. Дослідження динамічних характеристик об'єкту керування

При заданій моделі об'єкта дослідника цікавлять його динамічні властивості, які як правило отримують у формі

- Перехідної характеристики – $h(t)$
- Імпульсної характеристики – $g(t)$

- Амплітудо-фазової характеристики

Розглянемо команди, які дозволяють отримати вищезазначені характеристики: `step, impulse, nyquistplot`. Дані команди мають багато варіацій вхідних і вихідних параметрів, проте тут будуть наведені лише ті, які потрібні для нашого дослідження. Решту варіантів використання команд можна отримати з офіційної документації.

step	реакція LTI-моделі на одиничний ступінчастий сигнал

Синтаксис	`step(sys)`
	`y = step(sys,t)`
	`[y,t] = step(sys)`

Опис	У випадку застосування без вихідних аргументів, ця функція виводить реакцію системи на екран. `y = step(sys,t)` - явним чином встановлює тривалість спостереження. `[y,t] = step(sys)` - повертає вихідну реакцію – `y`, вектор часу – `t`, який використовувався для імітації.

impulse	реакція LTI-моделі на одиничний імпульс
Синтаксис	`impulse(sys)` `y=impulse(sys,t)`
Опис	Опис аналогічний до функції step.

Побудуємо графіки перехідної та імпульсної характеристики для нашого об’єкта

```
time_begin = 0;
time_end = 50;
Time = time_begin : 0.1 : time_end;

subplot(2, 1, 1), step(Wob, Time)
subplot(2, 1, 2), impulse(Wob, Time)
```

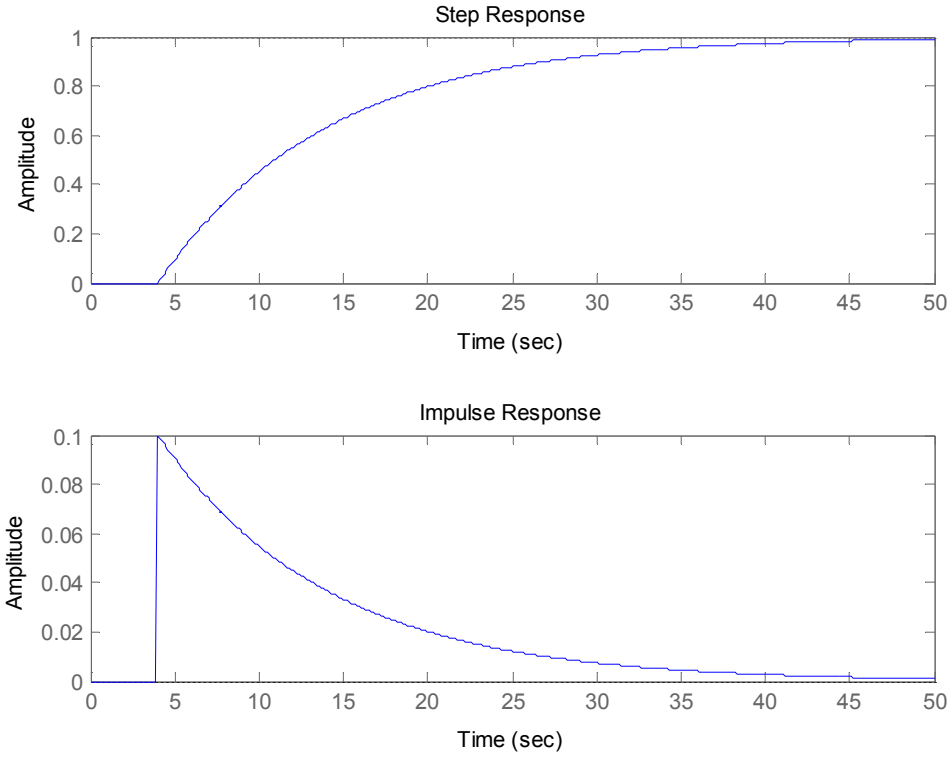


Рис. 3.1. Перехідна та імпульсна характеристика об’єкту

В даному лістингу використана команда subplot, яка дозволяє побудувати декілька графічних областей в одному вікні.

nyquistplot	побудова реакції на частотний сигнал
Синтаксис	`h = nyquistplot(sys)` `nyquistplot(sys,{wmin,wmax})`
Опис	`h = nyquistplot(sys)` - виводить АФХ для певної LTI-моделі `sys`, повертає вказівник на графік `nyquistplot(sys,{wmin,wmax})` - виводить АФХ для певної LTI-моделі `sys` в інтервалі частот `{wmin,wmax}`.

Побудуємо годограф АФХ для нашого об’єкта

```
Wbeg = 0.01;    %frequence begin
Wend = 2;       %frequence end
h = nyquistplot(Wob, {Wbeg, Wend});
setoptions(h, 'ShowFullContour', 'off');
```

Остання строка коду пояснюється тим, що за замовченням годограф розраховується і для від’ємних частот.

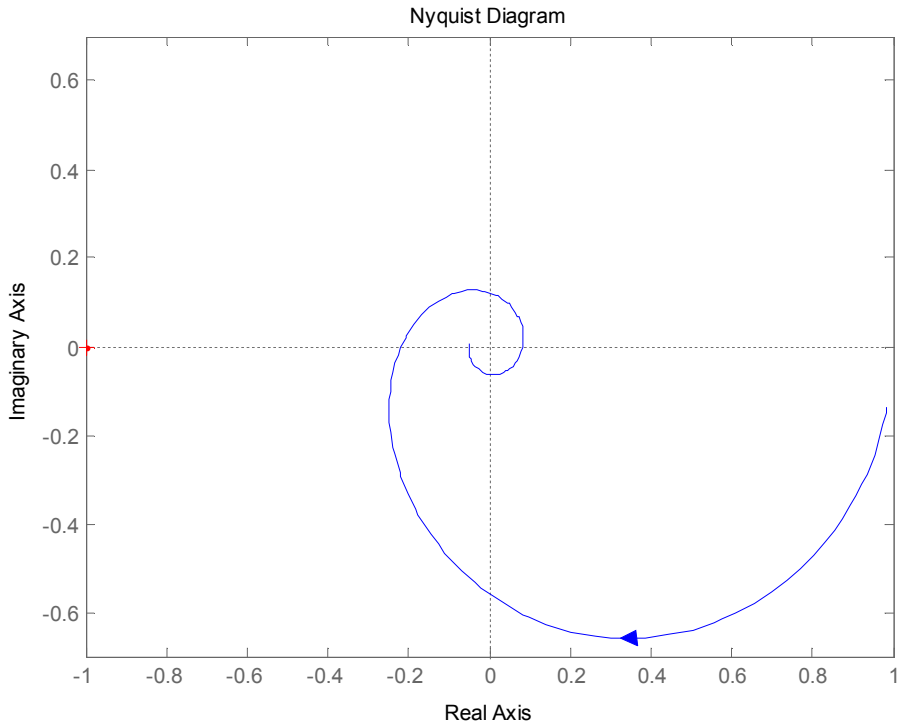
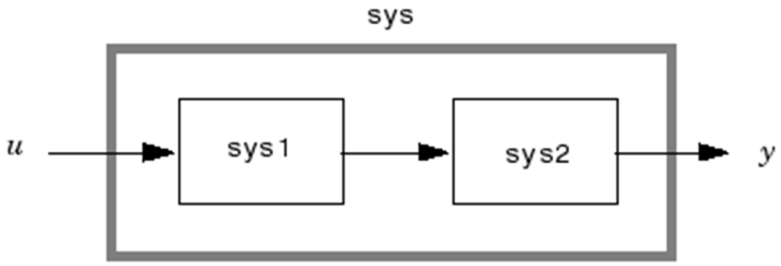


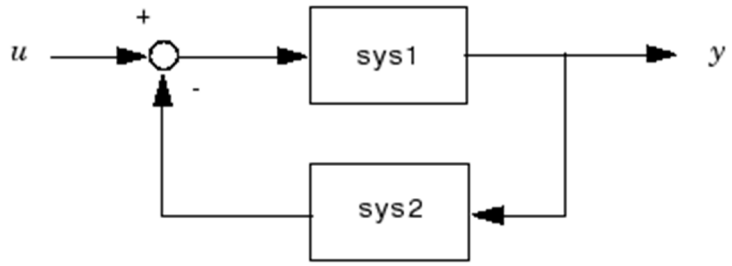
Рис. 3.2. АФХ об’єкту дослідження

3. Вибір і настройка автоматичного регулятора

В якості регулятора виберемо один з типових регуляторів, що реалізують лінійні закони керування. Розглянемо як утворити замкнуту і розімкнуту систему керування. Для утворення складних моделей Control System Toolbox пропонує функції конкатенації (`[,]`, `[;]` та `append`), паралельного і послідовного з’єднання (`parallel` та `series`)

зворотнього зв'язку (`feedback` and `lft`). Ці функції є корисними для моделювання розімкнених і замкнутих систем.

series	Послідовне з'єднання двох LTI - моделей
Синтаксис	`sys = series(sys1,sys2)`
Опис	Ця функція приймає будь-який тип LTI-моделі. Єдине обмеження - обидві системи повинні бути неперервні або дискретні з однаковим часом дискретизації. 

feedback	зворотній зв'язок двох LTI-моделей
Синтаксис	`sys = feedback(sys1,sys2)`
Опис	`sys = feedback(sys1,sys2)` повертає LTI – модель `sys` для від'ємного зворотнього зв'язку. 

Для створення одиничного зворотнього зв'язку використовуються наступні команди:

```
Cloop = feedback(G,1)
Cloop = feedback(1,G)
```

Розглянемо приклад створення розімкненої і замкненої системи для об’єкта, заданого передатною функцією (3.1) та ПІ-регулятора

```
%% represent contoller
Kp = 1;
Ti = 20;
p = tf('p');

Controller = Kp*(1 + 1/(Ti*p))
```

Якщо ввести вказані інструкції, то в командному вікні Matlab отримаємо наступний результат:

```
Transfer function:
20 p + 1
--------
  20 p
```

Одним з нюансів при об’єднанні об’єкта і регулятора є складова транспортного запізнювання в об’єкті. Для застосування кореневих методів синтезу системи керування, дане транспортне запізнювання має бути представлене у вигляді передатної функції відповідної структури. Control System Toolbox підтримує апроксимацію транспортного запізнювання на основі дробу Паде. Синтаксис цікавих для нашого дослідження форматів команди наступний:

pade	зворотній зв’язок двох LTI-моделей
Синтаксис	`sysx = pade(sys,N)`
Опис	повертає вільну від затримок апроксимовану передатну функцію.

Реалізуємо тепер – замкнену і розімкнену системи.

```
%% create openloop/closeloop  system
approximation_order = 2;
Wob_x = pade(Wob, approximation_order);

sys_open = series(Controller, Wob_x);
sys_close = feedback(sys_open, 1)
```

Порядок виконання роботи

1. Задати параметри об’єкту керування згідно варіанту.
2. Написати програму для дослідження характеристик системи керування з використанням інструментарію Control System Toolbox.
3. Виконати дослідження характеристик системи керування залежно від настроювань регулятора.
4. Написати програму, яка б реалізовувала систему заміщення для передатної функції об’єкту керування.
5. Написати програму побудови перехідних характеристик системи без використання вбудованих функцій Control System Toolbox. Дослідити точність програми залежно від кроку інтегрування.
6. Порівняти результати програм для моделювання системи керування, зробити висновки.

Зміст звіту

1. Назва, мета, порядок виконання роботи.
2. Математична модель об’єкта у вигляді дробово-раціональної передатної функції з запізненням.
3. Лістинг програми моделювання з використанням інструментарію Control System Toolbox
4. Лістинг програми моделювання без використанням інструментарію Control System Toolbox
5. Висновки за результатами досліджень.

Контрольні запитання

1. Лінійні закони керування.
2. Система заміщення і модифікована система заміщення.

3. Основні функції інструментарію Control System Toolbox.
4. Замкнена і розімкнена система керування, характеристики системи керування, критерії якості.

Приклад виконання роботи

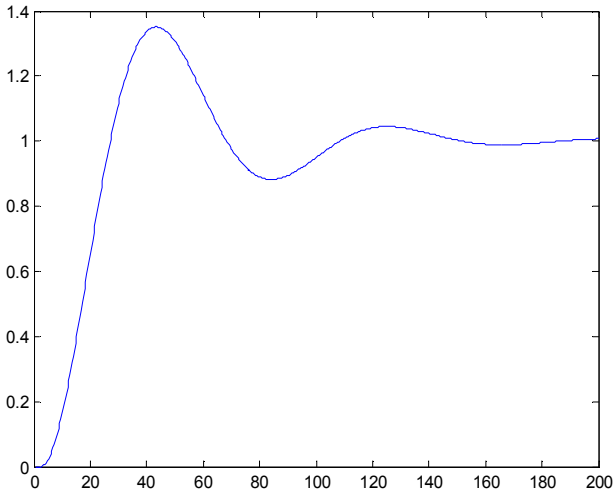
Підпрограма, що реалізує систему заміщення для дробово-раціональної передатної функції:

```
function [Y_current] = eil_wpm(A_coef, B_coef, M_X0, M_X1, v0, Y_prev)

%    eil_wpm - крок за методом Ейлера для модифікованої системи заміщення
%    A_coef - коефіцієнти поліному знаменника,
%    B_coef - коефіцієнти поліному чисельника,
%    M_X0 - значення входів на попередньому кроці інтегрування,
%    M_X1 - значення входів на поточному кроці інтегрування,
%    Y_prev - вихід обєкта до кроку інтегрування,
%    Y_current - вихід обєкта після кроку інтегрування,
%    v0 - крок інтегрування

    inputs_count = size(B_coef, 2);  %кількість входів обєкту
    outputs_count = size(A_coef, 2); %кількість виходів обєкту

    for i = inputs_count + 1 : outputs_count
        B_coef(i) = 0;
    end;

    Y_current(outputs_count + 1) = 0;
    Y_prev(outputs_count + 1) = 0;
    for z = 2 : outputs_count
        Y_current(z) = Y_prev(z) + v0 * (Y_prev(z+1) - A_coef(outputs_count
- z + 1)*Y_prev(1)+ B_coef(outputs_count - z + 1)* M_X0);
    end;
    Y_current(1) = (Y_current(2) + B_coef(outputs_count)* M_X1) /
A_coef(outputs_count);
```

Програма побудови перехідних характеристик системи без використання вбудованих функцій Control System Toolbox:

```
%передатна функція
B(1) = 1;
A(1) = 0; A(2) = 1; A(3) = 15; A(4) = 50;
tau = 2;

%параметри регулятора
Nzr = 4;
Kreg = 0.1;
Ti = 3000;
Tv = 0;
Zavdannya = 1;
```

```
%час спостереження і кількість точок, що будуть розраховані
D = 200;
L = 500;
Ks = 4;
Dt = D / L;
V0 = Dt / Ks;

%ініціалізація масиву транстпортера
Zt = round(tau/V0);
% Zt = 1;
for i = 1 : Zt
    Mz(i) = 0;
end;

%нульові початкові умови
for s = 1 : size(A, 2)
    Y(s) = 0;
end;

Hts(1) = 0; Eps = 0; Integral = 0; X = 0;

for z = 2 : L
    for s = 1 : Ks
        [Eps, Integral, X] = step_reg(Nzr, Kreg, Ti, Tv, Y(1), Zavdannya,
V0, Eps, Integral, X);
         Xt = Mz(Zt);
         for v = Zt : -1 : 2
             Mz(v) = Mz(v-1);
         end;
        Mz(1) = X;

        Y = eil_wpm(A, B, Xt, Mz(Zt), V0, Y);
    end;
    dt(z) = Dt * z;
    Hts(z) = Y(1);
end;

plot(dt, Hts);
```



Лабораторна робота №4
ДОСЛІДЖЕННЯ СТІЙКОСТІ СИСТЕМ КЕРУВАННЯ

Мета роботи - дослідити стійкість систем керування за розміщенням коренів характеристичного полінома, критеріями Гурвіца, Михайлова, Найквіста.

Теоретичні відомості

Стійкість системи за наявності її характеристичного полінома можна визначити дуже просто – треба знайти корені цього полінома і з'ясувати, чи всі вони знаходяться в лівій половині комплексної площини. Для розрахунку коренів та полюсів передатної функції і їх графічного представлення будемо використовувати наступні команди:

esort	Сортує полюси по дійсній частині
Синтаксис	`s = esort(p)`
Опис	Сортує полюси, розміщені у векторі `p`, по дійсній частині. Повертає відсортовані полюси в s.

pole	Розрахунок полюсів LTI – моделі
Синтаксис	`p=pole(sys)`
Опис	`pole(sys)` розраховує полюси `p` SISO чи MIMO LTI – моделі `sys`.

zero	Розрахунок нулів передатної функції
Синтаксис	`tzero (sys)`
Опис	`pzmap(sys)` виводить на екран карту полюсів-нулів неперервної чи дискретної LTI-моделі `sys`. Для For SISO системи, `pzmap` виводить полюси і нулі передатної функції.

Використаємо передатну функцію замкненої системи `sys_close` з попередньої роботи:

```
          20 p^3 - 29 p^2 + 13.5 p + 0.75
-----------------------------------------------
200 p^4 + 340 p^3 + 151 p^2 + 28.5 p + 0.75
```

і застосуємо до неї команди:

```
%% stability by poles
poles = esort(pole(sys_close))
zeros = esort(zero(sys_close))
pzmap(sys_close)
```

Результатом виконання буде:

```
poles =

  -0.0311
  -0.2605 + 0.1930i
  -0.2605 - 0.1930i
  -1.1480

zeros =

   0.7500 + 0.4330i
   0.7500 - 0.4330i
-0.0500
```

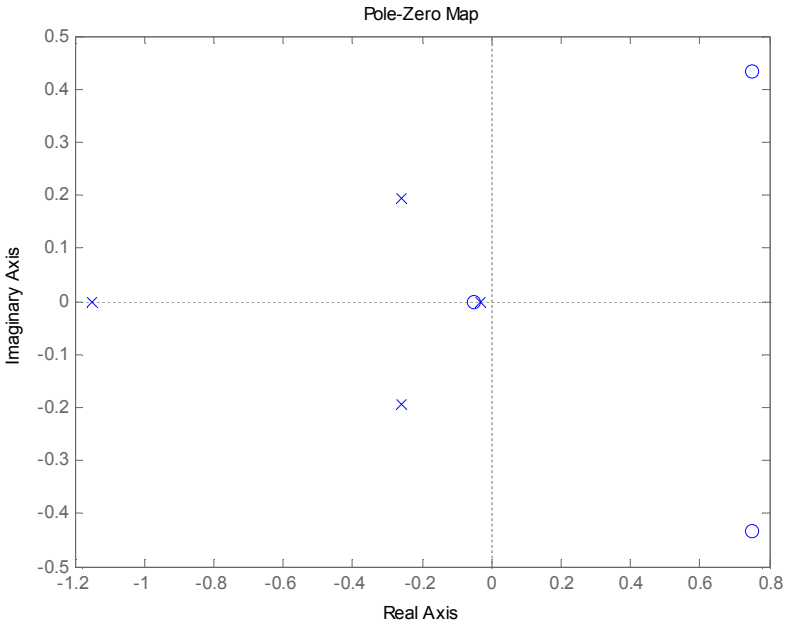


Рис. 4.1. Полюси і нулі передатної функції замкненої системи

Критерій Гурвіца

Ще до порівняно недавнього часу визначення коренів поліномів високого степеня вважалось далеко не тривіальною справою. Звідси природне бажання знайти алгоритм оцінювання стійкості, що був би менш трудомістким, ніж визначення коренів полінома. Для оцінювання стійкості системи знання самих коренів характеристичного полінома, власне, і не потрібне. Досить лише знати їх знаки (знаки дійсних частин). Так розв'язують задачу стійкості за критерієм Гурвіца. Для прикладу розглянемо систему, порядок якої $n = 6$.

Її характеристичний поліном має вигляд

$$A(p) = a_6 p^6 + a_5 p^5 + a_4 p^4 + a_3 p^3 + a_2 p^2 + a_1 p + a_0 . \qquad (4.1)$$

Корені полінома, як відомо, не зміняться, якщо цей поліном помножити або поділити на будь-яке число. Отже, якщо $\alpha_6 < 0$, помножимо всі коефіцієнти полінома $A(p)$ на –1, тобто вважатимемо, що в поліномі $A(p)$ старший коефіцієнт $\alpha_6 > 0$. Якщо б α_6 дорівнював нулю, то відповідним членом в $A(p)$ треба було б знехтувати як таким, якого немає, понижуючи тим самим степінь полінома, і так доти, доки α_n не стане відмінним від нуля, а потім і більшим від нуля. Решта коефіцієнтів мають бути наявними, навіть якщо деякі з них дорівнюють нулю. На базі характеристичного полінома (4.1) сформуємо визначник Гурвіца

$$\Delta = \begin{vmatrix} a_5 & a_3 & a_1 & 0 & 0 & 0 \\ a_6 & a_4 & a_2 & a_0 & 0 & 0 \\ 0 & a_5 & a_3 & a_1 & 0 & 0 \\ 0 & a_6 & a_4 & a_2 & a_0 & 0 \\ 0 & 0 & a_5 & a_3 & a_1 & 0 \\ 0 & 0 & a_6 & a_4 & a_2 & a_0 \end{vmatrix}. \qquad (4.2)$$

Формуємо його за таким алгоритмом. Уздовж головної діагоналі виписуємо послідовно коефіцієнти полінома, починаючи з передстаршого (α_{n-1}). Вниз від кожного діагонального елемента вписуємо коефіцієнти, що стоять в поліномі лівіше від діагонального, а вверх – що правіше від нього. Якщо індекс чергового коефіцієнта менший від нуля або більше *n*, записують нуль.

Критерій Гурвіца формулюється так. Для того, щоб система була стійкою, необхідно і достатньо, щоб при додатному старшому коефіцієнті характеристичного полінома усі діагональні мінори визначника Гурвіца були б строго додатними

Критерій Михайлова

Будемо розглядати графічну інтерпретацію даного критерію, яка полягає в наступному: для того, щоб система була стійкою необхідно щоб годограф характеристичного рівняння замкненої системи по-черзі проходив кількість квадрантів, який порядок характеристичного рівняння. Для розв'язання даної задачі необхідно розв'язати наступні підзадачі:

1. Створити передатну функцію замкненої системи
2. Виокремити знаменник передатної функції замкненої системи
3. Замінити в характеристичному поліномі p на jw та побудувати годограф

Візьмемо передатну функцію замкненої системи з попередньої роботи. Оскільки у нас є передатна функція замкненої системи (представлена у вигляді дробу), то тепер необхідно отримати характеристичний поліном. Функція `tf` «запаковує» дані про модель і час дискретизації в окремий LTI об'єкт. Також існують команди для забезпечення виокремлення даних з об'єкту. Для моделі представленої у вигляді передатної функції такою командою є:

```
[num,den,Ts] = tfdata(sys)     % Ts = sample time
```

Таким чином для отримання характеристичного поліному знаменника передатної функції виконаємо наступні команди

```
% retrive character polinom of closeloop system
[num_koef, den_koef, Ts] = tfdata(sys_close);

den_koef = den_koef{1,1}
num_koef = num_koef{1,1}
```

В результаті чого на екран виведуться коефіцієнти поліному:

```
den_koef =

  200.0000  340.0000  151.0000   28.5000    0.7500
```

Тепер залишається обчислити значення цього поліному в заданому діапазоні частот.

polyval	Розрахунок значення поліному
Синтаксис	`y = polyval(p,x)`
Опис	повертає значення поліному степеня `n`, розраховане в x. Перший аргумент – це вектор довжини `n+1` елементи якого є коефіцієнтами степенів поліному по спаданню.

```
%%Mihaylov
Wbeg = 0.01;              %frequence begin
Wend = 10;
w = Wbeg : 0.001 : Wend;
y = polyval(den_koef, j*w);
```

Побудуємо тепер годограф Михайлова, фактично це буде графік значень y, де по осі x буде дійсна частина, а по осі y - уявна

```
plot(y(:))
grid
```

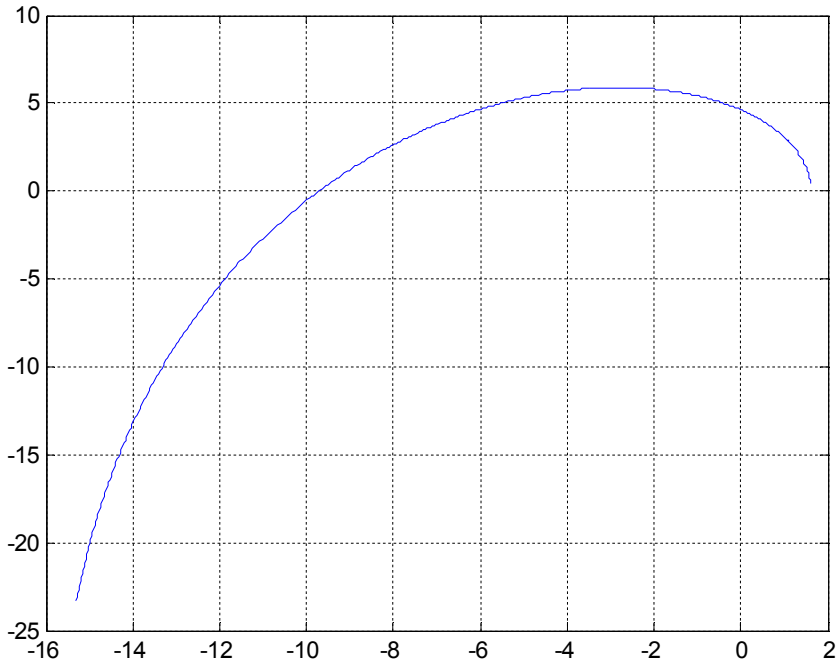


Рис. 4.2. Годограф Михайлова – низькочастотний діапазон (w=0-1)

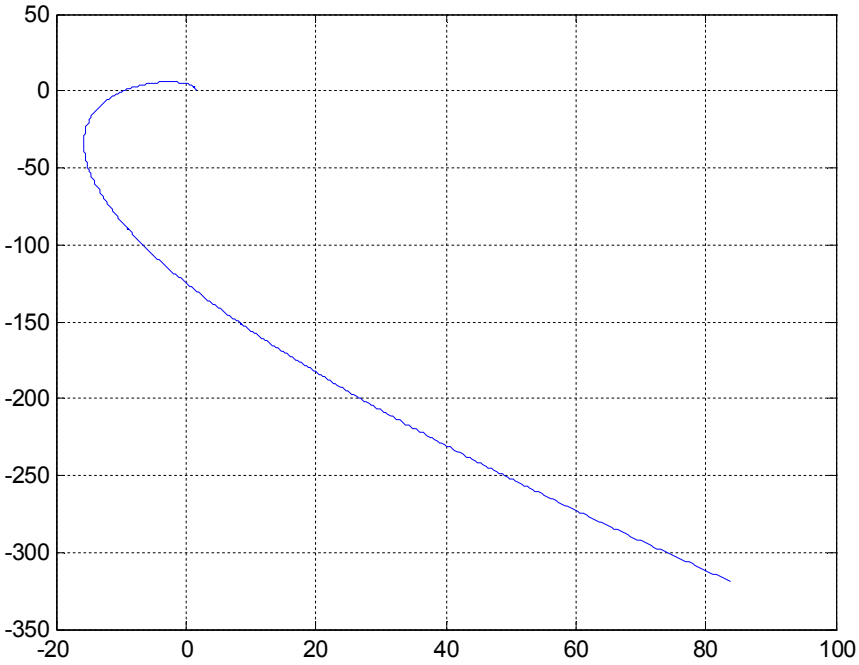


Рис. 4.3. Годограф Михайлова – діапазон (w=1-10)

Як бачимо, годограф по черзі проходить 4 квадранта що відповідає порядку системи. Таким чином, можна зробити висновок про стійкість системи.

Критерій Найквіста

Критерії Гурвіца–Рауса та Михайлова дозволяють оцінювати стійкість систем із зосередженими параметрами, коли порядок системи n – скінченний і, бажано, відносно невисокий. Для дослідження стійкості систем з розподіленими параметрами, коли $n = \infty$, вони явно непридатні. Критерій стійкості Найквіста, дозволяє досліджувати стійкість систем як із зосередженими, так і з розподіленими параметрами (порядок досліджуваної системи в ньому враховувати не треба).

Критерій Найквіста: для того щоб система, стійка в розімкненому стані, залишалась стійкою і після замикання, потрібно і достатньо, щоб годограф розімкненої системи $W_{\text{роз}}(j\omega)$ зі змінюванням частоти ω від 0 до ∞ не охоплював би точку $-1+0j$.

Порядок виконання роботи

1. Задати параметри об’єкту керування згідно варіанту.
2. Створити систему керування з довільними параметрами ПІ-регулятора.
3. Побудувати графік розміщення полюсів системи.
4. Дослідити стійкість системи за критерієм Михайлова.
5. Дослідити стійкість системи за критерієм Гурвиця.
6. Дослідити стійкість системи за критерієм Найквіста.
7. Зробити висновки.

Зміст звіту

1. Назва, мета, порядок виконання роботи.

2. Математична модель об’єкта у вигляді дробово-раціональної передатної функції.

3. Лістинг програм дослідження стійкості системи.

4. Висновки за результатами досліджень.

Контрольні запитання

1. Стійкість системи керування.
2. Алгебраїчні та частотні критерії стійкості.
3. Критерій Михайлова, Найквіста.
4. Критерій Гурвиця.
5. Функції Matlab для визначення полюсів системи керування.

Лабораторна робота №5

РОЗРАХУНОК СИСТЕМИ КЕРУВАННЯ НА ЗАДАНИЙ ПОКАЗНИК КОЛИВНОСТІ

Мета роботи – виконати розрахунок системи керування на заданий показник коливності.

Теоретичні відомості

Для того, щоб замкнена система відповідала заданому показнику коливності (тобто мала відповідний запас стійкості по амплітуді і фазі) необхідно щоб АФХ розімкненої системи дотикалась до М-кола: з центром на дійсній осі на відстані $L = \frac{M_1^2}{M_1^2 - 1}$ від початку координат (зліва, якщо $L > 0$, або відповідно справа на відстані $|L|$, якщо $L < 0$). Радіус кола $r = \frac{M_1}{M_1^2 - 1}$ (якщо $M_1^2 - 1 > 0$ або $r = \frac{M_1}{1 - M_1^2}$, якщо $M_1 < 1$).

Для систем автоматизації технологічних процесів у хімічній, енергетичній, харчовій промисловості можна рекомендувати значення показника коливності $1.1 \le M \le 1.4$. Чим більше M, тим оперативніша система, хоча при цьому збільшується схильність системи до слабозагасаючих коливань.

Розглянемо код Matlab, який реалізує даний алгоритм.

```
(1)  M = 1.3;
(2)  R = M/(M^2-1)
(3)  C = M^2/(M^2-1)
(4)  Centr = [-C; 0];
(5)  t = [0:pi/180:2*pi];
(6)  x = R*cos(t) + Centr(1);
(7)  y = R*sin(t) + Centr(2);
(8)  plot(x,y);
(9)  hold on
(10) Wbeg = 0.01
(11) Wend = 2;
```

```
(12) w = Wbeg:0.01:Wend;
(13) H = freqresp(sys_open, w);
(14) plot(H(:))
(15) grid
(16) axis equal
(17) axis([-5 5 -5 5])
```

Перші 8 строк призначенні для побудови самого М-кола. Задається показник коливності (1), за представленими вище формулами розраховується радіус (2) та відстань від уявної осі до центра (3) даного кола. Строка (4) задає координати центра кола. Строка (5) задає область побудови графіка функції – від 0 до 2π з кроком $\pi/180$. Строки (6)-(7) задають рівняння кола, а строка (8) виводить графік на екран. Строка (9) говорить, що наступний графік, який буде побудований, повинен відображатися в тій же графічній області, що і графік М-кола.

Строки (10-15) будують амплітудно-фазову характеристику розімкненої системи і виводять її на екран. Строки (10)-(12) задають діапазон частот, в якому буде побудована АФХ. Строка (13) розраховує реакцію системи на частотний сигнал за допомогою вбудованої функції `freqresp`.

Строка (14) будує АФХ. Форма запису H(:) означає, що для першої координати графіка (дійсна частина) буде використаний елемент з першого стовпчика матриці H, для другої координати графіка (уявна частина) буде використаний елемент з другого стовпчика H. Строки (15)-(16) встановлюють однаковий масштаб по осям, щоб уникнути спотворення графіку М-кола.

Підбір параметрів регулятора здійснюється ітеративно:

1. задаються параметри регулятора і будується передатна функція розімкненої системи;
2. будується графік – М-кола та АФХ розімкненої системи;

3. Аналізуються графіки і в разі необхідності виконується послідовність кроків, починаючи з першого.

Для ПІ-регулятора з параметрами $k = 1$, $T_i = 20$ маємо:

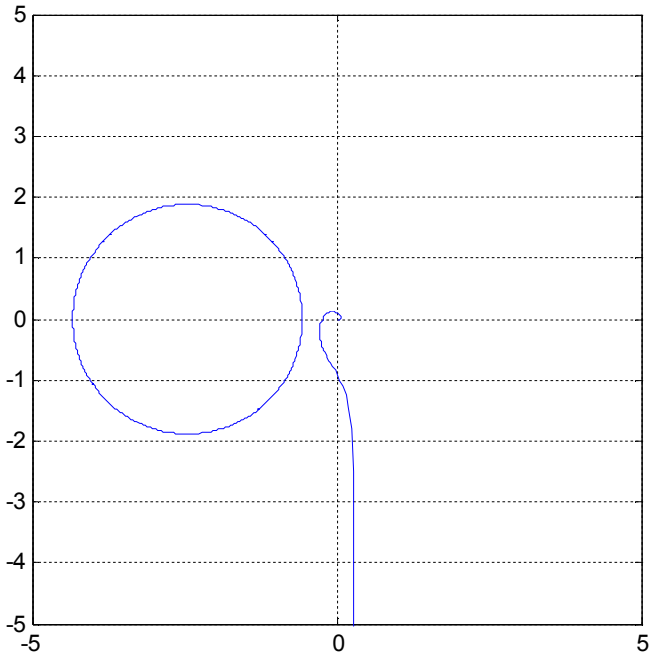


Рис. 5.1 М-коло при параметрах регулятора $k = 1$, $T_i = 20$

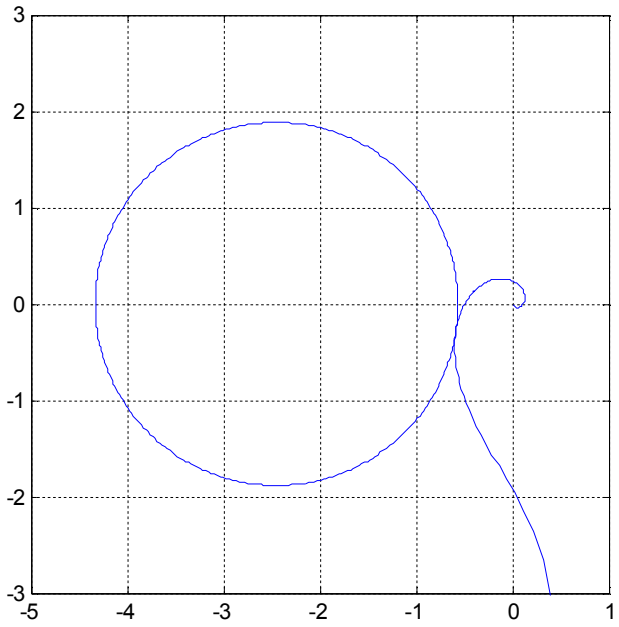


Рис. 5.2 М-коло при параметрах регулятора $k = 2.15$, $T_i = 20$

Для побудови перехідних процесів замкненої системи використаємо функції step та impulse, на вхід яких подамо уже lti-модель замкненої системи із оптимальними параметрами регулятора. Результати досліджень подані нижче.

```
%% dynamics of closeloop system
time_begin = 0;
time_end = 80;
Time = time_begin : 0.1 : time_end;

subplot(2, 1, 1), step(sys_close, Time)
subplot(2, 1, 2), impulse(sys_close, Time)
```

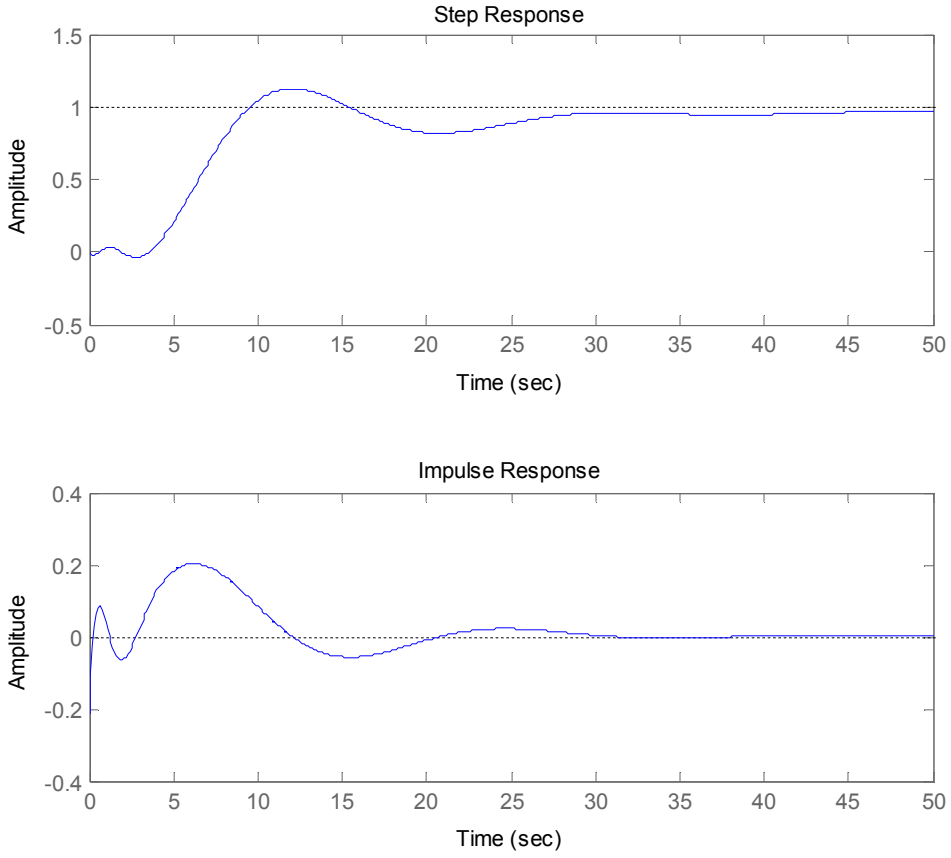


Рис. 5.3 Перехідна та імпульсні характеристики замкненої системи

Порядок виконання роботи

1. Задати параметри об’єкту керування згідно варіанту.
2. Створити систему керування з довільними параметрами ПІ-регулятора.
3. Настроїти систему на заданий показник коливності.
4. Побудувати характеристики замкненої системи.
5. Зробити висновки.

Зміст звіту

1. Назва, мета, порядок виконання роботи.
2. Математична модель об’єкта у вигляді дробово-раціональної передатної функції.
3. Лістинг програми настроювання системи.
4. Результати (графіки) роботи програми.
5. Висновки за результатами досліджень.

Контрольні запитання

1. Стійкість системи керування, запас стійкості.
2. Метод М-кола.
3. Розрахунок координат центру М-кола.
4. Вибір показника коливності M.

Список рекомендованої літератури

1. Кубрак А.И., Голинко И.М., Ситников А.В. Численный анализ и программирование. – Кам.–Под.: Каліграф, 2008. –256с.
2. Дорф Р. Современные системы управления. / Р. Дорф, Р. Бишоп. – М.: Лаборатория базовых знаний, 2002. – 832с
3. Остапенко Ю. О. Ідентифікація та моделювання технологічних об’єктів керування: Підручник для студентів вищих закладів освіти, що навчаються за напрямом “Автоматизація та комп’ютерно-інтегровані технології ” – Київ.: Задруга, 1999. – 424 с.: Іл.. – Бібліогр. в кінці розділів.
4. С.В. Поршнев, MATLAB 7. Основы работы и программирования. − Бином-Пресс. 2006.
5. Медведев В.С. Control System Toolbox. Matlab 5 для студентов / В.С. Медведев, В.Г. Потемкин, М.: ДИАЛОГ- МИФИ, 1999. – 287с. (Пакеты прикладных программ) – ISBN 5-86404-136-х
6. Control System Toolbox User’s Guide, version 4.2