# **Приложение 1**

clc

clear

% Определение параметров системы

H = 5;

M = 0.6;

V = 192;

Zb = -0.2;

sin = 0.08;

cos = 1;

g\_del\_V = 0.051;

M\_x\_b = -5.8;

M\_x\_omx = -1;

M\_x\_omy = -0.2;

M\_x\_sigme = -7;

M\_y\_b = -3;

M\_y\_omx = -0.05;

M\_y\_omy = -0.2;

M\_y\_sigmn = -2.5;

om\_pr = 20;

kzi\_pr = sqrt(2)/2;

% Синтез параметров

a = 4; % Коэффициент пропорциональности между tpp и временем релаксации

kzi = sqrt(a^2/(pi^2+a^2)); % Коэффициент затухания

t\_pp = 1; % Задаём примерное время переходного процесса

om0 = a / kzi / t\_pp; % Определяем собственную частоту колебаний

T = 2\*pi / (om0\*sqrt(1-kzi^2)); % Определяем время релаксации

disp('Синтезированные коэффициенты')

k\_sigm = - om0^2 / M\_x\_sigme % Значение коэффициент k\_sigm

k\_omx = - (M\_x\_omx + 2\*kzi\*om0) / M\_x\_sigme % Определяем коэффициент k\_omx

% Построение переходного процесса в системе с учётом привода

% с синтезированными коэффициентами

W1 = tf(M\_x\_sigme, [1 -M\_x\_omx]); % ПФ по угловой скорости

W2 = tf(om\_pr^2, [1 2\*kzi\_pr\*om\_pr om\_pr^2]); % ПФ привода

W3 = W1\*W2; % Связываем W1 и W2

integr = tf(1, [1, 0]); % ПФ интегратора

W4 = feedback(W3, k\_omx, 1); % Положительная ОС по угловой скорости

W5 = k\_sigm \* W4 \* integr; % Разомкнутая цепь по углу крена

disp('ПФ замкнутой цепи по углу крена с учётом привода')

W6 = feedback(W5, 1, 1) % Замкнутая цепь по углу крена

disp('Корни характеристического уравнения')

roots(W6.denominator{1}) % Корни характеристического уравнения

figure;

step(-1\*W6); % Построение переходного процесса

% Построение переходного процесса в системе без учёта привода

% с синтезированными коэффициентами

W1 = tf(M\_x\_sigme, [1 -M\_x\_omx]); % ПФ по угловой скорости

integr = tf(1, [1, 0]); % ПФ интегратора

W2 = feedback(W1, k\_omx, 1); % Положительная ОС по угловой скорости

W3 = k\_sigm \* W2 \* integr; % Разомкнутая цепь по углу крена

disp('ПФ замкнутой цепи по углу крена без учёта привода')

W4 = feedback(W3, 1, 1) % Замкнутая цепь по углу крена

disp('Корни характеристического уравнения')

roots(W4.denominator{1}) % Корни характеристического уравнения

hold on

step(-1\*W4); % Построение переходного процесса

hold off

grid on

legend('С учётом динамики привода','Без учёта динамики привода')

% Корни, полученные при моделирования в NCD-блоке

disp('Плученные в NCD-блоке коэффициенты')

k\_sigm = 5.4231

k\_omx = 1.2597

% Те же самые построение только уже для подобранных в NCD-блоке

% коэффициентов

W1 = tf(M\_x\_sigme, [1 -M\_x\_omx]);

W2 = tf(om\_pr^2, [1 2\*kzi\_pr\*om\_pr om\_pr^2]);

W3 = W1\*W2;

integr = tf(1, [1, 0]);

W4 = feedback(W3, k\_omx, 1);

W5 = k\_sigm \* W4 \* integr;

disp('ПФ замкнутой цепи по углу крена с учётом привода')

W6 = feedback(W5, 1, 1)

disp('Корни характеристического уравнения')

roots(W6.denominator{1})

figure;

step(-1\*W6)

W1 = tf(M\_x\_sigme, [1 -M\_x\_omx]);

integr = tf(1, [1, 0]);

W2 = feedback(W1, k\_omx, 1);

W3 = k\_sigm \* W2 \* integr;

disp('ПФ замкнутой цепи по углу крена без учёта привода')

W4 = feedback(W3, 1, 1)

disp('Корни характеристического уравнения')

roots(W4.denominator{1})

hold on

step(-1\*W4)

hold off

legend('С учётом динамики привода','Без учёта динамики привода')

grid on

Программа 1. Основной код для системы управления стабилизацией крена

clc

clear

syms k\_sigm k\_omx s

% Определение параметров системы

H = 5;

M = 0.6;

V = 192;

Zb = -0.2;

sin = 0.08;

cos = 1;

g\_del\_V = 0.051;

M\_x\_b = -5.8;

M\_x\_omx = -1;

M\_x\_omy = -0.2;

M\_x\_sigme = -7;

M\_y\_b = -3;

M\_y\_omx = -0.05;

M\_y\_omy = -0.2;

M\_y\_sigmn = -2.5;

om\_pr = 20;

kzi\_pr = sqrt(2)/2;

W\_pr = om\_pr^2 / (s^2 + 2\*om\_pr\*kzi\_pr\*s + om\_pr^2); % ПФ привода

W1 = collect(M\_x\_sigme / (s - M\_x\_omx)); % ПФ по угловой скорости

W2 = collect(W1 \* W\_pr); % Связываем W1 и W2

W3 = collect(W2 / (1 - W2\*k\_omx)); % Положительная ОС по угловой скорости

W4 = collect(k\_sigm \* W3 / s); % Разомкнутая цепь по углу крена

W5 = collect(W4 / (1 - W4)); % Замкнутая цепь по углу крена

disp('ПФ замкнутой цепи по углу крена с учётом привода')

pretty(W5)

[num, den] = numden(W5);

disp('Числитель передаточной функции')

num = vpa(collect(num), 4)

disp('Знаменатель передаточной функции')

den = vpa(collect(den), 4)

% Выделим коэффициенты в числителе и знаменателе

% Функция coeffs запишет их от младшей степени к старшей

koefs\_num = coeffs(num, 's'); % Массив коэффициентов числителя

koefs\_den = coeffs(den, 's'); % Массив коэффициентов знаменателя

% Перевернём массивы на 180 градусов, чтобы получить коэффициенты

% от старшего ко младшему

koefs\_num = rot90(rot90(koefs\_num));

koefs\_den = rot90(rot90(koefs\_den));

% Запишем соответствующие коэффициенты в соответствующие переменные

B0 = koefs\_num(1);

A4 = koefs\_den(1);

A3 = koefs\_den(2);

A2 = koefs\_den(3);

A1 = koefs\_den(4);

A0 = koefs\_den(5);

% Запишем неравенства жля критерия Рауса

eq1 = A0 > 0;

eq2 = A1 > 0;

eq3 = A2 > 0;

eq4 = A3 > 0;

eq5 = A4 > 0;

eq6 = simplify(A1\*A2\*A3 - A1^2\*A4 - A0\*A3^2) > 0;

% Преобразуем неравенства из символьного типа данных в функции

eq1 = matlabFunction(eq1);

eq2 = matlabFunction(eq2);

eq3 = matlabFunction(eq3);

eq4 = matlabFunction(eq4);

eq5 = matlabFunction(eq5);

eq6 = matlabFunction(eq6);

% Задание сетки

l1 = [-2:0.005:5];

l2 = [-2:0.005:20];

[ko, ks] = meshgrid(l1, l2);

% Проверка устойчивости узлов сетки

c1 = eq1(ks);

c2 = eq2(ko);

c3 = eq3();

c4 = eq4();

c5 = eq5();

c6 = eq6(ko, ks);

figure;

contourf(l1, l2, c1 & c2 & c3 & c4 & c5 & c6, [1 1 1 1 1 1])

colormap lines

grid on

hold on

% Синтезированные коэффициенты

k\_omx = 1;

k\_sigm = 3.6957;

pnt1 = scatter(k\_omx, k\_sigm, 'r','filled');

% Подобранные коэффициенты

k\_omx = 1.2597;

k\_sigm = 5.4231;

pnt2 = scatter(k\_omx, k\_sigm, 'b','filled');

legend([pnt1, pnt2],"Синтезированные коэффициенты", "Подобранные коэффициенты")

hold off

xlabel('Komx')

ylabel('Ksigm')

Программа 2. Область устойчивости для системы управления стабилизацией крена

clc

clear

syms k\_sigm k\_omx s

% Определение параметров системы

H = 5;

M = 0.6;

V = 192;

Zb = -0.2;

sin = 0.08;

cos = 1;

g\_del\_V = 0.051;

M\_x\_b = -5.8;

M\_x\_omx = -1;

M\_x\_omy = -0.2;

M\_x\_sigme = -7;

M\_y\_b = -3;

M\_y\_omx = -0.05;

M\_y\_omy = -0.2;

M\_y\_sigmn = -2.5;

om\_pr = 20;

kzi\_pr = sqrt(2)/2;

W\_pr = om\_pr^2 / (s^2 + 2\*om\_pr\*kzi\_pr\*s + om\_pr^2); % ПФ привода

W1 = collect(M\_x\_sigme / (s - M\_x\_omx)); % ПФ по угловой скорости

W2 = collect(W1 \* W\_pr); % Связываем W1 и W2

W3 = collect(W2 / (1 - W2\*k\_omx)); % Положительная ОС по угловой скорости

W4 = collect(k\_sigm \* W3 / s); % Разомкнутая цепь по углу крена

W5 = collect(W4 / (1 - W4)); % Замкнутая цепь по углу крена

disp('ПФ замкнутой цепи по углу крена с учётом привода')

pretty(W5)

[num, den] = numden(W5);

disp('Числитель передаточной функции')

num = vpa(collect(num), 4)

disp('Знаменатель передаточной функции')

den = vpa(collect(den), 4)

% Выделим коэффициенты в числителе и знаменателе

% Функция coeffs запишет их от младшей степени к старшей

koefs\_num = coeffs(num, 's'); % Массив коэффициентов числителя

koefs\_den = coeffs(den, 's'); % Массив коэффициентов знаменателя

% Перевернём массивы на 180 градусов, чтобы получить коэффициенты

% от старшего ко младшему

koefs\_num = rot90(rot90(koefs\_num));

koefs\_den = rot90(rot90(koefs\_den));

% Запишем соответствующие коэффициенты в соответствующие переменные

B0 = koefs\_num(1);

A4 = koefs\_den(1);

A3 = koefs\_den(2);

A2 = koefs\_den(3);

A1 = koefs\_den(4);

A0 = koefs\_den(5);

% Запишем неравенства для достаточных условий устойчивости через лямбды

lambd\_star = 2.15;

eq1d = A0^(-1) \* A1 \* A2 \* A3^(-1) >= lambd\_star;

eq2d = A1^(-1) \* A2 \* A3 \* A4^(-1) >= lambd\_star;

% Преобразуем неравенства из символьного типа данных в функции

eq1d = matlabFunction(eq1d);

eq2d = matlabFunction(eq2d);

% Запишем неравенства жля критерия Рауса

eq1r = A0 > 0;

eq2r = A1 > 0;

eq3r = A2 > 0;

eq4r = A3 > 0;

eq5r = A4 > 0;

eq6r = simplify(A1\*A2\*A3 - A1^2\*A4 - A0\*A3^2) > 0;

% Преобразуем неравенства из символьного типа данных в функции

eq1r = matlabFunction(eq1r);

eq2r = matlabFunction(eq2r);

eq3r = matlabFunction(eq3r);

eq4r = matlabFunction(eq4r);

eq5r = matlabFunction(eq5r);

eq6r = matlabFunction(eq6r);

% Задание сетки

l1 = [-2:0.005:5];

l2 = [-2:0.005:20];

[ko, ks] = meshgrid(l1, l2);

% Проверка устойчивости узлов сетки на достаточные условия устойчивости

c1d = eq1d(ko, ks);

c2d = eq2d(ko);

% Проверка устойчивости узлов сетки по критерию Рауса

c1r = eq1r(ks);

c2r = eq2r(ko);

c3r = eq3r();

c4r = eq4r();

c5r = eq5r();

c6r = eq6r(ko, ks);

figure;

grid on

hold on

% Контур для области устойчивости по критерию Рауса

contourf(l1, l2, c1r & c2r & c3r & c4r & c5r & c6r, [1 1 1 1 1 1], FaceAlpha=0.5, FaceColor="red")

% Контур для области устойчивости по достаточным условиям устойчивости

contourf(l1, l2, c1d & c2d, [1 1], FaceAlpha=0.5, FaceColor="blue")

title('Красная область - достаточные условия устойчивости. Синяя область - критерий Рауса')

% Синтезированные коэффициенты

k\_omx = 1;

k\_sigm = 3.6957;

pnt1 = scatter(k\_omx, k\_sigm, 'y','filled');

% Подобранные коэффициенты

k\_omx = 1.2597;

k\_sigm = 5.4231;

pnt2 = scatter(k\_omx, k\_sigm, 'b','filled');

legend([pnt1, pnt2],"Синтезированные коэффициенты", "Подобранные коэффициенты")

xlabel('Komx')

ylabel('Ksigm')

Программа 3. Программа для областей устойчивости для САУ крена для задания на зачёт

# **Приложение 2**

clc

clear

syms k\_psi k\_omegay s

% Определение параметров системы

H = 5;

M = 0.6;

V = 192;

Zb = -0.2;

sin = 0.08;

cos = 1;

g\_del\_V = 0.051;

M\_x\_b = -5.8;

M\_x\_omx = -1;

M\_x\_omy = -0.2;

M\_x\_sigme = -7;

M\_y\_b = -3;

M\_y\_omx = -0.05;

M\_y\_omy = -0.2;

M\_y\_sigmn = -2.5;

om\_pr = 20;

kzi\_pr = sqrt(2)/2;

% Передаточная функция привода

W\_pr = om\_pr^2 / (s^2 + 2\*om\_pr\*kzi\_pr\*s + om\_pr^2);

% Передаточная функция по угловой скорости рысканья с учётом привода

W\_sigmn\_omy\_raz = collect((M\_y\_sigmn\*s + (-M\_y\_sigmn\*Zb)) / (s^2 + (-M\_y\_omy-Zb)\*s + (M\_y\_omy\*Zb-M\_y\_b)) \* W\_pr);

% Обратная связь с коэффициентом обратной связи k\_omegay

W\_sigmn\_omy\_zamk = collect(W\_sigmn\_omy\_raz / (1 - k\_omegay\*W\_sigmn\_omy\_raz));

% Разомкнутая передаточная функция по углу рысканья с замкнутым внутренним

% контуром

W\_psizad\_psi\_raz = collect(k\_psi \* W\_sigmn\_omy\_zamk / s);

% Передаточная функция замкнутой системы по углу рысканья

W\_psizad\_psi\_zamk = collect(W\_psizad\_psi\_raz / (1 - W\_psizad\_psi\_raz));

% Выделим числитель и знаменатель итоговой ПФ

[num, den] = numden(W\_psizad\_psi\_zamk);

num = collect(num);

den = collect(den);

% Выделим коэффициенты числителя и знаменателя

% При этом сами коэффициенты будут записаны в порядке от младшего к

% старшему, то есть перевернутся

koefs\_num = coeffs(num, 's');

koefs\_den = coeffs(den, 's');

% Выделим старший коэффициент знаменателя. В списке он будет последний

starsh\_koef = koefs\_den(length(koefs\_den));

% Поделим все коэффициенты на это число

koefs\_num = vpa(koefs\_num / starsh\_koef);

koefs\_den = vpa(koefs\_den / starsh\_koef);

% Развернём оба списка с коэффициентами, чтобы они шли от старшего ко

% младшему

koefs\_num = rot90(rot90(koefs\_num));

koefs\_den = rot90(rot90(koefs\_den));

% Преобразованные коэффициенты передаточной функции

B1 = koefs\_num(1);

B0 = koefs\_num(2);

A5 = koefs\_den(1);

A4 = koefs\_den(2);

A3 = koefs\_den(3);

A2 = koefs\_den(4);

A1 = koefs\_den(5);

A0 = koefs\_den(6);

% Передаточная функция в нормальном виде (старший коэффициент знаменателя

% равен 1

num = (B1\*s + B0);

den = (A5\*s^5 + A4\*s^4 + A3\*s^3 + A2\*s^2 + A1\*s + A0);

PF = (num / den);

disp('Числитель ПФ')

pretty(num)

disp('Знаменатель ПФ')

pretty(den)

% Зададим параметры для оптимизации в fmicon

k0 = [10, 10];

A = [];

b = [];

Aeg = [];

beg = [];

lb = 0.1\*ones(2,1);

ub = 300\*ones(2,1);

% Осуществим параметрическую оптимизацию

[x, fval] = fmincon('fun\_min', k0, A, b, Aeg, beg, lb, ub, 'nonclon');

% Полученные коэффициенты

k\_omegay = x(1)

k\_psi = x(2)

% Преобразовывает коэффициенты характеристического полинома,

% подставляя в них найденные значения

B\_1 = vpa(-1000.0\*x(2));

B\_0 = vpa(-200.0\*x(2));

A\_5 = vpa(1);

A\_4 = vpa(28.6842);

A\_3 = vpa(414.3537);

A\_2 = vpa(1000\*x(1) + 245.98);

A\_1 = vpa(1000\*x(2) + 200\*x(1) + 1216);

A\_0 = vpa(200.0.\*x(2));

num = B\_1\*s + B\_0;

den = A\_5\*s^5 + A\_4\*s^4 + A\_3\*s^3 + A\_2\*s^2 + A\_1\*s + A\_0;

num = sym2poly(num);

den = sym2poly(den);

% Получаем передаточную функцию и строим переходный процесс

disp('Передаточная функция с полученными коэффициентами')

W\_sys = tf(num, den)

figure;

step(-1\*W\_sys)

grid on

Программа 4. Основной код для системы управление рысканием в режиме плоского разворота

function f = fun\_min(x)

% x(1) = k\_omegay, x(2) = k\_psi

A\_1 = 1000\*x(2) + 200\*x(1) + 1216;

A\_0 = 200.0\*x(2);

f = A\_1 \* A\_0^(-1);

end

Программа 5. Целевая функция для системы управления углом рыскания в режиме плоского разворота

function [c, seq] = nonclon(x)

% x(1) = k\_omegay, x(2) = k\_psi

A\_5 = 1;

A\_4 = 28.6842;

A\_3 = 414.3537;

A\_2 = 1000\*x(1) + 245.98;

A\_1 = 1000\*x(2) + 200\*x(1) + 1216;

A\_0 = 200.0\*x(2);

lambd\_star = 2.15; % Желаемое значение лямбды

lambd1 = lambd\_star - (A\_0^(-1) \* A\_1 \* A\_2 \* A\_3^(-1));

lambd2 = lambd\_star - (A\_1^(-1) \* A\_2 \* A\_3 \* A\_4^(-1));

lambd3 = lambd\_star - (A\_2^(-1) \* A\_3 \* A\_4 \* A\_5^(-1));

sigm\_star = 2; % Желаемое значение сигмы

sigm1 = sigm\_star - (A\_0^(-1) \* A\_1^2 \* A\_2^(-1));

sigm2 = sigm\_star - (A\_1^(-1) \* A\_2^2 \* A\_3^(-1));

sigm3 = sigm\_star - (A\_2^(-1) \* A\_3^2 \* A\_4^(-1));

c = [lambd1; lambd2; lambd3; sigm1; sigm2; sigm3];

seq = [];

end

Программа 6. Ограничения для системы управления углом рыскания в режиме плоского разворота

clc

clear

syms k\_psi k\_omegay s

% Задаём параметры системы

H = 5;

M = 0.6;

V = 192;

Zb = -0.2;

sin = 0.08;

cos = 1;

g\_del\_V = 0.051;

M\_x\_b = -5.8;

M\_x\_omx = -1;

M\_x\_omy = -0.2;

M\_x\_sigme = -7;

M\_y\_b = -3;

M\_y\_omx = -0.05;

M\_y\_omy = -0.2;

M\_y\_sigmn = -2.5;

om\_pr = 20;

kzi\_pr = 0.707;

% Передаточная функция привода

W\_pr = om\_pr^2 / (s^2 + 2\*om\_pr\*kzi\_pr\*s + om\_pr^2);

% Передаточная функция по угловой скорости рысканья с учётом привода

W\_sigmn\_omy\_raz = collect((M\_y\_sigmn\*s + (-M\_y\_sigmn\*Zb)) / (s^2 + (-M\_y\_omy-Zb)\*s + (M\_y\_omy\*Zb-M\_y\_b)) \* W\_pr);

% Обратная связь с коэффициентом обратной связи k\_omegay

W\_sigmn\_omy\_zamk = collect(W\_sigmn\_omy\_raz / (1 - k\_omegay\*W\_sigmn\_omy\_raz));

% Разомкнутая передаточная функция по углу рысканья с замкнутым внутренним

% контуром

W\_psizad\_psi\_raz = collect(k\_psi \* W\_sigmn\_omy\_zamk / s);

% Передаточная функция замкнутой системы по углу рысканья

W\_psizad\_psi\_zamk = collect(W\_psizad\_psi\_raz / (1 - W\_psizad\_psi\_raz));

% Выделим числитель и знаменатель итоговой ПФ

[num, den] = numden(W\_psizad\_psi\_zamk);

num = collect(num);

den = collect(den);

% Выделим коэффициенты числителя и знаменателя

% При этом сами коэффициенты будут записаны в порядке от младшего к

% старшему, то есть перевернутся

koefs\_num = coeffs(num, 's');

koefs\_den = coeffs(den, 's');

% Выделим старший коэффициент знаменателя. В списке он будет последний

starsh\_koef = koefs\_den(length(koefs\_den));

% Поделим все коэффициенты на это число

koefs\_num = vpa(koefs\_num / starsh\_koef);

koefs\_den = vpa(koefs\_den / starsh\_koef);

% Развернём оба списка с коэффициентами, чтобы они шли от старшего ко

% младшему

koefs\_num = rot90(rot90(koefs\_num));

koefs\_den = rot90(rot90(koefs\_den));

% Преобразованные коэффициенты передаточной функции

B1 = koefs\_num(1);

B0 = koefs\_num(2);

A5 = koefs\_den(1);

A4 = koefs\_den(2);

A3 = koefs\_den(3);

A2 = koefs\_den(4);

A1 = koefs\_den(5);

A0 = koefs\_den(6);

% Передаточная функция в нормальном виде (старший коэффициент знаменателя

% равен 1

num = (B1\*s + B0);

den = (A5\*s^5 + A4\*s^4 + A3\*s^3 + A2\*s^2 + A1\*s + A0);

PF = (num / den);

disp('Числитель ПФ')

pretty(num)

disp('Знаменатель ПФ')

pretty(den)

% Запишем неравенства ждя критерия Рауса

eq1 = A0 > 0;

eq2 = A1 > 0;

eq3 = A2 > 0;

eq4 = A3 > 0;

eq5 = A4 > 0;

eq6 = A5 > 0;

eq7 = simplify((A1\*A2-A0\*A3)\*(A3\*A4-A2\*A5) - (A1\*A4-A0\*A5)^2) > 0;

% Преобразуем неравенства из символьного типа данных в функции

eq1 = matlabFunction(eq1);

eq2 = matlabFunction(eq2);

eq3 = matlabFunction(eq3);

eq4 = matlabFunction(eq4);

eq5 = matlabFunction(eq5);

eq6 = matlabFunction(eq6);

eq7 = matlabFunction(eq7);

% Задание сетки

l1 = [-2:0.05:50];

l2 = [-2:0.05:50];

[ko, kp] = meshgrid(l1, l2);

% Проверка устойчивости узлов сетки

c1 = eq1(kp);

c2 = eq2(kp, ko);

c3 = eq3(ko);

c4 = eq4();

c5 = eq5();

c6 = eq6();

c7 = eq7(kp, ko);

% Построение области устойчивости

figure;

contourf(l1, l2, c1 & c2 & c3 & c4 & c5 & c6 & c7, [1 1 1 1 1 1 1])

colormap lines

hold on

% Синтезированные коэффициенты

k\_omegay = 2.7468;

k\_psi = 9.0425;

pnt1 = scatter(k\_omegay, k\_psi,'r','filled');

% Подобранные коэффициенты

k\_omegay = 2.8172;

k\_psi = 9.0425;

pnt2 = scatter(k\_omegay, k\_psi, 'b','filled');

legend([pnt1, pnt2],"Синтезированные коэффициенты", "Подобранные коэффициенты")

hold off

xlabel("Komy")

ylabel("Kpsi")

grid on

Программа 7. Построение области устойчивости для системы управления углом рыскания в режиме плоского разворота

clc

clear

syms k\_psi k\_omegay s

% Задаём параметры системы

H = 5;

M = 0.6;

V = 192;

Zb = -0.2;

sin = 0.08;

cos = 1;

g\_del\_V = 0.051;

M\_x\_b = -5.8;

M\_x\_omx = -1;

M\_x\_omy = -0.2;

M\_x\_sigme = -7;

M\_y\_b = -3;

M\_y\_omx = -0.05;

M\_y\_omy = -0.2;

M\_y\_sigmn = -2.5;

om\_pr = 20;

kzi\_pr = 0.707;

% Передаточная функция привода

W\_pr = om\_pr^2 / (s^2 + 2\*om\_pr\*kzi\_pr\*s + om\_pr^2);

% Передаточная функция по угловой скорости рысканья с учётом привода

W\_sigmn\_omy\_raz = collect((M\_y\_sigmn\*s + (-M\_y\_sigmn\*Zb)) / (s^2 + (-M\_y\_omy-Zb)\*s + (M\_y\_omy\*Zb-M\_y\_b)) \* W\_pr);

% Обратная связь с коэффициентом обратной связи k\_omegay

W\_sigmn\_omy\_zamk = collect(W\_sigmn\_omy\_raz / (1 - k\_omegay\*W\_sigmn\_omy\_raz));

% Разомкнутая передаточная функция по углу рысканья с замкнутым внутренним

% контуром

W\_psizad\_psi\_raz = collect(k\_psi \* W\_sigmn\_omy\_zamk / s);

% Передаточная функция замкнутой системы по углу рысканья

W\_psizad\_psi\_zamk = collect(W\_psizad\_psi\_raz / (1 - W\_psizad\_psi\_raz));

% Выделим числитель и знаменатель итоговой ПФ

[num, den] = numden(W\_psizad\_psi\_zamk);

num = collect(num);

den = collect(den);

% Выделим коэффициенты числителя и знаменателя

% При этом сами коэффициенты будут записаны в порядке от младшего к

% старшему, то есть перевернутся

koefs\_num = coeffs(num, 's');

koefs\_den = coeffs(den, 's');

% Выделим старший коэффициент знаменателя. В списке он будет последний

starsh\_koef = koefs\_den(length(koefs\_den));

% Поделим все коэффициенты на это число

koefs\_num = vpa(koefs\_num / starsh\_koef);

koefs\_den = vpa(koefs\_den / starsh\_koef);

% Развернём оба списка с коэффициентами, чтобы они шли от старшего ко

% младшему

koefs\_num = rot90(rot90(koefs\_num));

koefs\_den = rot90(rot90(koefs\_den));

% Преобразованные коэффициенты передаточной функции

B1 = koefs\_num(1);

B0 = koefs\_num(2);

A5 = koefs\_den(1);

A4 = koefs\_den(2);

A3 = koefs\_den(3);

A2 = koefs\_den(4);

A1 = koefs\_den(5);

A0 = koefs\_den(6);

% Запишем неравенства для достаточных условий устойчивости через лямбды

lambd\_star = 2.15;

eq1d = A0^(-1) \* A1 \* A2 \* A3^(-1) >= lambd\_star;

eq2d = A1^(-1) \* A2 \* A3 \* A4^(-1) >= lambd\_star;

eq3d = A2^(-1) \* A3 \* A4 \* A5^(-1) >= lambd\_star;

% Преобразуем неравенства из символьного типа данных в функции

eq1d = matlabFunction(eq1d);

eq2d = matlabFunction(eq2d);

eq3d = matlabFunction(eq3d);

% Запишем неравенства для критерия Рауса

eq1r = A0 > 0;

eq2r = A1 > 0;

eq3r = A2 > 0;

eq4r = A3 > 0;

eq5r = A4 > 0;

eq6r = A5 > 0;

eq7r = simplify((A1\*A2-A0\*A3)\*(A3\*A4-A2\*A5) - (A1\*A4-A0\*A5)^2) > 0;

% Преобразуем неравенства из символьного типа данных в функции

eq1r = matlabFunction(eq1r);

eq2r = matlabFunction(eq2r);

eq3r = matlabFunction(eq3r);

eq4r = matlabFunction(eq4r);

eq5r = matlabFunction(eq5r);

eq6r = matlabFunction(eq6r);

eq7r = matlabFunction(eq7r);

% Задание сетки

l1 = [-2:0.01:20];

l2 = [-2:0.01:45];

[ko, kp] = meshgrid(l1, l2);

% Проверка устойчивости узлов сетки на достаточные условия устойчивости

c1d = eq1d(kp, ko);

c2d = eq2d(kp, ko);

c3d = eq3d(ko);

% Проверка устойчивости узлов сетки по критерию Рауса

c1r = eq1r(kp);

c2r = eq2r(kp, ko);

c3r = eq3r(ko);

c4r = eq4r();

c5r = eq5r();

c6r = eq6r();

c7r = eq7r(kp, ko);

figure;

grid on

hold on

% Контур для области устойчивости по критерию Рауса

contourf(l1, l2, c1r & c2r & c3r & c4r & c5r & c6r & c7r, [1 1 1 1 1 1 1], FaceAlpha=0.5, FaceColor="red")

% Контур для области устойчивости по достаточным условиям устойчивости

contourf(l1, l2, c1d & c2d & c3d, [1 1 1], FaceAlpha=0.5, FaceColor="blue")

title('Красная область - достаточные условия устойчивости. Синяя область - критерий Рауса')

% Синтезированные коэффициенты

k\_omegay = 2.7468;

k\_psi = 9.0425;

pnt1 = scatter(k\_omegay, k\_psi,'r','filled');

% Подобранные коэффициенты

k\_omegay = 2.8172;

k\_psi = 9.0425;

pnt2 = scatter(k\_omegay, k\_psi, 'y','filled');

legend([pnt1, pnt2],"Синтезированные коэффициенты", "Подобранные коэффициенты")

hold off

xlabel("Komy")

ylabel("Kpsi")

grid on

Программа 8. Программа для областей устойчивости для САУ угла рыскания при плоском развороте для задания на зачёт

# **Приложение 3**

clc

clear

syms s k\_psi\_e k\_omx k\_sigm

% Зададим параметры в системе

H = 5;

M = 0.6;

V = 192;

Zb = -0.2;

sin = 0.08;

cos = 1;

g\_del\_V = 0.051;

M\_x\_b = -5.8;

M\_x\_omx = -1;

M\_x\_omy = -0.2;

M\_x\_sigme = -7;

M\_y\_b = -3;

M\_y\_omx = -0.05;

M\_y\_omy = -0.2;

M\_y\_sigmn = -2.5;

om\_pr = 20;

kzi\_pr = 0.707;

% Передаточная функция привода

W\_pr = om\_pr^2 / (s^2 + 2\*om\_pr\*kzi\_pr\*s + om\_pr^2);

% Передаточная функция ОУ

W1 = M\_x\_sigme / (s - M\_x\_omx);

% Передаточная функция ОУ и привода вместе

W2 = collect(W\_pr\*W1);

% Передаточная функция с обратной связью

W3 = collect(W2 / (1 - W2\*k\_omx));

% Умножаем на интегратор

W4 = collect(W3 / s);

% Ещё раз оборачиваем обратной связью

W5 = collect(W4 / (1 - W4\*k\_sigm));

% ПФ разомкнутой цепи

W6 = collect(k\_psi\_e \* W5 \* g\_del\_V / s);

% ПФ всей системы

W7 = collect(W6 / (1 - W6));

% Выделим числитель и знаменатель итоговой ПФ

[num, den] = numden(W7);

num = collect(num);

den = collect(den);

% Выделим коэффициенты числителя и знаменателя

% При этом сами коэффициенты будут записаны в порядке от младшего к

% старшему, то есть перевернутся

koefs\_num = coeffs(num, 's');

koefs\_den = coeffs(den, 's');

% Выделим старший коэффициент знаменателя. В списке он будет последний

starsh\_koef = koefs\_den(length(koefs\_den));

% Поделим все коэффициенты на это число

koefs\_num = vpa(koefs\_num / starsh\_koef);

koefs\_den = vpa(koefs\_den / starsh\_koef);

% Развернём оба списка с коэффициентами, чтобы они шли от старшего ко

% младшему

koefs\_num = rot90(rot90(koefs\_num));

koefs\_den = rot90(rot90(koefs\_den));

% Преобразованные коэффициенты передаточной функции

B0 = koefs\_num(1);

A5 = koefs\_den(1);

A4 = koefs\_den(2);

A3 = koefs\_den(3);

A2 = koefs\_den(4);

A1 = koefs\_den(5);

A0 = koefs\_den(6);

% Передаточная функция в нормальном виде (старший коэффициент знаменателя

% равен 1

num = B0;

den = (A5\*s^5 + A4\*s^4 + A3\*s^3 + A2\*s^2 + A1\*s + A0);

PF = (num / den);

disp('Числитель ПФ')

pretty(num)

disp('Знаменатель ПФ')

pretty(den)

% Зададим параметры для оптимизации в fmicon

k0 = [1;1;1];

A = [];

b = [];

Aeg = [];

beg = [];

lb = 0.1\*ones(2,1);

ub = 300\*ones(2,1);

% Осуществим параметрическую оптимизацию

[x, fval] = fmincon('fun\_min', k0, A, b, Aeg, beg, lb, ub, 'nonclon');

% Полученные коэффициенты

k\_omx = x(1)

k\_sigm = x(2)

k\_psi\_e = x(3)

% Преобразовывает коэффициенты характеристического полинома,

% подставляя в них найденные значения

B\_0 = vpa(-142.8\*x(3));

A\_5 = vpa(1);

A\_4 = vpa(29.28);

A\_3 = vpa(428.28);

A\_2 = vpa(400.0 + 2800.0\*x(1));

A\_1 = vpa(2800.0\*x(2));

A\_0 = vpa(142.8\*x(3));

num = B\_0;

den = A\_5\*s^5 + A\_4\*s^4 + A\_3\*s^3 + A\_2\*s^2 + A\_1\*s + A\_0;

num = sym2poly(num);

den = sym2poly(den);

% Получаем передаточную функцию и строим переходный процесс

disp('Передаточная функция с полученными коэффициентами')

W\_sys = tf(num, den)

figure;

step(-1\*W\_sys)

grid on

Программа 9. Основной код для системы управления углом рыскания в режиме координированного разворота

function f = fun\_min(x)

% x(1) = k\_omegay, x(2) = k\_psi

A\_1 = 2800\*x(2);

A\_0 = 142.8\*x(3);

f = A\_1 \* A\_0^(-1);

end

Программа 10. Целевая функция для системы управления углом рыскания в режиме координированного разворота

function [c, seq] = nonclon(x)

% x(1) = k\_omegay, x(2) = k\_psi

A\_5 = 1;

A\_4 = 29.28;

A\_3 = 428.28;

A\_2 = 400.0 + 2800.0\*x(1);

A\_1 = 2800\*x(2);

A\_0 = 142.8\*x(3);

lambd\_star = 2.15; % Желаемое значение лямбды

lambd1 = lambd\_star - (A\_0^(-1) \* A\_1 \* A\_2 \* A\_3^(-1));

lambd2 = lambd\_star - (A\_1^(-1) \* A\_2 \* A\_3 \* A\_4^(-1));

lambd3 = lambd\_star - (A\_2^(-1) \* A\_3 \* A\_4 \* A\_5^(-1));

sigm\_star = 2.1; % Желаемое значение сигмы

sigm1 = sigm\_star - (A\_0^(-1) \* A\_1^2 \* A\_2^(-1));

sigm2 = sigm\_star - (A\_1^(-1) \* A\_2^2 \* A\_3^(-1));

sigm3 = sigm\_star - (A\_2^(-1) \* A\_3^2 \* A\_4^(-1));

c = [lambd1; lambd2; lambd3; sigm1; sigm2; sigm3];

seq = [];

end

Программа 11. Ограничения для системы управления углом рыскания в режиме координированного разворота

clc

clear

syms k\_psi\_e k\_omx k\_sigm s

% Зададим параметры в системе

H = 5;

M = 0.6;

V = 192;

Zb = 0.2;

sin = 0.08;

cos = 1;

g\_del\_V = 0.051;

M\_x\_b = -5.8;

M\_x\_omx = -1;

M\_x\_omy = -0.2;

M\_x\_sigme = -7;

M\_y\_b = -3;

M\_y\_omx = -0.05;

M\_y\_omy = -0.2;

M\_y\_sigmn = -2.5;

om\_pr = 20;

kzi\_pr = 0.707;

% Передаточная функция привода

W\_pr = om\_pr^2 / (s^2 + 2\*om\_pr\*kzi\_pr\*s + om\_pr^2);

% Передаточная функция ОУ

W1 = M\_x\_sigme / (s - M\_x\_omx);

% Передаточная функция ОУ и привода вместе

W2 = collect(W\_pr\*W1);

% Передаточная функция с обратной связью

W3 = collect(W2 / (1 - W2\*k\_omx));

% Умножаем на интегратор

W4 = collect(W3 / s);

% Ещё раз оборачиваем обратной связью

W5 = collect(W4 / (1 - W4\*k\_sigm));

% ПФ разомкнутой цепи

W6 = collect(k\_psi\_e \* W5 \* g\_del\_V / s);

% ПФ всей системы

W7 = collect(W6 / (1 - W6));

% Выделим числитель и знаменатель итоговой ПФ

[num, den] = numden(W7);

num = collect(num);

den = collect(den);

% Выделим коэффициенты числителя и знаменателя

% При этом сами коэффициенты будут записаны в порядке от младшего к

% старшему, то есть перевернутся

koefs\_num = coeffs(num, 's');

koefs\_den = coeffs(den, 's');

% Выделим старший коэффициент знаменателя. В списке он будет последний

starsh\_koef = koefs\_den(length(koefs\_den));

% Поделим все коэффициенты на это число

koefs\_num = vpa(koefs\_num / starsh\_koef);

koefs\_den = vpa(koefs\_den / starsh\_koef);

% Развернём оба списка с коэффициентами, чтобы они шли от старшего ко

% младшему

koefs\_num = rot90(rot90(koefs\_num));

koefs\_den = rot90(rot90(koefs\_den));

% Преобразованные коэффициенты передаточной функции

B0 = koefs\_num(1);

A5 = koefs\_den(1);

A4 = koefs\_den(2);

A3 = koefs\_den(3);

A2 = koefs\_den(4);

A1 = koefs\_den(5);

A0 = koefs\_den(6);

% Передаточная функция в нормальном виде (старший коэффициент знаменателя

% равен 1

num = B0;

den = (A5\*s^5 + A4\*s^4 + A3\*s^3 + A2\*s^2 + A1\*s + A0);

PF = (num / den);

disp('Числитель ПФ')

pretty(num)

disp('Знаменатель ПФ')

pretty(den)

% Запишем неравенства ждя критерия Рауса

eq1 = A0 > 0;

eq2 = A1 > 0;

eq3 = A2 > 0;

eq4 = A3 > 0;

eq5 = A4 > 0;

eq6 = A5 > 0;

eq7 = simplify((A1\*A2-A0\*A3)\*(A3\*A4-A2\*A5) - (A1\*A4-A0\*A5)^2) > 0;

% Преобразуем неравенства из символьного типа данных в функции

eq1 = matlabFunction(eq1);

eq2 = matlabFunction(eq2);

eq3 = matlabFunction(eq3);

eq4 = matlabFunction(eq4);

eq5 = matlabFunction(eq5);

eq6 = matlabFunction(eq6);

eq7 = matlabFunction(eq7);

% Задание сетки

ko = [-1:0.05:4];

ks = [-1:0.05:20];

kp = [-1:0.5:300];

% Построение трехмерного графика

% Задаём k\_omega

% Строим плоскую область для параметров k\_sigm, k\_psi

% Прибавляет к параметру k\_omega шаг 0.05

% Повторяем действия

% Таким образом - получится много плоских рисунков,

% которые в сумме будут давать пространство устойчивости

hold on

grid on

view(3)

for i=1:length(ko)

X = [];

Y = [];

Z = [];

ind = 1;

for j=1:length(ks)

for k=1:length(kp)

c1 = eq1(kp(k));

c2 = eq2(ks(j));

c3 = eq3(ko(i));

c4 = eq4();

c5 = eq5();

c6 = eq6();

c7 = eq7(ko(i), kp(k), ks(j));

if c1 & c2 & c3 & c4 & c5 & c6 & c7

X(ind) = ko(i);

Y(ind) = ks(j);

Z(ind) = kp(k);

ind = ind + 1;

end

end

end

plot3(X, Y, Z)

end

xlabel('Komx')

ylabel('Ksigm')

zlabel('Kpsie')

Программа 12. Построение области устойчивости для системы управления углом рыскания в режиме координированного разворота

clc

clear

syms k\_psi\_e k\_omx k\_sigm s

% Зададим параметры в системе

H = 5;

M = 0.6;

V = 192;

Zb = 0.2;

sin = 0.08;

cos = 1;

g\_del\_V = 0.051;

M\_x\_b = -5.8;

M\_x\_omx = -1;

M\_x\_omy = -0.2;

M\_x\_sigme = -7;

M\_y\_b = -3;

M\_y\_omx = -0.05;

M\_y\_omy = -0.2;

M\_y\_sigmn = -2.5;

om\_pr = 20;

kzi\_pr = 0.707;

% Передаточная функция привода

W\_pr = om\_pr^2 / (s^2 + 2\*om\_pr\*kzi\_pr\*s + om\_pr^2);

% Передаточная функция ОУ

W1 = M\_x\_sigme / (s - M\_x\_omx);

% Передаточная функция ОУ и привода вместе

W2 = collect(W\_pr\*W1);

% Передаточная функция с обратной связью

W3 = collect(W2 / (1 - W2\*k\_omx));

% Умножаем на интегратор

W4 = collect(W3 / s);

% Ещё раз оборачиваем обратной связью

W5 = collect(W4 / (1 - W4\*k\_sigm));

% ПФ разомкнутой цепи

W6 = collect(k\_psi\_e \* W5 \* g\_del\_V / s);

% ПФ всей системы

W7 = collect(W6 / (1 - W6));

% Выделим числитель и знаменатель итоговой ПФ

[num, den] = numden(W7);

num = collect(num);

den = collect(den);

% Выделим коэффициенты числителя и знаменателя

% При этом сами коэффициенты будут записаны в порядке от младшего к

% старшему, то есть перевернутся

koefs\_num = coeffs(num, 's');

koefs\_den = coeffs(den, 's');

% Выделим старший коэффициент знаменателя. В списке он будет последний

starsh\_koef = koefs\_den(length(koefs\_den));

% Поделим все коэффициенты на это число

koefs\_num = vpa(koefs\_num / starsh\_koef);

koefs\_den = vpa(koefs\_den / starsh\_koef);

% Развернём оба списка с коэффициентами, чтобы они шли от старшего ко

% младшему

koefs\_num = rot90(rot90(koefs\_num));

koefs\_den = rot90(rot90(koefs\_den));

% Преобразованные коэффициенты передаточной функции

B0 = koefs\_num(1);

A5 = koefs\_den(1);

A4 = koefs\_den(2);

A3 = koefs\_den(3);

A2 = koefs\_den(4);

A1 = koefs\_den(5);

A0 = koefs\_den(6);

% Передаточная функция в нормальном виде (старший коэффициент знаменателя

% равен 1

num = B0;

den = (A5\*s^5 + A4\*s^4 + A3\*s^3 + A2\*s^2 + A1\*s + A0);

PF = (num / den);

disp('Числитель ПФ')

pretty(num)

disp('Знаменатель ПФ')

pretty(den)

% Запишем неравенства для достаточного условия устойчивости

lambd\_star = 2.15; % Желаемое значение лямбды

eq1d = A0^(-1) \* A1 \* A2 \* A3^(-1) >= lambd\_star;

eq2d = A1^(-1) \* A2 \* A3 \* A4^(-1) >= lambd\_star;

eq3d = A2^(-1) \* A3 \* A4 \* A5^(-1) >= lambd\_star;

% Преобразуем неравенства из символьного типа данных в функции

eq1d = matlabFunction(eq1d);

eq2d = matlabFunction(eq2d);

eq3d = matlabFunction(eq3d);

% Задание сетки

ko = [-1:0.05:8];

ks = [-1:0.05:20];

kp = [0.1:0.5:300];

disp('Строим график по достаточным условиям устойчивости')

% Построение трехмерного графика

% Задаём k\_omega

% Строим плоскую область для параметров k\_omega, k\_sigma

% Прибавляет к параметру k\_psi шаг

% Повторяем действия

% Таким образом - получится много плоских рисунков,

% которые в сумме будут давать пространство устойчивости

figure;

hold on

grid on

view(3)

colors = {'red', 'yellow'};

tic % Начинаем считать время

for i=1:length(kp)

X = [];

Y = [];

Z = [];

ind = 1;

for j=1:length(ko)

for k=1:length(ks)

c1 = eq1d(ko(j), kp(i), ks(k));

c2 = eq2d(ko(j), ks(k));

c3 = eq3d(ko(j));

if c1 & c2 & c3

X(ind) = ko(j);

Y(ind) = ks(k);

Z(ind) = kp(i);

ind = ind + 1;

end

end

end

col = colors(randi([1, length(colors)]));

col = string(col);

plot3(X, Y, Z, Color=col)

end

toc % Заканчиваем считать время

% Запишем неравенства ждя критерия Рауса

eq1r = A0 > 0;

eq2r = A1 > 0;

eq3r = A2 > 0;

eq4r = A3 > 0;

eq5r = A4 > 0;

eq6r = A5 > 0;

eq7r = simplify((A1\*A2-A0\*A3)\*(A3\*A4-A2\*A5) - (A1\*A4-A0\*A5)^2) > 0;

% Преобразуем неравенства из символьного типа данных в функции

eq1r = matlabFunction(eq1r);

eq2r = matlabFunction(eq2r);

eq3r = matlabFunction(eq3r);

eq4r = matlabFunction(eq4r);

eq5r = matlabFunction(eq5r);

eq6r = matlabFunction(eq6r);

eq7r = matlabFunction(eq7r);

% Задание сетки

ko = [-1:0.1:4];

ks = [-1:0.1:20];

kp = [0.1:30:300];

% Построение трехмерного графика

% Задаём k\_omega

% Строим плоскую область для параметров k\_sigm, k\_psi

% Прибавляет к параметру k\_omega шаг 0.05

% Повторяем действия

% Таким образом - получится много плоских рисунков,

% которые в сумме будут давать пространство устойчивости

hold on

grid on

view(3)

colors = {'green', 'blue'};

disp('Строим график по критериям Рауса')

tic % Начинаем считать время

for i=1:length(kp)

X = [];

Y = [];

Z = [];

ind = 1;

for j=1:length(ko)

for k=1:length(ks)

c1 = eq1r(kp(i));

c2 = eq2r(ks(k));

c3 = eq3r(ko(j));

c4 = eq4r();

c5 = eq5r();

c6 = eq6r();

c7 = eq7r(ko(j), kp(i), ks(k));

if c1 & c2 & c3 & c4 & c5 & c6 & c7

X(ind) = ko(j);

Y(ind) = ks(k);

Z(ind) = kp(i);

ind = ind + 1;

end

end

end

col = colors(randi([1, length(colors)]));

col = string(col);

plot3(X, Y, Z, Color=col)

end

toc % Заканчиваем считать время

xlabel('Komx')

ylabel('Ksigm')

zlabel('Kpsie')

hold off

clc

clear

syms k\_psi\_e k\_omx k\_sigm s

% Зададим параметры в системе

H = 5;

M = 0.6;

V = 192;

Zb = 0.2;

sin = 0.08;

cos = 1;

g\_del\_V = 0.051;

M\_x\_b = -5.8;

M\_x\_omx = -1;

M\_x\_omy = -0.2;

M\_x\_sigme = -7;

M\_y\_b = -3;

M\_y\_omx = -0.05;

M\_y\_omy = -0.2;

M\_y\_sigmn = -2.5;

om\_pr = 20;

kzi\_pr = 0.707;

% Передаточная функция привода

W\_pr = om\_pr^2 / (s^2 + 2\*om\_pr\*kzi\_pr\*s + om\_pr^2);

% Передаточная функция ОУ

W1 = M\_x\_sigme / (s - M\_x\_omx);

% Передаточная функция ОУ и привода вместе

W2 = collect(W\_pr\*W1);

% Передаточная функция с обратной связью

W3 = collect(W2 / (1 - W2\*k\_omx));

% Умножаем на интегратор

W4 = collect(W3 / s);

% Ещё раз оборачиваем обратной связью

W5 = collect(W4 / (1 - W4\*k\_sigm));

% ПФ разомкнутой цепи

W6 = collect(k\_psi\_e \* W5 \* g\_del\_V / s);

% ПФ всей системы

W7 = collect(W6 / (1 - W6));

% Выделим числитель и знаменатель итоговой ПФ

[num, den] = numden(W7);

num = collect(num);

den = collect(den);

% Выделим коэффициенты числителя и знаменателя

% При этом сами коэффициенты будут записаны в порядке от младшего к

% старшему, то есть перевернутся

koefs\_num = coeffs(num, 's');

koefs\_den = coeffs(den, 's');

% Выделим старший коэффициент знаменателя. В списке он будет последний

starsh\_koef = koefs\_den(length(koefs\_den));

% Поделим все коэффициенты на это число

koefs\_num = vpa(koefs\_num / starsh\_koef);

koefs\_den = vpa(koefs\_den / starsh\_koef);

% Развернём оба списка с коэффициентами, чтобы они шли от старшего ко

% младшему

koefs\_num = rot90(rot90(koefs\_num));

koefs\_den = rot90(rot90(koefs\_den));

% Преобразованные коэффициенты передаточной функции

B0 = koefs\_num(1);

A5 = koefs\_den(1);

A4 = koefs\_den(2);

A3 = koefs\_den(3);

A2 = koefs\_den(4);

A1 = koefs\_den(5);

A0 = koefs\_den(6);

% Передаточная функция в нормальном виде (старший коэффициент знаменателя

% равен 1

num = B0;

den = (A5\*s^5 + A4\*s^4 + A3\*s^3 + A2\*s^2 + A1\*s + A0);

PF = (num / den);

disp('Числитель ПФ')

pretty(num)

disp('Знаменатель ПФ')

pretty(den)

% Запишем неравенства для достаточного условия устойчивости

lambd\_star = 2.15; % Желаемое значение лямбды

eq1d = A0^(-1) \* A1 \* A2 \* A3^(-1) >= lambd\_star;

eq2d = A1^(-1) \* A2 \* A3 \* A4^(-1) >= lambd\_star;

eq3d = A2^(-1) \* A3 \* A4 \* A5^(-1) >= lambd\_star;

% Преобразуем неравенства из символьного типа данных в функции

eq1d = matlabFunction(eq1d);

eq2d = matlabFunction(eq2d);

eq3d = matlabFunction(eq3d);

% Задание сетки

ko = [-1:0.01:4];

ks = [-1:0.01:15];

kp = [30:50:300];

[KO, KS] = meshgrid(ko, ks);

figure;

hold on;

grid on;

colors = {'green', 'blue', 'yellow', 'blue', "magenta", 'red', 'black'};

ind\_col = 1;

for i=1:length(kp)

eq1 = eq1d(k\_omx, kp(i), k\_sigm);

eq2 = eq2d(k\_omx, k\_sigm);

eq3 = eq3d(k\_omx);

eq1 = matlabFunction(eq1);

eq2 = matlabFunction(eq2);

eq3 = matlabFunction(eq3);

c1 = eq1(KO, KS);

c2 = eq2(KO, KS);

c3 = eq3(KO);

col = colors(ind\_col);

col = string(col);

ind\_col = ind\_col + 1;

contourf(ko, ks, c1 & c2 & c3, [1 1 1], FaceAlpha=0.2, FaceColor=col)

colormap lines

end

hold off

xlabel('Komx')

ylabel('Ksigm')

title('Область устойчивости по достаточным условиям для kpsie=30, 80, 130, 180, 230, 280')

% Запишем неравенства для достаточного условия устойчивости

eq1r = A0 > 0;

eq2r = A1 > 0;

eq3r = A2 > 0;

eq4r = A3 > 0;

eq5r = A4 > 0;

eq6r = A5 > 0;

eq7r = simplify((A1\*A2-A0\*A3)\*(A3\*A4-A2\*A5) - (A1\*A4-A0\*A5)^2) > 0;

% Преобразуем неравенства из символьного типа данных в функции

eq1r = matlabFunction(eq1r);

eq2r = matlabFunction(eq2r);

eq3r = matlabFunction(eq3r);

eq4r = matlabFunction(eq4r);

eq5r = matlabFunction(eq5r);

eq6r = matlabFunction(eq6r);

eq7r = matlabFunction(eq7r);

% Задание сетки

ko = [-1:0.01:4];

ks = [-1:0.01:18];

kp = [30:50:300];

[KO, KS] = meshgrid(ko, ks);

figure;

hold on;

grid on;

colors = {'green', 'blue', 'yellow', "magenta", 'red', 'black'};

ind\_col = 1;

for i=1:length(kp)

eq2 = eq2r(k\_sigm);

eq3 = eq3r(k\_omx);

eq7 = eq7r(k\_omx, kp(i), k\_sigm);

eq2 = matlabFunction(eq2);

eq3 = matlabFunction(eq3);

eq7 = matlabFunction(eq7);

c1 = eq2(KS);

c2 = eq3(KO);

c3 = eq7(KO, KS);

col = colors(ind\_col);

col = string(col);

ind\_col = ind\_col + 1;

contourf(ko, ks, c1 & c2 & c3, [1 1 1], FaceAlpha=0.2, FaceColor=col)

end

hold off

xlabel('Komx')

ylabel('Ksigm')

title('Область устойчивости по критерию Рауса для kpsie=30, 80, 130, 180, 230, 280')

% Рабочая точка

komx = 0.9225;

ksigm = 3.5337;

kpsie = 109.4343;

figure;

grid on;

hold on;

ko = [-1:0.01:4];

ks = [-1:0.01:15];

[KO, KS] = meshgrid(ko, ks);

eq1 = eq1d(k\_omx, kpsie, k\_sigm);

eq2 = eq2d(k\_omx, k\_sigm);

eq3 = eq3d(k\_omx);

eq1 = matlabFunction(eq1);

eq2 = matlabFunction(eq2);

eq3 = matlabFunction(eq3);

c1 = eq1(KO, KS);

c2 = eq2(KO, KS);

c3 = eq3(KO);

contourf(ko, ks, c1 & c2 & c3, [1 1 1], FaceAlpha=0.2, FaceColor='red')

ko = [-1:0.01:4];

ks = [-1:0.01:18];

kp = [30:50:300];

[KO, KS] = meshgrid(ko, ks);

eq2 = eq2r(k\_sigm);

eq3 = eq3r(k\_omx);

eq7 = eq7r(k\_omx, kpsie, k\_sigm);

eq2 = matlabFunction(eq2);

eq3 = matlabFunction(eq3);

eq7 = matlabFunction(eq7);

c1 = eq2(KS);

c2 = eq3(KO);

c3 = eq7(KO, KS);

contourf(ko, ks, c1 & c2 & c3, [1 1 1], FaceAlpha=0.2, FaceColor='green')

scatter(komx, ksigm, 'filled')

hold off

xlabel('Komx')

ylabel('Ksigm')

title('Область устойчивости по двум критериям для kpsie=109.4343')

Программа 13. Программа для областей устойчивости для САУ угла рыскания при координированном развороте для задания на зачёт