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