```
function smolkina_hw2()
% Матрицы системы
A = [0              -1.53921/41.1368    -1.53921/41.1368*0.62;
     41.1368*10^-6      0                   0;
     0              -1.5*10^-3          -1.5*10^-3];

B = [1;
     0;
     0];
% матрица коэф лкр регулятора
Q = eye(3,3); % положительно полуопределенная весова матрица
% Q(1, 1) = 5000;
% Q(3, 3) = 100;
R = 1;

P = care(A, B, Q, R); % Решение Риккати
K_LQR = R^-1 * B' * P ;
% Матрицы эталонной системы
A_m = A - B * K_LQR;
B_m = B;

T = 20;
tspan = 0:0.1:T;
% Вектор начальных условий для расширенной системы
% sys_ICs = [0.3, 0.1, 0.1]; % Я пробовала несколько разных
sys_ICs = [1, 1, -1];
x_m_ICs = sys_ICs;
Theta_ICs = [0,0];
ICs = [sys_ICs x_m_ICs Theta_ICs];

% g_ref = @(t) 0;
g_ref = @(t)sin(20 * t);
% g_ref = @(t)sin(20 * t) * cos(40 * t);

% Траектория эталонной системы с референсом
sys_m = ss(A_m, B_m, eye(size(A, 1), size(A, 1)), zeros(size(B, 1), size(B, 2)));
% 3 3 3 1

% Значение задающего воздействия
g_ref_vals = zeros(size(tspan)); % 1 51
for i = 1:size(tspan, 2)
 g_ref_vals(i) = g_ref(i);
end

% вычисляем переходные процессы системы
x_m = lsim(sys_m, g_ref_vals, tspan, x_m_ICs);

figure('Name', 'Reference model trajectories'); % открываем новое окно
plot(tspan, x_m);
legend('u_m', 'w_m', 'q_m', 'theta_m');

% интегрируем систему
[t_plot, x_cl_adept] = ode45(@(t, x) compas_RHS(t, A, B, K_LQR, ...
```

```
    A_m, B_m, x, g_ref, @MIMO_adaptive_control), tspan, ICs);

% MIMO_adaptive_control - адаптивное управление для системы с многими
% входами и выходами
figure('Name', 'Расширенная система')
plot(t_plot, x_cl_adept);
legend('theta_{dot}', 'theta', 'x_{dot}', 'x', ...
    'theta\_m_{dot}', 'theta\_m', 'x\_m_{dot}', 'x\_m', ...
    'Theta_1', 'Theta_2')

figure('Name', 'Errors')
plot(t_plot, x_cl_adept(:, 1:4) - x_cl_adept(:, 5:8));
legend('e_1', 'e_2', 'e_3', 'e_4')


[t_plot, x_cl_adept] = ode45(@(t, x) compas_RHS(t, A, B, K_LQR, ...
    A_m, B_m, x, g_ref, @LQR_control), tspan, ICs);

% MIMO_adaptive_control - адаптивное управление для системы с многими
% входами и выходами
figure('Name', 'Расширенная система LQR')
plot(t_plot, x_cl_adept); % Просмотр динамики вектора состояния
legend('theta_{dot}', 'theta', 'x_{dot}', 'x', ...
    'theta\_m_{dot}', 'theta\_m', 'x\_m_{dot}', 'x\_m', ...
    'Theta_1', 'Theta_2')

figure('Name', 'Errors LQR')
plot(t_plot, x_cl_adept(:, 1:4) - x_cl_adept(:, 5:8));
legend('e_1', 'e_2', 'e_3', 'e_4')
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% вспомогательные функции
function dxdt = compas_RHS(t, A, B, K_lqr, ...
    A_m, B_m, x, g_ref, u_ref)
    dxdt = zeros(size(x, 1), size(x, 2));
    cur_g_ref = g_ref(t);

    % Размерность системы
    n = 3; % состояние
    r = 1;% управление
    l = 2; % неопределенные параметры
    % Индексы элементов для подсистемы
    sys_x_ind = 1:n;
    x_m_ind = (n+1) : 2*n;
    hat_Theta_ind = (2*n+1) : (2*n + l*r);

   sys_x = x(sys_x_ind);% вектор состояния системы
   x_m = x(x_m_ind); % выделяем вектор ЭМ
   % Выделение вектора элементов матрицы тхета, транспонируем,
   % восстанавливаем матрицу
   hat_Theta = reshape(x(hat_Theta_ind), r, l)';
```

```
    % Задаем неопределенное действие w
    % w(x) = [-k1; -k2]' * [x1; x3]
    k1 = 0.01; k2 = 0.01; % неизвестные коэффициеннты
    Theta = [-k1; -k2];   %; -k3];
    Phi = [sys_x(1); sys_x(3)];
    w = Theta'*Phi;

    if isequal(u_ref, @MIMO_adaptive_control)
        u = MIMO_adaptive_control(K_lqr, sys_x, r, hat_Theta, Phi, cur_g_ref);
    else
        u = LQR_control(K_lqr, sys_x);
    end

    % Правая часть для расширенного вектора
    dxdt(sys_x_ind) = A*sys_x + B*(u + w); % для замкнутой системы
    % Правая часть для ЭМ
    dxdt(x_m_ind) = A_m*x_m + B_m*cur_g_ref;

    e = sys_x - x_m;
    d_hat_Theta = get_d_hat_Theta(n, l, A_m, B, Phi, e);
    dxdt(hat_Theta_ind) = reshape(d_hat_Theta', [], 1);
end
% функция лкр регулятора
function u = LQR_control(K_lqr, sys_x)
    u = -K_lqr*sys_x;
end
% функция адаптивного управлени
function u = MIMO_adaptive_control(K_lqr, sys_x, r, hat_Theta, Phi, cur_g_ref)
    K_x = -K_lqr;
    K_g = eye(r);
    u = K_x*sys_x + K_g*cur_g_ref - hat_Theta'*Phi;
end
% функкция адаптации
function d_hat_Theta = get_d_hat_Theta(n, l, A_m, B, Phi, e)
    G = eye(l);
    Q = eye(n);
    P = lyap(A_m, Q);

    d_hat_Theta = G*Phi*e'*P*B;
end
```