```
syms x1 x2 x3 x4 x5 x6 u
syms M m1 m2 l1 l2 g
M = 1.5;
m1 = 0.5;
l1 = 0.5;
m2 = 0.75;
l2 = 0.75;
g = 9.81;
T_cart = 0.5*M*x4^2;
T_pendulum1 = 0.5*m1*((x4+l1*x5*cos(x2))^2 + (l1*x5*sin(x2))^2);
T_pendulum2 = 0.5*m2*((x4 + l1*x5*cos(x2) + l2*x6*cos(x3))^2 + (l1*x5*sin(x2) +
l2*x6*sin(x3))^2);
T_total = simplify(T_cart + T_pendulum1 + T_pendulum2);
V_Cart = 0;
V_pendulum1 = m1*g*l1*cos(x2);
V_pendulum2 = m2*g*(l1*cos(x2)+l2*cos(x3));
V_total = simplify(V_Cart + V_pendulum1 + V_pendulum2);
L = simplify(T_total - V_total);

dL_dx1 = diff(L, x1);
dL_dx2 = diff(L, x2);
dL_dx3 = diff(L, x3);
dL_dx4 = diff(L, x4);
dL_dx5 = diff(L, x5);
dL_dx6 = diff(L, x6);
q = [x1 ;x2; x3];
qd = [x4; x5; x6];
syms qdd1 qdd2 qdd3 real
qdd = [qdd1; qdd2; qdd3];
Jaco = jacobian([dL_dx4;dL_dx5;dL_dx6],[q;qd]);
d_dt_dLdqdot = Jaco*[qd; qdd];
dLdq = [dL_dx1;dL_dx2;dL_dx3];
Force = [u; 0; 0];
Lagrange_eq = simplify(d_dt_dLdqdot - dLdq -Force);
D = sym(zeros(3,3));
for i=1:3
  for j=1:3
    D(i,j) = simplify(diff(diff(T_total, qd(i)), qd(j)));
  end

end
D
```

```
D =
```

$$\begin{pmatrix} \frac{11}{4} & \frac{5\cos(x_2)}{8} & \frac{9\cos(x_3)}{16} \\ \frac{5\cos(x_2)}{8} & \frac{5}{16} & \frac{9\cos(x_2-x_3)}{32} \\ \frac{9\cos(x_3)}{16} & \frac{9\cos(x_2-x_3)}{32} & \frac{27}{64} \end{pmatrix}$$

```
G = simplify([ diff(V_total, x1);
      diff(V_total, x2);
      diff(V_total, x3) ])
```

G =

$$\begin{pmatrix} 0 \\ -\frac{981\sin(x_2)}{160} \\ -\frac{8829\sin(x_3)}{1600} \end{pmatrix}$$

```
n = 3;
c = sym(zeros(n,n,n));
for i=1:n
  for j=1:n
    for k=1:n
      c(i,j,k) = 1/2*( diff(D(i,j), q(k)) + diff(D(i,k), q(j)) - diff(D(j,k),
q(i)) );
    end
  end
end

C = sym(zeros(n,n));
for i=1:n
  for j=1:n
    C(i,j) = simplify( reshape(c(i,j,:),[n,1]).' * qd );
  end
end

H = [1;0;0];
qdd = simplify(D\(H*u - C*qd-G));
X = [q;qd];
f_sym = [qd; qdd]
```

f_sym =

$$\begin{pmatrix} x_4 \\ x_5 \\ x_6 \\ -\dfrac{280\,u - 981\sin(2\,x_2) + 100\,{x_5}^2\sin(x_2) + 45\,{x_6}^2\sin(x_3) - 120\,u\cos(\\ 20\,\sigma_2 \\ \dfrac{1575\,{x_6}^2\sin(x_2 - x_3) - 30411\sin(x_2) - 8829\sin(x_2 - 2\,x_3) + 250\,{x_5}^2\sin(2\,x_2) + 1400\,u\cos(x_2) + 450\,{x_5}^2\,s}{50\,\sigma_2} \\ -\dfrac{300\sin(x_2 - x_3)\,{x_5}^2 + 135\sin(\sigma_3)\,{x_6}^2 - 2943\,\sigma_1 + 2943\sin(x_3) + 200\,u\cos(2\,.}{75\cos(2\,x_2) + 135\cos(\sigma_3) - 390} \end{pmatrix}$$

where

$$\sigma_1 = \sin(2\,x_2 - x_3)$$

$$\sigma_2 = 5\cos(2\,x_2) + 9\cos(\sigma_3) - 26$$

$$\sigma_3 = 2\,x_2 - 2\,x_3$$

```
D_inv = inv(D)
```

```
D_inv =
```

$$\begin{pmatrix} \dfrac{4\,\left(3\cos(x_2 - x_3)^2 - 5\right)}{\sigma_4} & \sigma_2 & \sigma_3 \\ \sigma_2 & \dfrac{16\,\left(3\cos(x_3)^2 - 11\right)}{\sigma_4} & \sigma_1 \\ \sigma_3 & \sigma_1 & \dfrac{320\,\left(5\cos(x_2)^2 - 11\right)}{27\,\sigma_4} \end{pmatrix}$$

where

$$\sigma_1 = \frac{32\,\left(11\cos(x_2 - x_3) - 5\cos(x_2)\cos(x_3)\right)}{3\,\sigma_4}$$

$$\sigma_2 = \frac{8\,\left(5\cos(x_2) - 3\cos(x_2 - x_3)\cos(x_3)\right)}{\sigma_4}$$

$$\sigma_3 = \frac{80\,\left(\cos(x_3) - \cos(x_2 - x_3)\cos(x_2)\right)}{3\,\sigma_4}$$

$$\sigma_4 = 33\cos(x_2 - x_3)^2 - 30\cos(x_2 - x_3)\cos(x_2)\cos(x_3) + 25\cos(x_2)^2 + 15\cos(x_3)^2 - 55$$

```
C
```

C =

$$\begin{pmatrix} 0 & -\frac{5\,x_5\sin(x_2)}{8} & -\frac{9\,x_6\sin(x_3)}{16} \\ 0 & 0 & \frac{9\,x_6\sin(x_2-x_3)}{32} \\ 0 & -\frac{9\,x_5\sin(x_2-x_3)}{32} & 0 \end{pmatrix}$$

```
syms u_1 u_2
 u_new = [u_1; u_2];
 H_new = [1 0;
          0 1;
          0 0];
 qdd_new_1 = simplify(D\(H_new*u_new - C*qd-G));
 X_new_1 = [q;qd];
 f_sym_new_1 = [qd; qdd_new_1]
```

f_sym_new_1 =

$$\begin{pmatrix} x_4 \\ x_5 \\ x_6 \\ -\frac{280\,u_1 - 981\sin(2\,x_2) + 100\,{x_5}^2\sin(x_2) + 45\,{x_6}^2\sin(x_3) - 120\,u_1\cos(\sigma_4)}{20\,\sigma_3} \\ \frac{1200\,u_2\cos(2\,x_3) - 8829\sin(x_2-2\,x_3) - 30411\sin(x_2) - 7600\,u_2 + 1575\,{x_6}^2\sin(x_2-x_3) + 250\,{x_5}^2\sin(2\,x}{50\,\sigma_3} \\ -\frac{300\sin(x_2-x_3)\,{x_5}^2 + 135\sin(\sigma_4)\,{x_6}^2 - 2943\,\sigma_1 + 2943\sin(x_3) + 200\,u_1\cos(2\,x_2-x_3) + 4(}{75\cos(2\,x_2) + 135\cos(\sigma_4) - 390} \end{pmatrix}$$

where

$\sigma_1 = \sin(2\,x_2 - x_3)$

$\sigma_2 = \cos(x_2 - 2\,x_3)$

$\sigma_3 = 5\cos(2\,x_2) + 9\cos(\sigma_4) - 26$

$\sigma_4 = 2\,x_2 - 2\,x_3$

```
syms u_e1 u_e2 real
q_val_1 = [0.1; deg2rad(60); deg2rad(45)];
qd_val_1 = [0; 0; 0];
G_val = double(subs(G, [x1 x2 x3 x4 x5 x6], [q_val_1' qd_val_1']));
eq = H_new*[u_e1 ; u_e2] == G_val;
sol_u = solve(eq, u_e1, u_e2, 'Real', true)
```

```
sol_u = struct with fields:
    u_e1: [0×1 sym]
    u_e2: [0×1 sym]
```

```
syms u_3 u_e3 real
u_new_2 = [u_1; u_2; u_3];
H_new_2 = [1 0 0;
        0 1 0 ;
        0 0 1];
qdd_new_2 = simplify(D\(H_new_2*u_new_2 - C*qd-G));
X_new_2 = [q;qd];
f_sym_new_2 = [qd; qdd_new_2]
```

f_sym_new_2 =

$$\left(\begin{array}{c} -\frac{840\,u_1 - 2943\sin(2\,x_2) + 300\,x_5^2\sin(x_2) + 135\,x_6^2\sin(x_3) - 36\ldots}{\ldots} \\ \frac{3600\,u_2\cos(2\,x_3) - 26487\sin(x_2 - 2\,x_3) - 91233\sin(x_2) - 22800\,u_2 + 4725\,x_6^2\sin(x_2 - x_3) + 750\,x_5^2\sin(\ldots}{\ldots} \\ -\frac{2700\sin(x_2 - x_3)\,x_5^2 + 1215\sin(\sigma_5)\,x_6^2 + 13600\,u_3 - 26487\,\sigma_1 + 26487\sin(x_3)}{\ldots} \end{array} \right.$$

where

$$\sigma_1 = \sin(2\,x_2 - x_3)$$

$$\sigma_2 = \cos(2\,x_2 - x_3)$$

$$\sigma_3 = \cos(x_2 - 2\,x_3)$$

$$\sigma_4 = 5\cos(2\,x_2) + 9\cos(\sigma_5) - 26$$

$$\sigma_5 = 2\,x_2 - 2\,x_3$$

```
q_val_1 = [0.1; deg2rad(60); deg2rad(45)];
qd_val_1 = [0; 0; 0];
G_val = double(subs(G, [x1 x2 x3 x4 x5 x6], [q_val_1' qd_val_1']));
eq = H_new_2*[u_e1 ; u_e2 ; u_e3] == G_val;
sol_u = solve(eq, u_e1, u_e2, u_e3, 'Real', true);
u_e_new = double(struct2array(sol_u))
```

```
u_e_new = 1×3
         0   -5.3098   -3.9019
```

```
A_sym_new = simplify(jacobian(f_sym_new_2, X));
B_sym_new = simplify(jacobian(f_sym_new_2, u_new_2));
vars = [x1; x2; x3; x4; x5; x6; u_1; u_2; u_3];
vals = [0.1; deg2rad(60); deg2rad(45); 0; 0; 0; u_e_new'];

A_lin_syms_new = double(subs(A_sym_new, vars, vals))
```

```
A_lin_syms_new = 6×6
         0         0         0    1.0000         0         0
         0         0         0         0    1.0000         0
         0         0         0         0         0    1.0000
         0   -0.5341   -1.1264         0         0         0
         0   22.5046  -17.8041         0         0         0
         0  -13.9883   21.7759         0         0         0
```

```
B_lin_syms_new = double(subs(B_sym_new, vars, vals))
```

```
B_lin_syms_new = 6×3
         0         0         0
         0         0         0
         0         0         0
    0.4252   -0.1742   -0.2887
   -0.1742    7.3409   -4.5629
   -0.2887   -4.5629    5.5808
```

```
C_new = [1 0 0 0 0 0;
         0 1 0 0 0 0;
         0 0 1 0 0 0];
```

Problem 1. (4 pts) Discretize the linearized model about the equilibrium pair f
use it as the design model. You as a designer select the sampling period h.

Problem 2. (7 pts) First design an MPC using the augmented model without ar
implement the controller on the nonlinear continuous model. Write a script tha
behavior of the closed-loop system.

$$x[k+1] = \phi x[k] + \Gamma u[k]$$

Augmented Model :

$$x_a[k+1] = \phi_a x_a[k] + \Gamma_a u[k]$$

$$y_a[k] = C_a[k] x_a[k]$$

$$\phi_a = \begin{bmatrix} \phi & 0 \\ C\phi & I_p \end{bmatrix}$$

$$\Gamma_a = \begin{bmatrix} \Gamma \\ C\Gamma \end{bmatrix}$$

$$C_a = [O \quad I_p]$$

I am choosing Np = 20

$$y = \mathrm{W}x[k] + \mathrm{ZU}$$

```
h = 0.005;
C_tracking = eye(6);
D_zero = zeros(6, size(B_lin_syms_new,2));

sysd_plant = c2d(ss(A_lin_syms_new, B_lin_syms_new, C_tracking, D_zero), h, 'zoh');
Ad = sysd_plant.A;
Bd = sysd_plant.B;
Cd = sysd_plant.C;
Dd = sysd_plant.D;

[n, m] = size(Bd);     % n = 6 states, m = 3 inputs
p = size(Cd,1);       % p = 6 outputs (we track full states)
Phi = [Ad, Bd;
       zeros(m,n), eye(m)];
Gamma = [Bd;
         eye(m)];

Ca = [Cd, zeros(p,m);
      zeros(m,n), eye(m)];
Da = zeros(p+m, m);

sys_aug = ss(Phi, Gamma, Ca, Da, h);

Np = 80; Nc = 40;
mpc_obj = mpc(sys_aug, h, Np, Nc);
```

```
-->"Weights.ManipulatedVariables" is empty. Assuming default 0.00000.
-->"Weights.ManipulatedVariablesRate" is empty. Assuming default 0.10000.
-->"Weights.OutputVariables" is empty. Assuming default 1.00000.
   for output(s) y1 y2 y3 and zero weight for output(s) y4 y5 y6 y7 y8 y9
```

```
mpc_obj.Weights.OutputVariables = [2 4 4 0 0 0, 0.01*ones(1,m)];
mpc_obj.Weights.ManipulatedVariables = [0.1 0.1 0.1];
```

```
mpc_obj.Weights.ManipulatedVariablesRate = [0.2 0.2 0.2];

mpc_obj.Model.Nominal.X = [x_eq(:); u_eq(:)]';
mpc_obj.Model.Nominal.Y = (Ca * [x_eq(:); u_eq(:)])';
mpc_obj.Model.Nominal.U = zeros(1,m);
```

```
disp('Starting Augmented MPC (Δu) simulation with 2D Animation...');
```

```
Starting Augmented MPC (Δu) simulation with 2D Animation...
```

```
x_eq
```

```
x_eq = 6×1
    0.1000
    1.0472
    0.7854
         0
         0
         0
```

```
tf = 10;
t_history = 0:h:tf;
Nsim = length(t_history);

% --- Initial Conditions (from augmented model) ---
x0 = [0; deg2rad(35); deg2rad(50); 0; 0; 0];
x = x0;
u_prev = u_eq(:); % Initial absolute MV

% Get dimensions
n = size(x, 1);
m = size(u_prev, 1);

% --- History Initialization (from augmented model) ---
x_history = zeros(n, Nsim);
u_history = zeros(m, Nsim);
du_history = zeros(m, Nsim);
x_aug_history = zeros(n+m, Nsim);

x_history(:,1) = x;
u_history(:,1) = u_prev; % u_history(1) is the initial u_eq
du_history(:,1) = zeros(m,1);
x_aug_history(:,1) = [x; u_prev];

% --- MPC State and Reference (from augmented model) ---
mpc_state = mpcstate(mpc_obj);
```

```
-->Assuming output disturbance added to measured output #2 is integrated white noise.
-->Assuming output disturbance added to measured output #3 is integrated white noise.
   Assuming no disturbance added to measured output #1.
-->Assuming output disturbance added to measured output #7 is integrated white noise.
```

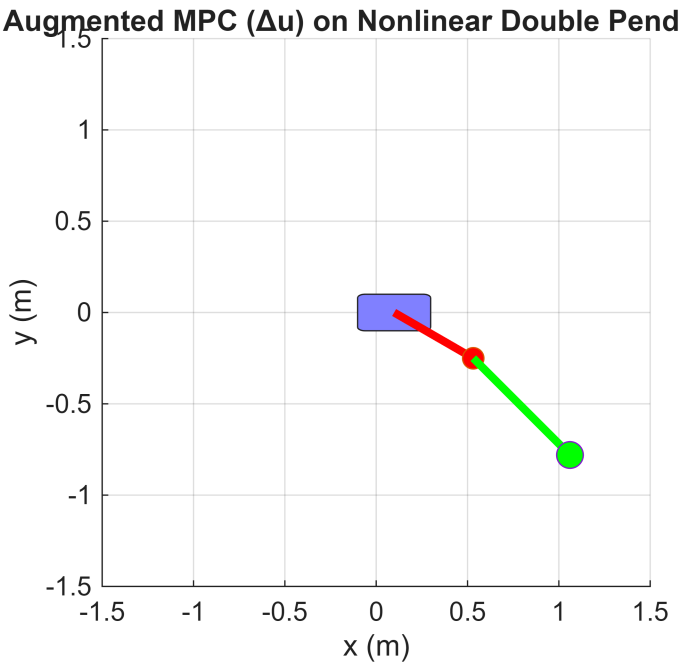
```
   Assuming no disturbance added to measured output #8.
   Assuming no disturbance added to measured output #9.
-->Assuming output disturbance added to measured output #4 is integrated white noise.
-->Assuming output disturbance added to measured output #5 is integrated white noise.
-->Assuming output disturbance added to measured output #6 is integrated white noise.
-->"Model.Noise" is empty. Assuming white noise on each measured output.
```

```
% Define the 9x1 reference vector for the 9 outputs
r_mpc = [Cd*x_eq + Dd*u_eq; u_eq];

% --- 2D Animation Setup (from first script) ---
figure('Name','Cart-Pendula Animation (nonlinear plant, augmented
MPC)','Color','w');
ax = gca; hold(ax,'on'); axis(ax,'equal'); grid(ax,'on');
title(ax,'Augmented MPC (Δu) on Nonlinear Double Pendulum');
xlabel(ax,'x (m)'); ylabel(ax,'y (m)');

cart_w = 0.4; cart_h = 0.2;

p_cart = [x(1), 0];
p_m1   = [x(1) + l1_val*sin(x(2)), -l1_val*cos(x(2))];
p_m2   = [p_m1(1) + l2_val*sin(x(3)), p_m1(2) - l2_val*cos(x(3))];

h_cart = rectangle(ax, 'Position', [p_cart(1)-cart_w/2, -cart_h/2, cart_w, cart_h],
...
                   'FaceColor',[0.5 0.5 1], 'Curvature',[0.2 0.2]);
h_line1 = line(ax, [p_cart(1), p_m1(1)], [p_cart(2), p_m1(2)], 'LineWidth', 3,
'Color', 'r');
h_mass1 = plot(ax, p_m1(1), p_m1(2), 'o', 'MarkerSize', 8, 'MarkerFaceColor','r');
h_line2 = line(ax, [p_m1(1), p_m2(1)], [p_m1(2), p_m2(2)], 'LineWidth', 3, 'Color',
'g');
h_mass2 = plot(ax, p_m2(1), p_m2(2), 'o', 'MarkerSize', 10, 'MarkerFaceColor','g');

if ~isinf(mpc_obj.OV(1).Min)
    xline(ax, mpc_obj.OV(1).Min, '--k', 'Min x');
end
if ~isinf(mpc_obj.OV(1).Max)
    xline(ax, mpc_obj.OV(1).Max, '--k', 'Max x');
end

axis(ax, [-1.5 1.5 -1.5 1.5]); % Adjust as needed
drawnow;
ode_opts = odeset('RelTol', 1e-6, 'AbsTol', 1e-8);
fprintf('Running simulation for %d steps...\n', Nsim);
```

```
Running simulation for 2001 steps...
```

```
for k = 1:Nsim-1
    % 1. Construct 9x1 measurement vector [y; u_prev]
    % (using current state x and previous input u_prev)
    y_plant = Cd * x + Dd * u_prev;
```

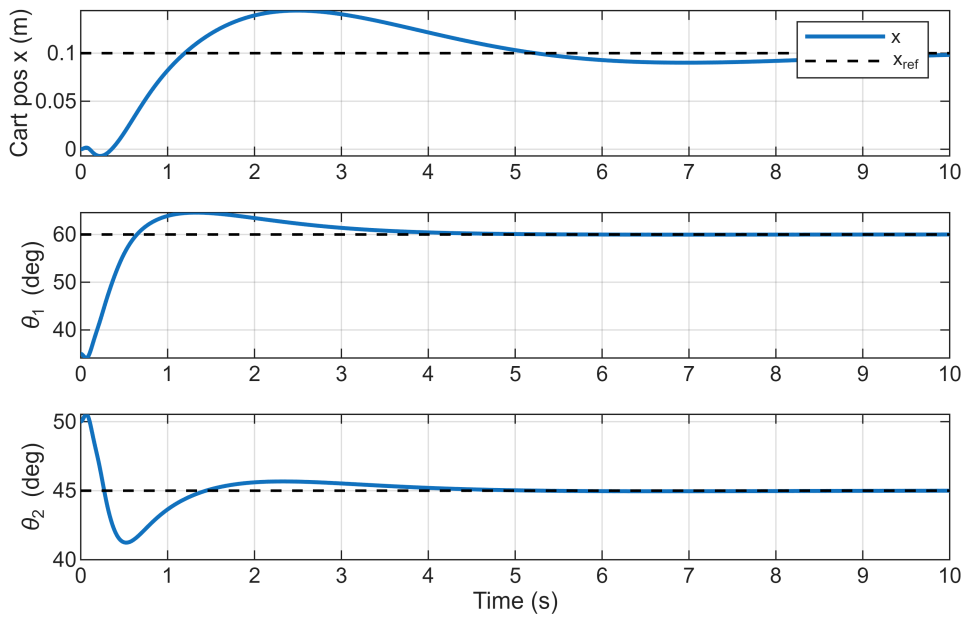
```
    y_mpc = [ y_plant ; u_prev ];

    % 2. Calculate optimal control increment (du)
    try
        du = mpcmove(mpc_obj, mpc_state, y_mpc, r_mpc);
    catch ME
        warning('mpcmove threw an error at step %d: %s', k, ME.message);
        if k==1
            du = zeros(m,1);
        else
            du = du_history(:,k-1); % Reuse last increment
        end
    end
    if isempty(du)
        du = zeros(m,1); % Fallback
    end

    u_applied = u_prev + du;
    u_cmd = u_applied(:);

    % 4. Simulate nonlinear plant one step
    odefun = @(tt,xx) f_num(tt, xx, u_cmd);
    [~, XX] = ode45(odefun, [0 h], x, ode_opts);
    x_next = XX(end, :)';

    x = x_next; % This is x(k+1)

    x_history(:, k+1) = x;
    u_history(:, k) = u_applied; % u(k)
    du_history(:, k) = du;       % du(k)
    x_aug_history(:, k) = [x; u_prev];

    % 6. Update u_prev for next loop
    u_prev = u_applied;
    p_cart = [x(1), 0];
    p_m1   = [x(1) + l1_val*sin(x(2)), -l1_val*cos(x(2))];
    p_m2   = [p_m1(1) + l2_val*sin(x(3)), p_m1(2) - l2_val*cos(x(3))];

    set(h_cart, 'Position', [p_cart(1)-cart_w/2, -cart_h/2, cart_w, cart_h]);
    set(h_line1, 'XData', [p_cart(1), p_m1(1)], 'YData', [p_cart(2), p_m1(2)]);
    set(h_mass1, 'XData', p_m1(1), 'YData', p_m1(2));
    set(h_line2, 'XData', [p_m1(1), p_m2(1)], 'YData', [p_m1(2), p_m2(2)]);
    set(h_mass2, 'XData', p_m2(1), 'YData', p_m2(2));

    drawnow limitrate;
end
```



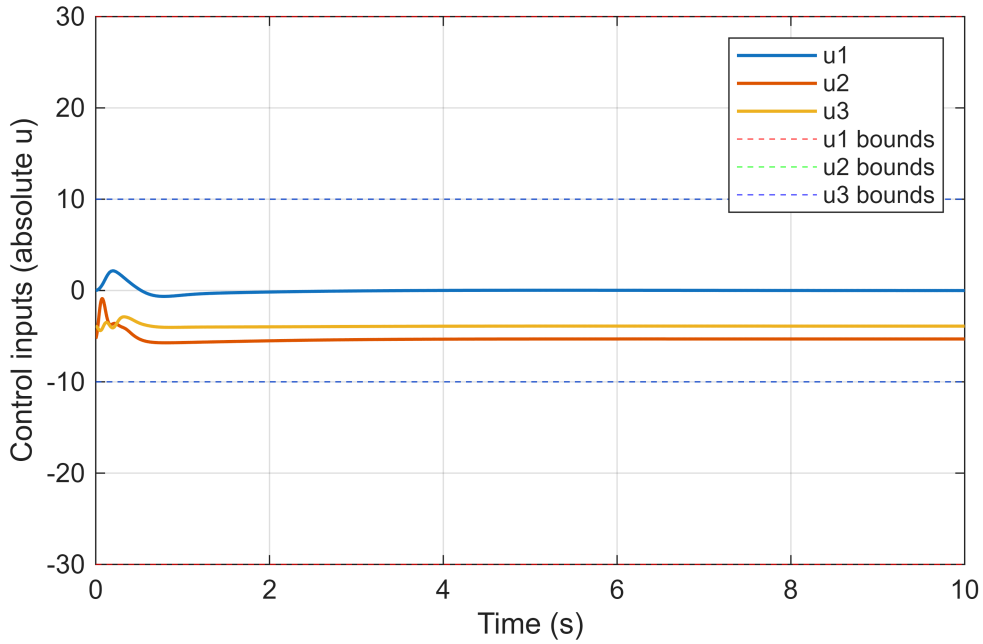

```
u_history(:, Nsim) = u_history(:, Nsim-1);
du_history(:, Nsim) = du_history(:, Nsim-1);
x_aug_history(:, Nsim) = [x_history(:,Nsim); u_history(:,Nsim)];
disp('Simulation complete.');
```

```
Simulation complete.
```

```
figure('Name','State Trajectories (Augmented MPC)','Color','w');
subplot(3,1,1);
plot(t_history, x_history(1,:), 'LineWidth', 1.5); hold on;
if ~isinf(mpc_obj.OV(1).Min), yline(mpc_obj.OV(1).Min,'r--','Min x'); end
if ~isinf(mpc_obj.OV(1).Max), yline(mpc_obj.OV(1).Max,'r--','Max x'); end
plot(t_history, ones(1,Nsim)*x_eq(1), 'k--', 'LineWidth', 1);
ylabel('Cart pos x (m)'); grid on; legend('x','x_{ref}');

subplot(3,1,2);
plot(t_history, rad2deg(x_history(2,:)), 'LineWidth', 1.5); hold on;
if ~isinf(mpc_obj.OV(2).Min), yline(rad2deg(mpc_obj.OV(2).Min),'r--'); end
if ~isinf(mpc_obj.OV(2).Max), yline(rad2deg(mpc_obj.OV(2).Max),'r--'); end
plot(t_history, ones(1,Nsim)*rad2deg(x_eq(2)), 'k--', 'LineWidth', 1);
ylabel('\theta_1 (deg)'); grid on;

subplot(3,1,3);
plot(t_history, rad2deg(x_history(3,:)), 'LineWidth', 1.5); hold on;
if ~isinf(mpc_obj.OV(3).Min), yline(rad2deg(mpc_obj.OV(3).Min),'r--'); end
if ~isinf(mpc_obj.OV(3).Max), yline(rad2deg(mpc_obj.OV(3).Max),'r--'); end
plot(t_history, ones(1,Nsim)*rad2deg(x_eq(3)), 'k--', 'LineWidth', 1);
ylabel('\theta_2 (deg)'); grid on;
xlabel('Time (s)');
```




```
figure('Name','Control Effort (with absolute MV bounds)','Color','w');
plot(t_history, u_history(1,:), 'LineWidth', 1.2); hold on;
plot(t_history, u_history(2,:), 'LineWidth', 1.2);
plot(t_history, u_history(3,:), 'LineWidth', 1.2);
if exist('u_abs_min', 'var') && exist('u_abs_max', 'var')
    yline(u_abs_min(1),'r--'); yline(u_abs_max(1),'r--');
    yline(u_abs_min(2),'g--'); yline(u_abs_max(2),'g--');
    yline(u_abs_min(3),'b--'); yline(u_abs_max(3),'b--');
    legend('u1','u2','u3','u1 bounds','','u2 bounds','','u3 bounds','');
else
    legend('u1','u2','u3');
    warning('u_abs_min or u_abs_max not found. Plotting without bounds.');
end
ylabel('Control inputs (absolute u)'); xlabel('Time (s)'); grid on;
```



Problem 3. (7 pts) Impose constraints on the inputs and outputs. Re-design your MPC. Impleme the controller on the nonlinear continuous model. Write a script that animates the behavior of t closed-loop system.

```
h = 0.005;
C_tracking = eye(6);
D_zero = zeros(6, size(B_lin_syms_new,2));

sysd_plant = c2d(ss(A_lin_syms_new, B_lin_syms_new, C_tracking, D_zero), h, 'zoh');
Ad = sysd_plant.A;
Bd = sysd_plant.B;
Cd = sysd_plant.C;
Dd = sysd_plant.D;

[n, m] = size(Bd);     % n = 6 states, m = 3 inputs
p = size(Cd,1);       % p = 6 outputs (we track full states)

Phi = [Ad, Bd;
       zeros(m,n), eye(m)];
Gamma = [Bd;
         eye(m)];
Ca = [Cd, zeros(p,m);
      zeros(m,n), eye(m)];
Da = zeros(p+m, m);
```

```
sys_aug = ss(Phi, Gamma, Ca, Da, h);
mpc_obj = mpc(sys_aug, h, Np, Nc);
```

```
-->"Weights.ManipulatedVariables" is empty. Assuming default 0.00000.
-->"Weights.ManipulatedVariablesRate" is empty. Assuming default 0.10000.
-->"Weights.OutputVariables" is empty. Assuming default 1.00000.
   for output(s) y1 y2 y3 and zero weight for output(s) y4 y5 y6 y7 y8 y9
```

```
mpc_obj.Weights.OutputVariables = [2 4 4 0 0 0,  0.01*ones(1,m)]; % last m entries
are for u_prev outputs
mpc_obj.Model.Nominal.X = [ x_eq(:) ; u_eq(:) ]';          % 1 x (n+m)
mpc_obj.Model.Nominal.U = zeros(1,m);                      % du steady = 0
mpc_obj.Model.Nominal.Y = (Ca * [x_eq(:); u_eq(:)])';      % 1 x (p+m)

mpc_obj.OV(1).Min = -0.5;     mpc_obj.OV(1).Max =  0.5;
mpc_obj.OV(2).Min = -pi/3;    mpc_obj.OV(2).Max =  pi/3;
mpc_obj.OV(3).Min = -pi/3;    mpc_obj.OV(3).Max =  pi/3;
% Now set absolute MV (u_prev) bounds on outputs (indices p+1 : p+m)
u_abs_min = [-30; -10; -10];    % put your intended absolute bounds here
u_abs_max = [ 30;  10;  10];
for i = 1:m
    idx = p + i;
    mpc_obj.OV(idx).Min = u_abs_min(i);
    mpc_obj.OV(idx).Max = u_abs_max(i);
end

%% --- MV (du) bounds (these constrain the delta u) ---
% If you want wide du bounds:
for i = 1:m
    mpc_obj.MV(i).Min = -inf; mpc_obj.MV(i).Max = inf;
    mpc_obj.MV(i).RateMin = -50; mpc_obj.MV(i).RateMax = 50;
end

disp('Augmented (Δu) MPC object configured (with u_prev outputs).');
```

```
Augmented (Δu) MPC object configured (with u_prev outputs).
```

```
x = x0;
u_prev = u_eq(:);          % initial absolute MV
x_aug = [x; u_prev];

% initialize mpcstate
mpc_state = mpcstate(mpc_obj);
```

```
-->Assuming output disturbance added to measured output #2 is integrated white noise.
-->Assuming output disturbance added to measured output #3 is integrated white noise.
   Assuming no disturbance added to measured output #1.
-->Assuming output disturbance added to measured output #7 is integrated white noise.
   Assuming no disturbance added to measured output #8.
   Assuming no disturbance added to measured output #9.
-->Assuming output disturbance added to measured output #4 is integrated white noise.
-->Assuming output disturbance added to measured output #5 is integrated white noise.
```

```
-->Assuming output disturbance added to measured output #6 is integrated white noise.
-->"Model.Noise" is empty. Assuming white noise on each measured output.
```

```
% --- Setup and Initialization ---
disp('Starting Augmented MPC (Δu) simulation with OV-based u-bounds...');
```

```
Starting Augmented MPC (Δu) simulation with OV-based u-bounds...
```

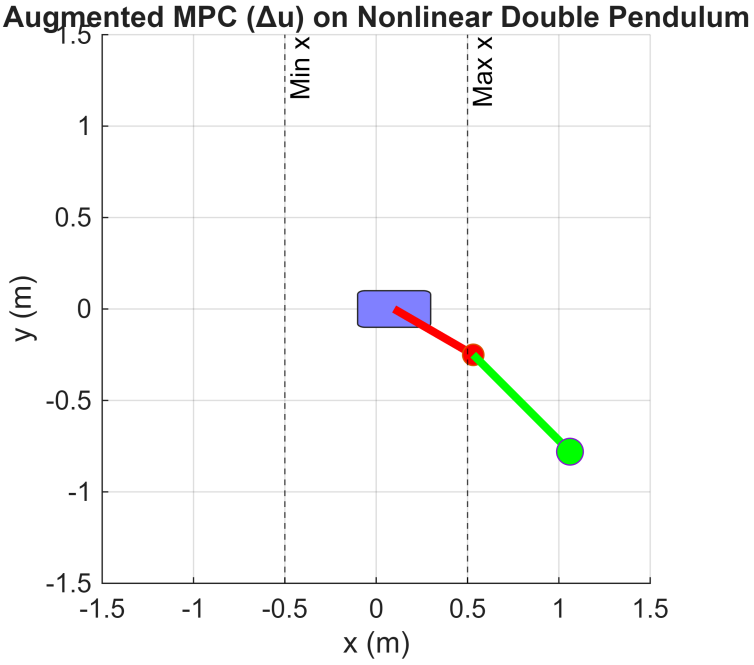
```
tf = 10;
t_history = 0:h:tf;
Nsim = length(t_history);

ode_opts = odeset('RelTol', 1e-6, 'AbsTol', 1e-8);

x0 = [0; deg2rad(35); deg2rad(50); 0; 0; 0];
x = x0;
u_prev = u_eq(:); % Initial absolute MV

% --- History Initialization ---
n = size(x, 1); % n=6
% m is assumed to be 3
x_history = zeros(n, Nsim);
u_history = zeros(m, Nsim);
du_history = zeros(m, Nsim);
x_aug_history = zeros(n+m, Nsim);

x_history(:,1) = x;
u_history(:,1) = u_prev; % u_history(1) is the initial u_eq
du_history(:,1) = zeros(m,1);
x_aug_history(:,1) = [x; u_prev];

% --- MPC State and Reference ---
mpc_state = mpcstate(mpc_obj);
% Define the 9x1 reference vector for the 9 outputs
r_mpc = [Cd*x_eq + Dd*u_eq; u_eq];

% --- 2D Animation Setup ---
figure('Name','Cart-Pendula Animation (Augmented MPC, OV u-bounds)','Color','w');
ax = gca; hold(ax,'on'); axis(ax,'equal'); grid(ax,'on');
title(ax,'Augmented MPC (Δu) on Nonlinear Double Pendulum');
xlabel(ax,'x (m)'); ylabel(ax,'y (m)');

cart_w = 0.4; cart_h = 0.2;
% l1_val and l2_val are assumed to exist

% Initial drawing based on x0
p_cart = [x(1), 0];
p_m1   = [x(1) + l1_val*sin(x(2)), -l1_val*cos(x(2))];
p_m2   = [p_m1(1) + l2_val*sin(x(3)), p_m1(2) - l2_val*cos(x(3))];
```

```
h_cart = rectangle(ax, 'Position', [p_cart(1)-cart_w/2, -cart_h/2, cart_w, cart_h],
...
                   'FaceColor',[0.5 0.5 1], 'Curvature',[0.2 0.2]);
h_line1 = line(ax, [p_cart(1), p_m1(1)], [p_cart(2), p_m1(2)], 'LineWidth', 3,
'Color', 'r');
h_mass1 = plot(ax, p_m1(1), p_m1(2), 'o', 'MarkerSize', 8, 'MarkerFaceColor','r');
h_line2 = line(ax, [p_m1(1), p_m2(1)], [p_m1(2), p_m2(2)], 'LineWidth', 3, 'Color',
'g');
h_mass2 = plot(ax, p_m2(1), p_m2(2), 'o', 'MarkerSize', 10, 'MarkerFaceColor','g');

if ~isinf(mpc_obj.OV(1).Min)
    xline(ax, mpc_obj.OV(1).Min, '--k', 'Min x');
end
if ~isinf(mpc_obj.OV(1).Max)
    xline(ax, mpc_obj.OV(1).Max, '--k', 'Max x');
end

axis(ax, [-1.5 1.5 -1.5 1.5]); % Adjust as needed
drawnow;

fprintf('Running simulation for %d steps...\n', Nsim);
```

```
Running simulation for 2001 steps...
```

```
for k = 1:Nsim-1
    % 1. Construct 9x1 measurement vector [y; u_prev]
    y_plant = Cd * x + Dd * u_prev;
    y_mpc = [ y_plant ; u_prev ];

    % 2. Calculate optimal control increment (du)
    try
        du = mpcmove(mpc_obj, mpc_state, y_mpc, r_mpc);
    catch ME
        warning('mpcmove threw an error at step %d: %s', k, ME.message);
        if k==1
            du = zeros(m,1);
        else
            du = du_history(:,k-1); % Reuse last increment
        end
    end
    if isempty(du)
        du = zeros(m,1); % Fallback
    end

    % 3. Apply absolute control u(k) = u(k-1) + du(k)
    u_applied = u_prev + du;
    u_cmd = u_applied(:);

    % 4. Simulate nonlinear plant one step
    odefun = @(tt,xx) f_num(tt, xx, u_cmd);
```

```
    [~, XX] = ode45(odefun, [0 h], x, ode_opts);
    x_next = XX(end, :)';

    % 5. Store histories
    x = x_next; % This is x(k+1)

    x_history(:, k+1) = x;
    u_history(:, k) = u_applied; % u(k)
    du_history(:, k) = du;       % du(k)
    x_aug_history(:, k) = [x; u_prev];

    % 6. Update u_prev for next loop
    u_prev = u_applied; % u_prev becomes u(k)

    % 7. Update Animation
    p_cart = [x(1), 0];
    p_m1   = [x(1) + l1_val*sin(x(2)), -l1_val*cos(x(2))];
    p_m2   = [p_m1(1) + l2_val*sin(x(3)), p_m1(2) - l2_val*cos(x(3))];

    set(h_cart, 'Position', [p_cart(1)-cart_w/2, -cart_h/2, cart_w, cart_h]);
    set(h_line1, 'XData', [p_cart(1), p_m1(1)], 'YData', [p_cart(2), p_m1(2)]);
    set(h_mass1, 'XData', p_m1(1), 'YData', p_m1(2));
    set(h_line2, 'XData', [p_m1(1), p_m2(1)], 'YData', [p_m1(2), p_m2(2)]);
    set(h_mass2, 'XData', p_m2(1), 'YData', p_m2(2));

    drawnow limitrate;
end
```

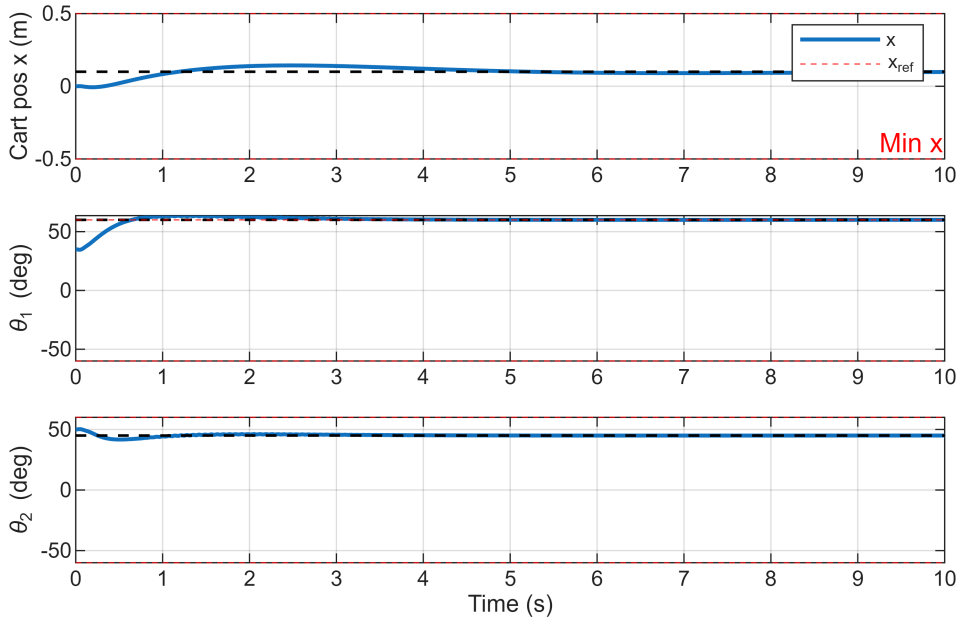



```
% --- Final History Fill ---
u_history(:, Nsim) = u_history(:, Nsim-1);
```

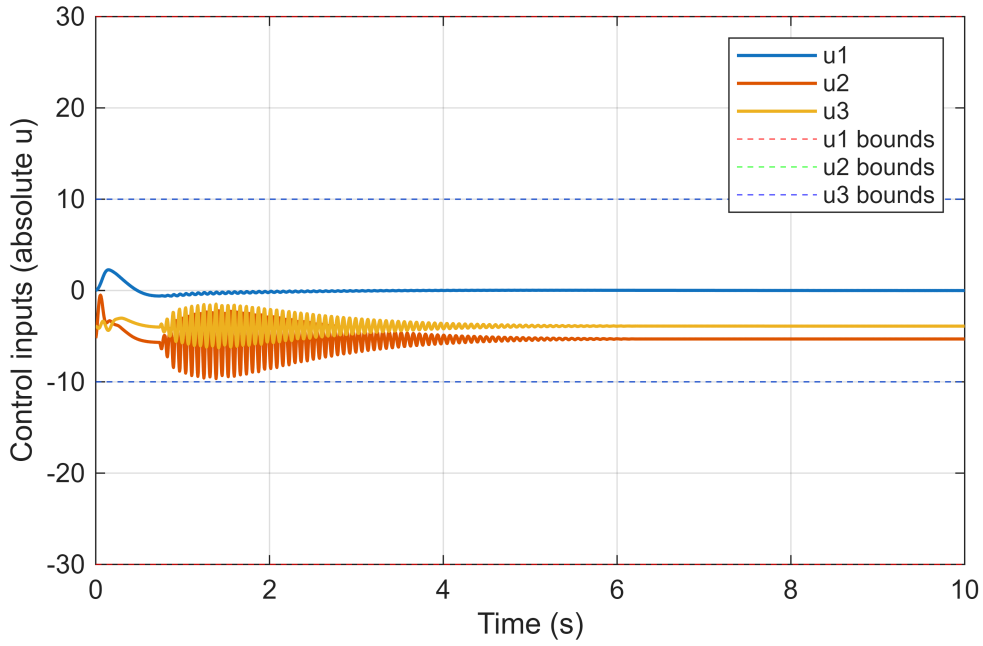
```
du_history(:, Nsim) = du_history(:, Nsim-1);
x_aug_history(:, Nsim) = [x_history(:,Nsim); u_history(:,Nsim)];
disp('Simulation complete.');
```

```
Simulation complete.
```

```
% --- Plot 1: State Trajectories ---
figure('Name','State Trajectories (Augmented MPC, OV u-bounds)','Color','w');
subplot(3,1,1);
plot(t_history, x_history(1,:), 'LineWidth', 1.5); hold on;
% Plot state bounds from mpc_obj.OV(1)
if ~isinf(mpc_obj.OV(1).Min), yline(mpc_obj.OV(1).Min,'r--','Min x'); end
if ~isinf(mpc_obj.OV(1).Max), yline(mpc_obj.OV(1).Max,'r--','Max x'); end
plot(t_history, ones(1,Nsim)*x_eq(1), 'k--', 'LineWidth', 1);
ylabel('Cart pos x (m)'); grid on; legend('x','','x_{ref}');

subplot(3,1,2);
plot(t_history, rad2deg(x_history(2,:)), 'LineWidth', 1.5); hold on;
if ~isinf(mpc_obj.OV(2).Min), yline(rad2deg(mpc_obj.OV(2).Min),'r--'); end
if ~isinf(mpc_obj.OV(2).Max), yline(rad2deg(mpc_obj.OV(2).Max),'r--'); end
plot(t_history, ones(1,Nsim)*rad2deg(x_eq(2)), 'k--', 'LineWidth', 1);
ylabel('\theta_1 (deg)'); grid on;

subplot(3,1,3);
plot(t_history, rad2deg(x_history(3,:)), 'LineWidth', 1.5); hold on;
if ~isinf(mpc_obj.OV(3).Min), yline(rad2deg(mpc_obj.OV(3).Min),'r--'); end
if ~isinf(mpc_obj.OV(3).Max), yline(rad2deg(mpc_obj.OV(3).Max),'r--'); end
plot(t_history, ones(1,Nsim)*rad2deg(x_eq(3)), 'k--', 'LineWidth', 1);
ylabel('\theta_2 (deg)'); grid on;
xlabel('Time (s)');
```



```
figure('Name','Control Effort (with absolute MV bounds)','Color','w');
plot(t_history, u_history(1,:), 'LineWidth', 1.2); hold on;
plot(t_history, u_history(2,:), 'LineWidth', 1.2);
plot(t_history, u_history(3,:), 'LineWidth', 1.2);
yline(u_abs_min(1),'r--'); yline(u_abs_max(1),'r--');
yline(u_abs_min(2),'g--'); yline(u_abs_max(2),'g--');
yline(u_abs_min(3),'b--'); yline(u_abs_max(3),'b--');
ylabel('Control inputs (absolute u)'); xlabel('Time (s)');
legend('u1','u2','u3','u1 bounds','','u2 bounds','','u3 bounds','');
grid on;
```



Problem 4. (7 pts) Implement the combined MPC controller-observer compensator and test it
the nonlinear continuous-time (CT) model. Write a script that animates the behavior of the clo
loop system.

```
h = 0.005;
C_tracking = eye(6);
D_zero = zeros(6, size(B_lin_syms_new,2));

sysd_plant = c2d(ss(A_lin_syms_new, B_lin_syms_new, C_tracking, D_zero), h, 'zoh');
Ad = sysd_plant.A;
Bd = sysd_plant.B;
Cd = sysd_plant.C;
Dd = sysd_plant.D;
```

```
[n, m] = size(Bd);     % n = 6 states, m = 3 inputs
p = size(Cd,1);       % p = 6 outputs (we track full states)

Phi = [Ad, Bd;
       zeros(m,n), eye(m)];
Gamma = [Bd;
         eye(m)];
Ca = [Cd, zeros(p,m)];
Da = zeros(p,m);

sys_aug = ss(Phi, Gamma, Ca, Da, h);

Np = 80; Nc = 40;
mpc_obj = mpc(sys_aug, h, Np, Nc);
```

```
-->"Weights.ManipulatedVariables" is empty. Assuming default 0.00000.
-->"Weights.ManipulatedVariablesRate" is empty. Assuming default 0.10000.
-->"Weights.OutputVariables" is empty. Assuming default 1.00000.
   for output(s) y1 y2 y3 and zero weight for output(s) y4 y5 y6
```

```
mpc_obj.Weights.OutputVariables = [2 4 4 0 0 0];  % heavier weight on pos/angles;
zero for velocities optional
mpc_obj.Weights.ManipulatedVariables = [0.1 0.1 0.1];        % on Δu
mpc_obj.Weights.ManipulatedVariablesRate = [0.2 0.2 0.2];

% Nominal conditions
x_eq = [q_val_1; qd_val_1];               % equilibrium plant state (6x1)
u_eq = u_e_new(:);                        % equilibrium input (3x1)
x_aug_eq = [x_eq; u_eq];                  % augmented equilibrium state (9x1)

mpc_obj.Model.Nominal.X = x_aug_eq';      % note: mpc expects row-vector
mpc_obj.Model.Nominal.Y = (Ca * x_aug_eq)';
mpc_obj.Model.Nominal.U = zeros(1,m);

disp('Augmented (Δu) MPC object configured');
```

```
Augmented (Δu) MPC object configured
```

```
vars_state = [x1;x2;x3;x4;x5;x6];
vars_input = [u_1; u_2; u_3];
f_handle = matlabFunction(f_sym_new_2, 'Vars', {vars_state, vars_input});
f_num = @(t,xx,u_cmd) double( f_handle(xx, u_cmd) );

tf = 15;
t_history = 0:h:tf;
Nsim = length(t_history);

x0 = [0; 0; 0; 0; 0; 0];
u_prev = u_eq(:);
x_aug = [x0; u_prev];
```

```
r = x_eq;


x_history = zeros(n, Nsim);
u_history = zeros(m, Nsim);
du_history = zeros(m, Nsim);
x_aug_history = zeros(n+m, Nsim);

x_history(:,1) = x0;
u_history(:,1) = u_prev;
du_history(:,1) = zeros(m,1);
x_aug_history(:,1) = x_aug;

mpc_state = mpcstate(mpc_obj);
```

```
   Assuming no disturbance added to measured output #2.
   Assuming no disturbance added to measured output #3.
   Assuming no disturbance added to measured output #1.
-->Assuming output disturbance added to measured output #4 is integrated white noise.
-->Assuming output disturbance added to measured output #5 is integrated white noise.
-->Assuming output disturbance added to measured output #6 is integrated white noise.
-->"Model.Noise" is empty. Assuming white noise on each measured output.
```

```
obs_poles = [0.2 0.3 0.4 0.5 0.6 0.1];
L = place(Ad', Cd', obs_poles)';
fprintf('Observer designed. Eigenvalues (phi-L*C):\n');
```

```
Observer designed. Eigenvalues (phi-L*C):
```

```
disp(eig(Ad - L*Cd))
```

```
    0.1000
    0.2000
    0.3000
    0.4000
    0.5000
    0.6000
```

```
disp('Starting MPC + Luenberger Observer simulation for Augmented Model...');
```

```
Starting MPC + Luenberger Observer simulation for Augmented Model...
```

```
ode_opts = odeset('RelTol', 1e-6, 'AbsTol', 1e-8);
tf = 10;
t_history = 0:h:tf;
Nsim = length(t_history);
x0_true = [0.1; deg2rad(45); deg2rad(60); 0; 0; 0];
x0_hat = x_eq;
u_prev = u_eq;
r = x_eq;

x_true = x0_true;
x_hat = x0_hat;
```

```
n = size(Ad, 1);
m = size(Bd, 2);
x_history = zeros(n, Nsim);
xhat_history = zeros(n, Nsim);
u_history = zeros(m, Nsim);

x_history(:, 1) = x_true;
xhat_history(:, 1) = x_hat;
u_history(:, 1) = u_prev;

mpc_state = mpcstate(mpc_obj);
disp('MPC state initialized.');
```

```
MPC state initialized.
```

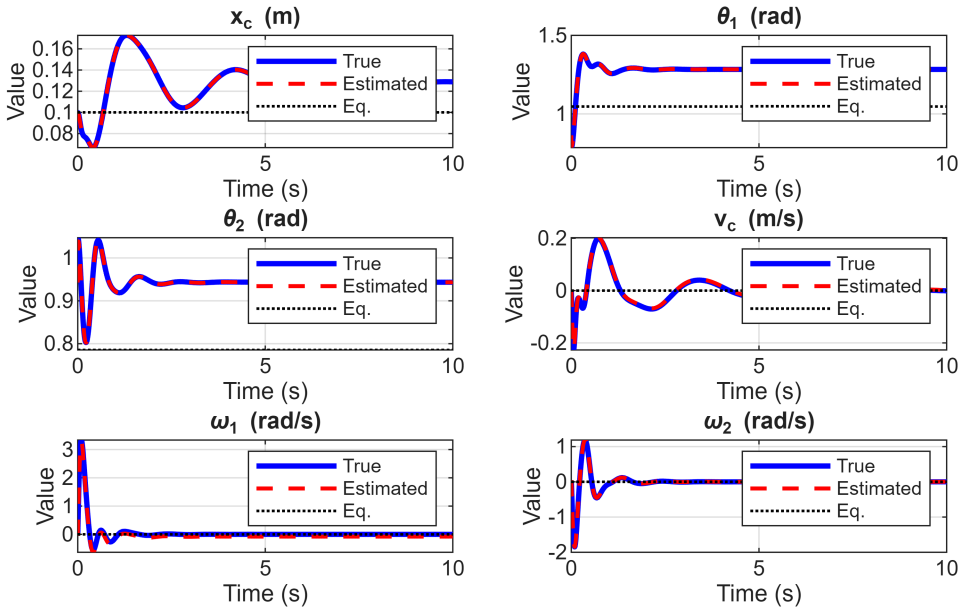
```
fprintf('Running simulation for %d steps...\n', Nsim);
```

```
Running simulation for 2001 steps...
```

```
for k = 1:Nsim-1

    y_meas = Cd * x_true;

    x_hat = x_hat + L * (y_meas - Cd * x_hat);
    xhat_history(:, k) = x_hat;

    x_aug_hat = [x_hat; u_prev];
    mpc_state.Plant = x_aug_hat;

    try
        u_cmd = mpcmove(mpc_obj, mpc_state, y_meas, r);
    catch ME
        warning('mpcmove threw an error at step %d: %s', k, ME.message);
        u_cmd = u_prev;
    end
    if isempty(u_cmd)
        u_cmd = u_eq;
    end
    u_cmd = u_cmd(:);
    u_history(:, k) = u_cmd;
    odefun = @(tt, xx) f_num(tt, xx, u_cmd);
    [~, XX] = ode45(odefun, [t_history(k) t_history(k+1)], x_true, ode_opts);
    x_true = XX(end, :)';
    x_history(:, k+1) = x_true;
    x_hat = Ad * x_hat + Bd * u_cmd;
    xhat_history(:, k+1) = x_hat;     u_prev = u_cmd;
end
u_history(:, Nsim) = u_history(:, Nsim-1);
disp('Simulation complete.');
```

```
Simulation complete.
```

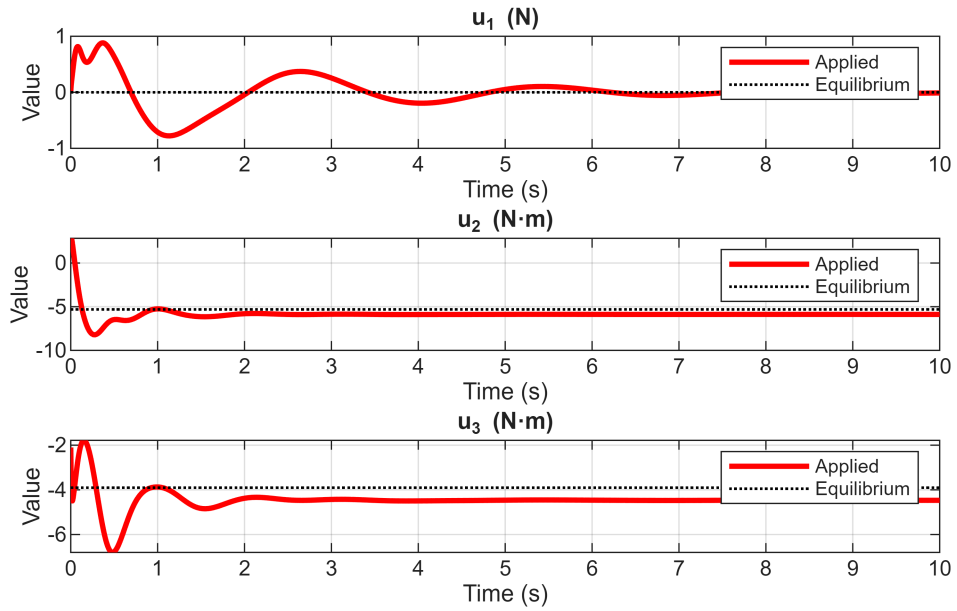
```
t_plot = t_history;

figure('Name', 'Observer Performance (True vs. Estimated States)');
state_names = {'x_c (m)', '\theta_1 (rad)', '\theta_2 (rad)', 'v_c (m/s)',
'\omega_1 (rad/s)', '\omega_2 (rad/s)'};
for i = 1:6
    subplot(3, 2, i);
    plot(t_plot, x_history(i,:), 'b-', 'LineWidth', 2); hold on;
    plot(t_plot, xhat_history(i,:), 'r--', 'LineWidth', 1.5);
    plot(t_plot, x_eq(i)*ones(size(t_plot)), 'k:', 'LineWidth', 1);
    title(state_names{i});
    ylabel('Value');
    xlabel('Time (s)');
    grid on;
end
```




```
% --- Plot Control Inputs ---
figure('Name', 'Control Inputs (Applied to Plant)');
input_names = {'u_1 (N)', 'u_2 (N·m)', 'u_3 (N·m)'};
for i = 1:3
    subplot(3, 1, i);
    plot(t_plot, u_history(i,:), 'r-', 'LineWidth', 2); hold on;
    plot(t_plot, u_eq(i)*ones(size(t_plot)), 'k:', 'LineWidth', 1);
    % Check if constraints are defined before plotting
    if isfield(mpc_obj.MV(i), 'Min') && ~isinf(mpc_obj.MV(i).Min)
        yline(mpc_obj.MV(i).Min,'b--');
    end
```

```
    if isfield(mpc_obj.MV(i), 'Max') && ~isinf(mpc_obj.MV(i).Max)
        yline(mpc_obj.MV(i).Max,'b--');
    end
    title(input_names{i});
    ylabel('Value');
    xlabel('Time (s)');
    legend('Applied', 'Equilibrium', 'Constraints');
    grid on;
end
```

```
Warning: Ignoring extra legend entries.
Warning: Ignoring extra legend entries.
Warning: Ignoring extra legend entries.
```




```
disp('Starting 3D animation rendering...');
```

```
Starting 3D animation rendering...
```

```
cart_w = 0.4;
cart_h = 0.2;
cart_d = 0.2;

w = cart_w / 2; d = cart_d / 2; h_anim = cart_h;
verts_base = [
    -w, -d, 0;  w, -d, 0;  w,  d, 0; -w,  d, 0;
    -w, -d, h_anim;  w, -d, h_anim;  w,  d, h_anim; -w,  d, h_anim
];
faces = [
    1, 2, 6, 5; 2, 3, 7, 6; 3, 4, 8, 7; 4, 1, 5, 8;
    1, 2, 3, 4; 5, 6, 7, 8
```

```
];

fig3d = figure('Name','3D Animation: MPC+Observer (Augmented)','Color','w');
ax3 = axes('Parent',fig3d);
hold(ax3,'on'); grid(ax3,'on'); axis(ax3,'equal');
xlabel(ax3,'X (m)'); ylabel(ax3,'Y (m)'); zlabel(ax3,'Z (m)');
title(ax3,'3D Double Pendulum on Cart (MPC-Controlled w/ Observer)');
view(25, 15); camproj(ax3,'perspective');
axis(ax3,[-1.5 1.5 -1.5 1.5 -1.5 1.5]);

[Xg, Yg] = meshgrid([-2 2], [-1 1]);
surf(ax3, Xg, Yg, zeros(size(Xg)), 'FaceColor',[0.9 0.9 0.9], ...
    'EdgeColor','none', 'FaceAlpha',0.5);

x_k = x_history(:,1);
p_pivot = [x_k(1), 0, cart_h];
p_m1 = [p_pivot(1) + l1_val*sin(x_k(2)), 0, p_pivot(3) - l1_val*cos(x_k(2))];
p_m2 = [p_m1(1) + l2_val*sin(x_k(3)), 0, p_m1(3) - l2_val*cos(x_k(3))];
verts_current = verts_base + [x_k(1), 0, 0];

h_cart = patch(ax3, 'Vertices', verts_current, 'Faces', faces, ...
               'FaceColor',[0.5 0.5 1], 'FaceAlpha', 0.9);
h_line1 = plot3(ax3, [p_pivot(1) p_m1(1)], [0 0], [p_pivot(3) p_m1(3)], 'r-',
'LineWidth',3);
h_mass1 = plot3(ax3, p_m1(1), 0, p_m1(3), 'ro', 'MarkerSize',8,
'MarkerFaceColor','r');
h_line2 = plot3(ax3, [p_m1(1) p_m2(1)], [0 0], [p_m1(3) p_m2(3)], 'g-',
'LineWidth',3);
h_mass2 = plot3(ax3, p_m2(1), 0, p_m2(3), 'go', 'MarkerSize',10,
'MarkerFaceColor','g');
plot3(ax3, [x_eq(1) x_eq(1)], [-1 1], [0 1.5], 'k--', 'LineWidth',1.5);

vidName = 'mpc_observer_augmented_3D.mp4';
fps = round(1/h);
fps = max(1, min(fps, 60));

v3d = VideoWriter(vidName, 'MPEG-4');
v3d.FrameRate = fps;
v3d.Quality = 95;

disp('Fixing video frame size...');
```

```
Fixing video frame size...
```

```
set(fig3d, 'Visible', 'on');
drawnow;

try
    frame = getframe(fig3d);
    sz = size(frame.cdata);
    height = sz(1);
```

```
    width  = sz(2);

    if mod(height,2) ~= 0, height = height - 1; end
    if mod(width,2)  ~= 0, width  = width - 1;  end
    frame.cdata = frame.cdata(1:height, 1:width, :);

    fprintf('Corrected frame size: %d x %d\n', height, width);
    fprintf('Opening video with FrameRate = %d fps, Quality = %d...\n', fps,
v3d.Quality);

    open(v3d);
    writeVideo(v3d, frame);
    disp('Initial frame written successfully.');
    isVideoWritingFailed = false;

catch ME_vid
    warning('Initial frame capture failed: %s', ME_vid.message);
    isVideoWritingFailed = true;
end
```

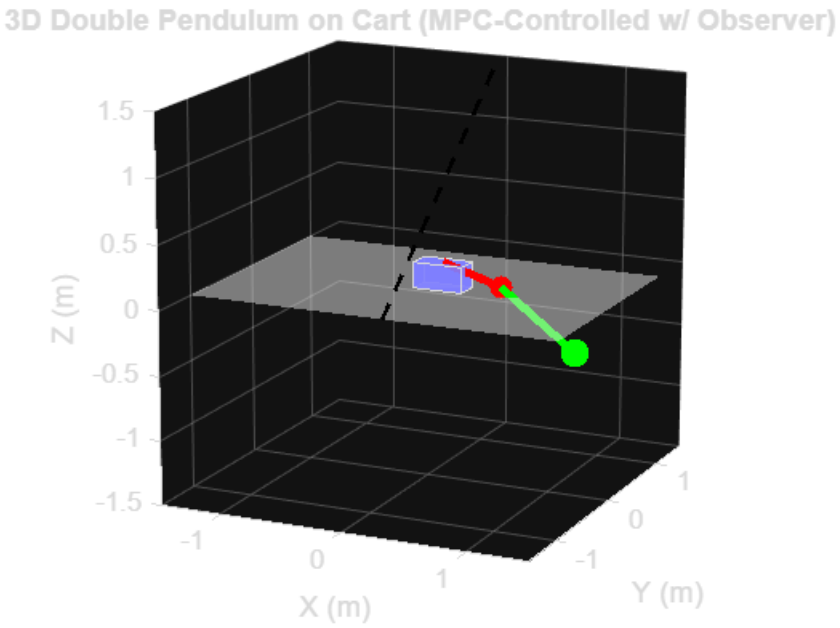
```
Corrected frame size: 422 x 700
Opening video with FrameRate = 60 fps, Quality = 95...
Initial frame written successfully.
```

```
animation_step = 5;
disp('Writing animation frames...');
```

```
Writing animation frames...
```

```
for k = 1+animation_step:animation_step:Nsim
    x_k = x_history(:,k);

    % Calculate new positions
    p_pivot = [x_k(1), 0, cart_h];
    p_m1 = [p_pivot(1) + l1_val*sin(x_k(2)), 0, p_pivot(3) - l1_val*cos(x_k(2))];
    p_m2 = [p_m1(1) + l2_val*sin(x_k(3)), 0, p_m1(3) - l2_val*cos(x_k(3))];
    verts_current = verts_base + [x_k(1), 0, 0];

    % Update graphics handles
    set(h_cart, 'Vertices', verts_current);
    set(h_line1, 'XData', [p_pivot(1) p_m1(1)], 'ZData', [p_pivot(3) p_m1(3)]);
    set(h_mass1, 'XData', p_m1(1), 'ZData', p_m1(3));
    set(h_line2, 'XData', [p_m1(1) p_m2(1)], 'ZData', [p_m1(3) p_m2(3)]);
    set(h_mass2, 'XData', p_m2(1), 'ZData', p_m2(3));

    drawnow;

    % Write frame to video
    if ~isVideoWritingFailed
        try
            frame = getframe(fig3d);
```

```
            frame.cdata = frame.cdata(1:height, 1:width, :);
            writeVideo(v3d, frame);
        catch ME_vid_loop
            warning('Error writing frame %d: %s', k, ME_vid_loop.message);
            isVideoWritingFailed = true; % Stop trying to write
        end
    end
end
```




```
% --- Finalize Video ---
if ~isVideoWritingFailed
    try
        close(v3d);
        disp(['3D animation saved successfully as ', vidName]);
    catch
        warning('Could not close video properly.');
    end
else
    warning('Video writing failed. File may be incomplete or missing.');
end
```

```
3D animation saved successfully as mpc_observer_augmented_3D.mp4
```

```
figure('Name','Delta-u (Δu) inputs','Color','w');
plot(t_history, du_history(1,:), 'LineWidth', 1.2); hold on;
```

```
Error using plot
Specify the coordinates as vectors or matrices of the same size, or as a vector and a matrix that share the same length in at least one dimension.
```

```
plot(t_history, du_history(2,:), 'LineWidth', 1.2);
plot(t_history, du_history(3,:), 'LineWidth', 1.2);
```

```
ylabel('Δu (N or N·m)'); xlabel('Time (s)');
legend('\Delta u_1','\Delta u_2','\Delta u_3'); grid on;

xhat_history = zeros(n, Nsim);
xhat = x_aug_eq(1:n);
xhat_history(:,1) = xhat;
for k=1:Nsim-1
    y_meas = Cd * x_history(:,k);
    xhat = xhat + L*(y_meas - Cd*xhat);           % measurement update
    xhat_history(:,k) = xhat;
    % prediction
    xhat = Ad * xhat + Bd * u_history(:,k);
end
xhat_history(:,Nsim) = xhat;

figure('Name','Observer Performance (True vs. Estimated States)');
state_names = {'x_c (m)', '\theta_1 (rad)', '\theta_2 (rad)', 'v_c (m/s)',
'\omega_1 (rad/s)', '\omega_2 (rad/s)'};
for i = 1:6
    subplot(3, 2, i);
    plot(t_history, x_history(i,:), 'b-', 'LineWidth', 2); hold on;
    plot(t_history, xhat_history(i,:), 'r--', 'LineWidth', 1.5);
    plot(t_history, x_eq(i)*ones(size(t_history)), 'k:', 'LineWidth', 1);
    title(state_names{i});
    ylabel('Value');
    xlabel('Time (s)');
    grid on;
    if i==1
        legend('True','Estimated','Equilibrium');
    end
end

disp('All plots generated. Augmented Δu MPC run finished.');
```

Problem 5. (12 pts) Repeat your MPC designs using the non-augmented model.

Note: no animation video submission for this problem.

```
syms x1 x2 x3 x4 x5 x6 u
syms M m1 m2 l1 l2 g
M = 1.5;
m1 = 0.5;
l1 = 0.5;
m2 = 0.75;
l2 = 0.75;
g = 9.81;
T_cart = 0.5*M*x4^2;
T_pendulum1 = 0.5*m1*((x4+l1*x5*cos(x2))^2 + (l1*x5*sin(x2))^2);
T_pendulum2 = 0.5*m2*((x4 + l1*x5*cos(x2) + l2*x6*cos(x3))^2 + (l1*x5*sin(x2) +
l2*x6*sin(x3))^2);
T_total = simplify(T_cart + T_pendulum1 + T_pendulum2);
V_Cart = 0;
V_pendulum1 = m1*g*l1*cos(x2);
V_pendulum2 = m2*g*(l1*cos(x2)+l2*cos(x3));
V_total = simplify(V_Cart + V_pendulum1 + V_pendulum2);
L = simplify(T_total - V_total);

dL_dx1 = diff(L, x1);
dL_dx2 = diff(L, x2);
dL_dx3 = diff(L, x3);
dL_dx4 = diff(L, x4);
dL_dx5 = diff(L, x5);
dL_dx6 = diff(L, x6);
q = [x1 ;x2; x3];
qd = [x4; x5; x6];
syms qdd1 qdd2 qdd3 real
qdd = [qdd1; qdd2; qdd3];
Jaco = jacobian([dL_dx4;dL_dx5;dL_dx6],[q;qd]);
d_dt_dLdqdot = Jaco*[qd; qdd];
dLdq = [dL_dx1;dL_dx2;dL_dx3];
Force = [u; 0; 0];
Lagrange_eq = simplify(d_dt_dLdqdot - dLdq -Force);
D = sym(zeros(3,3));
for i=1:3
  for j=1:3
    D(i,j) = simplify(diff(diff(T_total, qd(i)), qd(j)));
  end

end
D
```

```
D =
```

$$\begin{pmatrix} \frac{11}{4} & \frac{5\cos(x_2)}{8} & \frac{9\cos(x_3)}{16} \\ \frac{5\cos(x_2)}{8} & \frac{5}{16} & \frac{9\cos(x_2-x_3)}{32} \\ \frac{9\cos(x_3)}{16} & \frac{9\cos(x_2-x_3)}{32} & \frac{27}{64} \end{pmatrix}$$

```
G = simplify([ diff(V_total, x1);
      diff(V_total, x2);
      diff(V_total, x3) ])
```

G =

$$\begin{pmatrix} 0 \\ -\frac{981\sin(x_2)}{160} \\ -\frac{8829\sin(x_3)}{1600} \end{pmatrix}$$

```
n = 3;
c = sym(zeros(n,n,n));
for i=1:n
  for j=1:n
    for k=1:n
      c(i,j,k) = 1/2*( diff(D(i,j), q(k)) + diff(D(i,k), q(j)) - diff(D(j,k),
q(i)) );
    end
  end
end

C = sym(zeros(n,n));
for i=1:n
  for j=1:n
    C(i,j) = simplify( reshape(c(i,j,:),[n,1]).' * qd );
  end
end

H = [1;0;0];
qdd = simplify(D\(H*u - C*qd-G));
X = [q;qd];
f_sym = [qd; qdd]
```

f_sym =

$$\begin{pmatrix} x_4 \\ x_5 \\ x_6 \\ -\dfrac{280\,u - 981\sin(2\,x_2) + 100\,{x_5}^2\sin(x_2) + 45\,{x_6}^2\sin(x_3) - 120\,u\cos(} \\ \dfrac{1575\,{x_6}^2\sin(x_2 - x_3) - 30411\sin(x_2) - 8829\sin(x_2 - 2\,x_3) + 250\,{x_5}^2\sin(2\,x_2) + 1400\,u\cos(x_2) + 450\,{x_5}^2\,s} \\ -\dfrac{300\sin(x_2 - x_3)\,{x_5}^2 + 135\sin(\sigma_3)\,{x_6}^2 - 2943\,\sigma_1 + 2943\sin(x_3) + 200\,u\cos(2\,} \end{pmatrix}$$

where

$$\sigma_1 = \sin(2\,x_2 - x_3)$$

$$\sigma_2 = 5\cos(2\,x_2) + 9\cos(\sigma_3) - 26$$

$$\sigma_3 = 2\,x_2 - 2\,x_3$$

```
D_inv = inv(D)
```

```
D_inv =
```

$$\begin{pmatrix} \dfrac{4\,\left(3\cos(x_2 - x_3)^2 - 5\right)}{\sigma_4} & \sigma_2 & \sigma_3 \\ \sigma_2 & \dfrac{16\,\left(3\cos(x_3)^2 - 11\right)}{\sigma_4} & \sigma_1 \\ \sigma_3 & \sigma_1 & \dfrac{320\,\left(5\cos(x_2)^2 - 11\right)}{27\,\sigma_4} \end{pmatrix}$$

where

$$\sigma_1 = \frac{32\,\left(11\cos(x_2 - x_3) - 5\cos(x_2)\cos(x_3)\right)}{3\,\sigma_4}$$

$$\sigma_2 = \frac{8\,\left(5\cos(x_2) - 3\cos(x_2 - x_3)\cos(x_3)\right)}{\sigma_4}$$

$$\sigma_3 = \frac{80\,\left(\cos(x_3) - \cos(x_2 - x_3)\cos(x_2)\right)}{3\,\sigma_4}$$

$$\sigma_4 = 33\cos(x_2 - x_3)^2 - 30\cos(x_2 - x_3)\cos(x_2)\cos(x_3) + 25\cos(x_2)^2 + 15\cos(x_3)^2 - 55$$

```
C
```

C =

$$\begin{pmatrix} 0 & -\frac{5\,x_5\sin(x_2)}{8} & -\frac{9\,x_6\sin(x_3)}{16} \\ 0 & 0 & \frac{9\,x_6\sin(x_2-x_3)}{32} \\ 0 & -\frac{9\,x_5\sin(x_2-x_3)}{32} & 0 \end{pmatrix}$$

```
syms u_1 u_2
 u_new = [u_1; u_2];
 H_new = [1 0;
          0 1;
          0 0];
 qdd_new_1 = simplify(D\(H_new*u_new - C*qd-G));
 X_new_1 = [q;qd];
 f_sym_new_1 = [qd; qdd_new_1]
```

f_sym_new_1 =

$$\begin{pmatrix} x_4 \\ x_5 \\ x_6 \\ -\frac{280\,u_1 - 981\sin(2\,x_2) + 100\,{x_5}^2\sin(x_2) + 45\,{x_6}^2\sin(x_3) - 120\,u_1\cos(\sigma_4)}{20\,\sigma_3} \\ \frac{1200\,u_2\cos(2\,x_3) - 8829\sin(x_2-2\,x_3) - 30411\sin(x_2) - 7600\,u_2 + 1575\,{x_6}^2\sin(x_2-x_3) + 250\,{x_5}^2\sin(2\,x}{50\,\sigma_3} \\ -\frac{300\sin(x_2-x_3)\,{x_5}^2 + 135\sin(\sigma_4)\,{x_6}^2 - 2943\,\sigma_1 + 2943\sin(x_3) + 200\,u_1\cos(2\,x_2-x_3) + 4(}{75\cos(2\,x_2) + 135\cos(\sigma_4) - 390} \end{pmatrix}$$

where

$$\sigma_1 = \sin(2\,x_2 - x_3)$$

$$\sigma_2 = \cos(x_2 - 2\,x_3)$$

$$\sigma_3 = 5\cos(2\,x_2) + 9\cos(\sigma_4) - 26$$

$$\sigma_4 = 2\,x_2 - 2\,x_3$$

```
syms u_e1 u_e2 real
q_val_1 = [0.1; deg2rad(60); deg2rad(45)];
qd_val_1 = [0; 0; 0];
G_val = double(subs(G, [x1 x2 x3 x4 x5 x6], [q_val_1' qd_val_1']));
eq = H_new*[u_e1 ; u_e2] == G_val;
sol_u = solve(eq, u_e1, u_e2, 'Real', true)
```

```
sol_u = struct with fields:
    u_e1: [0×1 sym]
    u_e2: [0×1 sym]
```

```
syms u_3 u_e3 real
u_new_2 = [u_1; u_2; u_3];
H_new_2 = [1 0 0;
        0 1 0 ;
        0 0 1];
qdd_new_2 = simplify(D\(H_new_2*u_new_2 - C*qd-G));
X_new_2 = [q;qd];
f_sym_new_2 = [qd; qdd_new_2]
```

f_sym_new_2 =

$$\left(\begin{array}{c} -\frac{840\,u_1 - 2943\,\sin(2\,x_2) + 300\,{x_5}^2\,\sin(x_2) + 135\,{x_6}^2\,\sin(x_3) - 36\text{(}}{} \\ \frac{3600\,u_2\,\cos(2\,x_3) - 26487\,\sin(x_2 - 2\,x_3) - 91233\,\sin(x_2) - 22800\,u_2 + 4725\,{x_6}^2\,\sin(x_2 - x_3) + 750\,{x_5}^2\,\sin(\text{:}}{} \\ -\frac{2700\,\sin(x_2 - x_3)\,{x_5}^2 + 1215\,\sin(\sigma_5)\,{x_6}^2 + 13600\,u_3 - 26487\,\sigma_1 + 26487\,\sin(x_3)}{} \end{array} \right.$$

where

$\sigma_1 = \sin(2\,x_2 - x_3)$

$\sigma_2 = \cos(2\,x_2 - x_3)$

$\sigma_3 = \cos(x_2 - 2\,x_3)$

$\sigma_4 = 5\,\cos(2\,x_2) + 9\,\cos(\sigma_5) - 26$

$\sigma_5 = 2\,x_2 - 2\,x_3$

```
q_val_1 = [0.1; deg2rad(60); deg2rad(45)];
qd_val_1 = [0; 0; 0];
G_val = double(subs(G, [x1 x2 x3 x4 x5 x6], [q_val_1' qd_val_1']));
eq = H_new_2*[u_e1 ; u_e2 ; u_e3] == G_val;
sol_u = solve(eq, u_e1, u_e2, u_e3, 'Real', true);
u_e_new = double(struct2array(sol_u))
```

```
u_e_new = 1×3
         0   -5.3098   -3.9019
```

```
A_sym_new = simplify(jacobian(f_sym_new_2, X));
B_sym_new = simplify(jacobian(f_sym_new_2, u_new_2));
vars = [x1; x2; x3; x4; x5; x6; u_1; u_2; u_3];
vals = [0.1; deg2rad(60); deg2rad(45); 0; 0; 0; u_e_new'];

A_lin_syms_new = double(subs(A_sym_new, vars, vals))
```

```
A_lin_syms_new = 6×6
         0         0         0    1.0000         0         0
         0         0         0         0    1.0000         0
         0         0         0         0         0    1.0000
         0   -0.5341   -1.1264         0         0         0
         0   22.5046  -17.8041         0         0         0
         0  -13.9883   21.7759         0         0         0
```

```
B_lin_syms_new = double(subs(B_sym_new, vars, vals))
```

```
B_lin_syms_new = 6×3
         0         0         0
         0         0         0
         0         0         0
    0.4252   -0.1742   -0.2887
   -0.1742    7.3409   -4.5629
   -0.2887   -4.5629    5.5808
```

```
C_new = [1 0 0 0 0 0;
         0 1 0 0 0 0;
         0 0 1 0 0 0];
h = 0.005
```

```
h =
0.0050
```

```
[phi_non_aug,gamma_non_aug]=c2d(A_lin_syms_new,B_lin_syms_new,h)
```

```
phi_non_aug = 6×6
    1.0000   -0.0000   -0.0000    0.0050   -0.0000   -0.0000
         0    1.0003   -0.0002         0    0.0050   -0.0000
         0   -0.0002    1.0003         0   -0.0000    0.0050
         0   -0.0027   -0.0056    1.0000   -0.0000   -0.0000
         0    0.1125   -0.0890         0    1.0003   -0.0002
         0   -0.0700    0.1089         0   -0.0002    1.0003
gamma_non_aug = 6×3
    0.0000   -0.0000   -0.0000
   -0.0000    0.0001   -0.0001
   -0.0000   -0.0001    0.0001
    0.0021   -0.0009   -0.0014
   -0.0009    0.0367   -0.0228
   -0.0014   -0.0228    0.0279
```

```
q_val_1 = [0.1; deg2rad(60); deg2rad(45)];
qd_val_1 = [0; 0; 0];
```

```
x_eq = [q_val_1; qd_val_1];

if exist('u_e_new','var')
    u_e = u_e_new(:);
else
    u_e = zeros(3,1);
end
A_lin = A_lin_syms_new;
B_lin = B_lin_syms_new;

C_new = [1 0 0 0 0 0;
         0 1 0 0 0 0;
         0 0 1 0 0 0];
h = 0.005;
Np = 80;
Nc = 40;

D_new = zeros(size(C_new,1), size(B_lin,2));
sys_c = ss(A_lin, B_lin, C_new, D_new);
sys_c.StateName = {'x_pos','theta1','theta2','x_vel','theta1_vel','theta2_vel'};
sys_c.InputName = {'u1','u2','u3'};
sys_c.OutputName = {'y_x','y_theta1','y_theta2'};
mpc_obj = mpc(sys_c, h, Np, Nc);
```

```
-->"Weights.ManipulatedVariables" is empty. Assuming default 0.00000.
-->"Weights.ManipulatedVariablesRate" is empty. Assuming default 0.10000.
-->"Weights.OutputVariables" is empty. Assuming default 1.00000.
```

```
mpc_obj.Weights.OutputVariables = [2 4 4];
mpc_obj.Weights.ManipulatedVariablesRate = [0.2 0.2 0.2];
mpc_obj.Weights.ManipulatedVariables = [0.1 0.1 0.1];
mpc_obj.Model.Nominal.U = u_e;
mpc_obj.Model.Nominal.X = x_eq;
mpc_obj.Model.Nominal.Y = C_new * x_eq;

disp('MPC object configured');
```

```
MPC object configured
```

```
sys_d = c2d(sys_c, h);
phi = sys_d.A;
eig(phi)
```

```
ans = 6×1
    1.0000
    1.0000
    1.0313
    1.0127
    0.9697
    0.9875
```

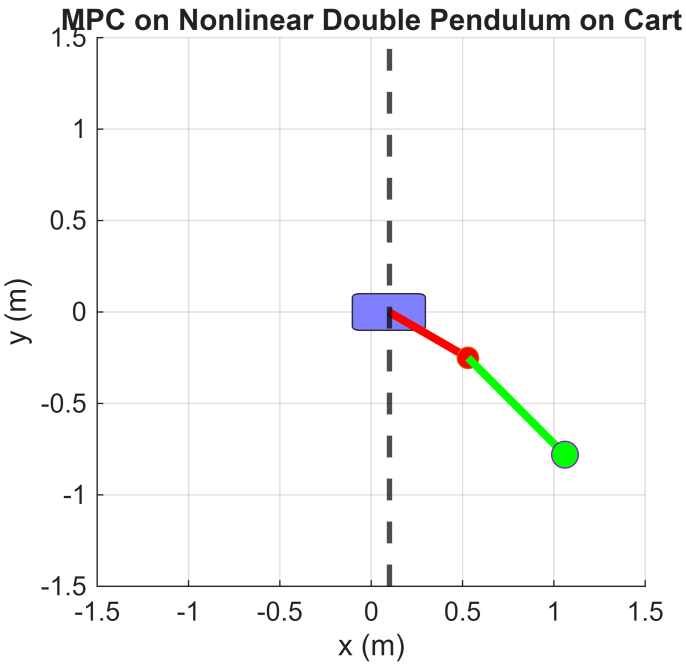
```
gamma = sys_d.B;
vars_state = [x1;x2;x3;x4;x5;x6];
vars_input = [u_1; u_2; u_3];
```

```
f_handle = matlabFunction(f_sym_new_2, 'Vars', {vars_state, vars_input});

f_num = @(t,xx,u_cmd) double( f_handle(xx, u_cmd) );

tf = 10;
t_history = 0:h:tf;
Nsim = length(t_history);

x0 = [0; 0; 0; 0; 0; 0];
x = x0;
y = C_new * x;

r = C_new * x_eq;

x_history = zeros(length(x_eq), Nsim);
u_history = zeros(size(B_lin,2), Nsim);
x_history(:,1) = x;
u_history(:,1) = u_e;

mpc_state = mpcstate(mpc_obj);
```

```
-->Converting model to discrete time.
-->Assuming output disturbance added to measured output #2 is integrated white noise.
-->Assuming output disturbance added to measured output #3 is integrated white noise.
   Assuming no disturbance added to measured output #1.
-->"Model.Noise" empty. Assuming white noise on each measured output.
```

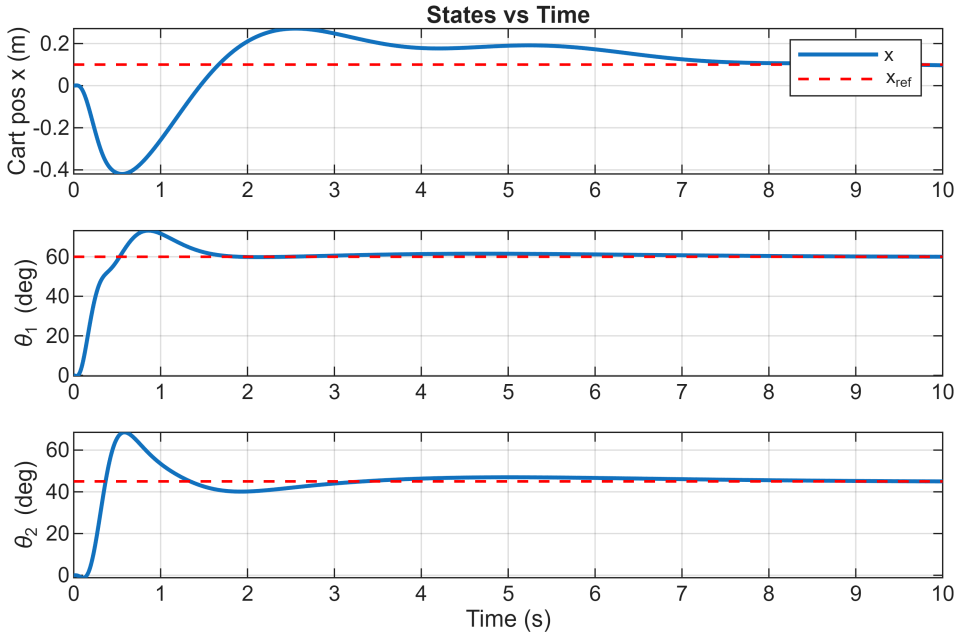
```
figure('Name','Cart-Pendula Animation (nonlinear plant)','Color','w');
ax = gca; hold(ax,'on'); axis(ax,'equal'); grid(ax,'on');
title(ax,'MPC on Nonlinear Double Pendulum on Cart');
xlabel(ax,'x (m)'); ylabel(ax,'y (m)');

cart_w = 0.4; cart_h = 0.2;
l1_val = double(l1); l2_val = double(l2);

p_cart = [x(1), 0];
p_m1   = [x(1) + l1_val*sin(x(2)), -l1_val*cos(x(2))];
p_m2   = [p_m1(1) + l2_val*sin(x(3)), p_m1(2) - l2_val*cos(x(3))];

h_cart = rectangle(ax, 'Position', [p_cart(1)-cart_w/2, -cart_h/2, cart_w, cart_h],
...
                   'FaceColor',[0.5 0.5 1], 'Curvature',[0.2 0.2]);
h_line1 = line(ax, [p_cart(1), p_m1(1)], [p_cart(2), p_m1(2)], 'LineWidth', 3,
'Color', 'r');
h_mass1 = plot(ax, p_m1(1), p_m1(2), 'o', 'MarkerSize', 8, 'MarkerFaceColor','r');
h_line2 = line(ax, [p_m1(1), p_m2(1)], [p_m1(2), p_m2(2)], 'LineWidth', 3, 'Color',
'g');
h_mass2 = plot(ax, p_m2(1), p_m2(2), 'o', 'MarkerSize', 10, 'MarkerFaceColor','g');
h_target_pos = xline(ax, x_eq(1), 'k--', 'LineWidth', 2);

axis(ax, [-1.5 1.5 -1.5 1.5]);
```

```
drawnow;

for k = 1:Nsim-1
    y = C_new * x;

    u = mpcmove(mpc_obj, mpc_state, y, r);
    if isempty(u)
        u = u_e;
    end
    u_cmd = u(:);
    odefun = @(tt,xx) f_num(tt, xx, u_cmd);
    [~, XX] = ode45(odefun, [0 h], x);
    x_next = XX(end, :)';

    x = x_next;
    x_history(:, k+1) = x;
    u_history(:, k) = u;
    p_cart = [x(1), 0];
    p_m1   = [x(1) + l1_val*sin(x(2)), -l1_val*cos(x(2))];
    p_m2   = [p_m1(1) + l2_val*sin(x(3)), p_m1(2) - l2_val*cos(x(3))];

    set(h_cart, 'Position', [p_cart(1)-cart_w/2, -cart_h/2, cart_w, cart_h]);
    set(h_line1, 'XData', [p_cart(1), p_m1(1)], 'YData', [p_cart(2), p_m1(2)]);
    set(h_mass1, 'XData', p_m1(1), 'YData', p_m1(2));
    set(h_line2, 'XData', [p_m1(1), p_m2(1)], 'YData', [p_m1(2), p_m2(2)]);
    set(h_mass2, 'XData', p_m2(1), 'YData', p_m2(2));

    drawnow limitrate;
end
```

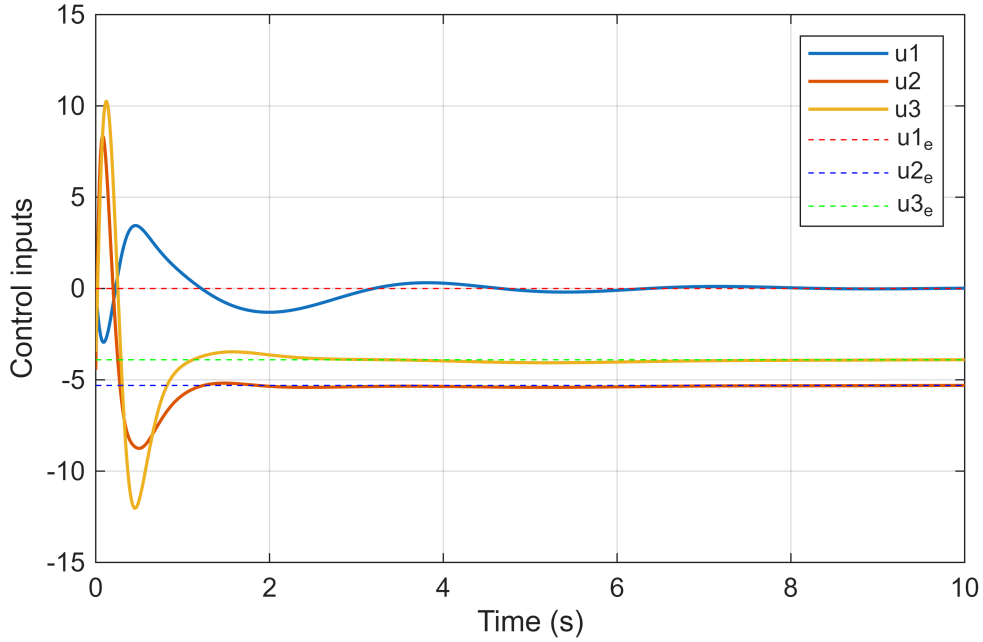


Warning: An error occurred while drawing the scene: Error in SceneTree: Could not find node in replaceChild

```
u_history(:, Nsim) = u_history(:, Nsim-1);

figure('Name','State Trajectories','Color','w');
subplot(3,1,1);
plot(t_history, x_history(1,:), 'LineWidth', 1.5); hold on;
plot(t_history, ones(1,Nsim)*x_eq(1), 'r--', 'LineWidth', 1);
ylabel('Cart pos x (m)'); grid on; legend('x','x_{ref}');
title('States vs Time');

subplot(3,1,2);
plot(t_history, rad2deg(x_history(2,:)), 'LineWidth', 1.5); hold on;
plot(t_history, ones(1,Nsim)*rad2deg(x_eq(2)), 'r--', 'LineWidth', 1);
ylabel('\theta_1 (deg)'); grid on;

subplot(3,1,3);
plot(t_history, rad2deg(x_history(3,:)), 'LineWidth', 1.5); hold on;
plot(t_history, ones(1,Nsim)*rad2deg(x_eq(3)), 'r--', 'LineWidth', 1);
ylabel('\theta_2 (deg)'); grid on;
xlabel('Time (s)');
```




```
figure('Name','Control Effort','Color','w');
plot(t_history, u_history(1,:), 'LineWidth', 1.2); hold on;
plot(t_history, u_history(2,:), 'LineWidth', 1.2);
plot(t_history, u_history(3,:), 'LineWidth', 1.2);
plot(t_history, ones(1,Nsim)*u_e(1), 'r--');
plot(t_history, ones(1,Nsim)*u_e(2), 'b--');
plot(t_history, ones(1,Nsim)*u_e(3), 'g--');
ylabel('Control inputs'); xlabel('Time (s)');
legend('u1','u2','u3','u1_e','u2_e','u3_e'); grid on;
```




```
disp('Simulation complete.');
```

```
Simulation complete.
```

```
h = 0.005;
sys_c = ss(A_lin_syms_new, B_lin_syms_new, C_new, zeros(size(C_new,1),
size(B_lin_syms_new,2)));
sys_c.StateName = {'x_pos','theta1','theta2','x_vel','theta1_vel','theta2_vel'};
sys_c.InputName = {'u1','u2','u3'};
sys_c.OutputName = {'y_x','y_theta1','y_theta2'};
Np = 80;
Nc = 40;
mpc_obj = mpc(sys_c, h, Np, Nc);
```

```
-->"Weights.ManipulatedVariables" is empty. Assuming default 0.00000.
-->"Weights.ManipulatedVariablesRate" is empty. Assuming default 0.10000.
-->"Weights.OutputVariables" is empty. Assuming default 1.00000.
```

```
mpc_obj.Weights.OutputVariables = [2 4 4];
mpc_obj.Weights.ManipulatedVariablesRate = [0.2 0.2 0.2];
mpc_obj.Weights.ManipulatedVariables = [0.1 0.1 0.1];

% Nominal values
x_eq = [q_val_1; qd_val_1];
mpc_obj.Model.Nominal.U = u_e_new(:)';
mpc_obj.Model.Nominal.X = x_eq;
mpc_obj.Model.Nominal.Y = C_new * x_eq;

mpc_obj.MV(1).Min = -30;
mpc_obj.MV(1).Max =  30;
```

```
mpc_obj.MV(2).Min = -10;
mpc_obj.MV(2).Max =  10;
mpc_obj.MV(3).Min = -10;
mpc_obj.MV(3).Max =  10;
mpc_obj.MV(1).RateMin = -50;
mpc_obj.MV(1).RateMax =  50;
mpc_obj.MV(2).RateMin = -50;
mpc_obj.MV(2).RateMax =  50;
mpc_obj.MV(3).RateMin = -50;
mpc_obj.MV(3).RateMax =  50;

mpc_obj.OV(1).Min = -0.5;
mpc_obj.OV(1).Max =  0.5;
mpc_obj.OV(2).Min = -pi/3;
mpc_obj.OV(2).Max =  pi/3;
mpc_obj.OV(3).Min = -pi/3;
mpc_obj.OV(3).Max =  pi/3;
disp('MPC object configured with MV & OV constraints.');
```

```
MPC object configured with MV & OV constraints.
```

```
mpc_obj
```

```
MPC object (created on 08-Nov-2025 23:30:35):
---------------------------------------------
Sampling time:      0.005 (seconds)
Prediction Horizon: 80
Control Horizon:    40

Plant Model:
                                    --------------
      3  manipulated variable(s)   -->|  6 states  |
                                    |              |-->  3 measured output(s)
      0  measured disturbance(s)   -->|  3 inputs  |
                                    |              |-->  0 unmeasured output(s)
      0  unmeasured disturbance(s) -->|  3 outputs |
                                    --------------
Disturbance and Noise Models:
        Output disturbance model: default (type "getoutdist(mpc_obj)" for details)
         Measurement noise model: default (unity gain after scaling)

Weights:
        ManipulatedVariables: [0.1000 0.1000 0.1000]
    ManipulatedVariablesRate: [0.2000 0.2000 0.2000]
             OutputVariables: [2 4 4]
                         ECR: 100000

State Estimation:  Default Kalman Filter (type "getEstimator(mpc_obj)" for details)

Constraints:
 -30 <= u1 <= 30, -50 <= u1/rate <= 50,            -0.5 <= y_x <= 0.5
 -10 <= u2 <= 10, -50 <= u2/rate <= 50, -1.0472 <= y_theta1 <= 1.0472
 -10 <= u3 <= 10, -50 <= u3/rate <= 50, -1.0472 <= y_theta2 <= 1.0472

Use built-in "active-set" QP solver with MaxIterations of 120.
```

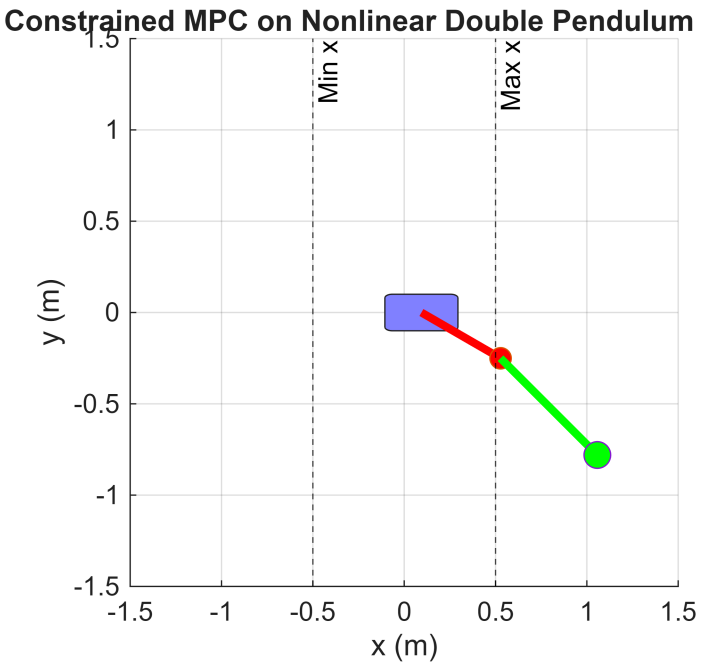
```
sys_d = c2d(sys_c, h);
```

```
phi = sys_d.A;
gamma = sys_d.B;
vars_state = [x1;x2;x3;x4;x5;x6];
vars_input = [u_1; u_2; u_3];
f_handle = matlabFunction(f_sym_new_2, 'Vars', {vars_state, vars_input});
f_num = @(t,xx,u_cmd) double( f_handle(xx, u_cmd) );

tf = 10;
t_history = 0:h:tf;
Nsim = length(t_history);
x0 = [0; 0; 0; 0; 0; 0];
x = x0;
y = C_new * x;

r = C_new * x_eq;

x_history = zeros(length(x_eq), Nsim);
u_history = zeros(size(B_lin_syms_new,2), Nsim);
x_history(:,1) = x;
u_history(:,1) = u_e_new;
mpc_state = mpcstate(mpc_obj);
```

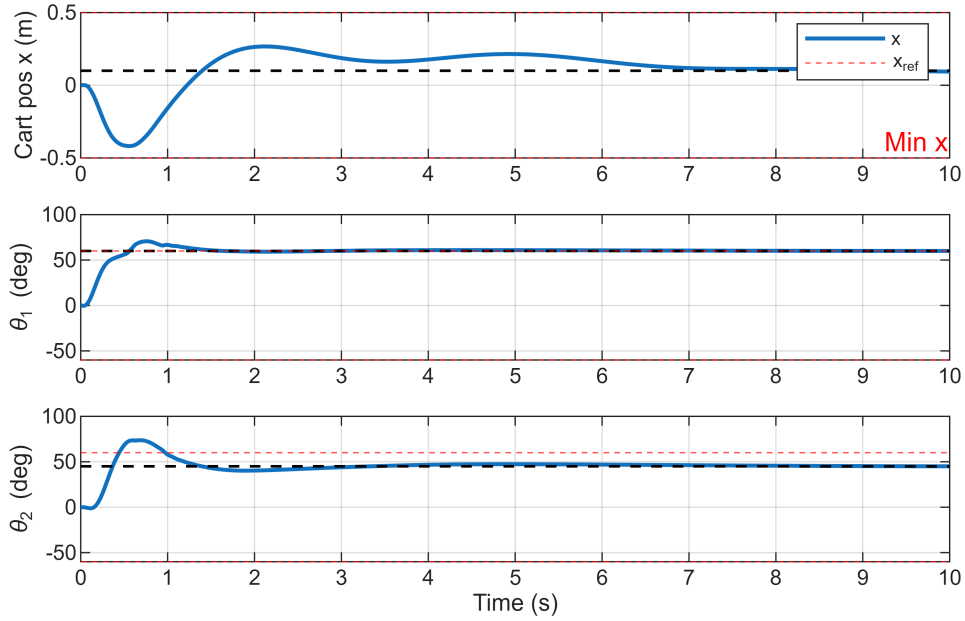
```
-->Converting model to discrete time.
-->Assuming output disturbance added to measured output #2 is integrated white noise.
-->Assuming output disturbance added to measured output #3 is integrated white noise.
   Assuming no disturbance added to measured output #1.
-->"Model.Noise" empty. Assuming white noise on each measured output.
```

```
figure('Name','Cart-Pendula Animation (nonlinear plant, constrained
MPC)','Color','w');
ax = gca; hold(ax,'on'); axis(ax,'equal'); grid(ax,'on');
title(ax,'Constrained MPC on Nonlinear Double Pendulum on Cart');
xlabel(ax,'x (m)'); ylabel(ax,'y (m)');

cart_w = 0.4; cart_h = 0.2;
l1_val = double(l1); l2_val = double(l2);

p_cart = [x(1), 0];
p_m1   = [x(1) + l1_val*sin(x(2)), -l1_val*cos(x(2))];
p_m2   = [p_m1(1) + l2_val*sin(x(3)), p_m1(2) - l2_val*cos(x(3))];

h_cart = rectangle(ax, 'Position', [p_cart(1)-cart_w/2, -cart_h/2, cart_w, cart_h],
...
                   'FaceColor',[0.5 0.5 1], 'Curvature',[0.2 0.2]);
h_line1 = line(ax, [p_cart(1), p_m1(1)], [p_cart(2), p_m1(2)], 'LineWidth', 3,
'Color', 'r');
h_mass1 = plot(ax, p_m1(1), p_m1(2), 'o', 'MarkerSize', 8, 'MarkerFaceColor','r');
h_line2 = line(ax, [p_m1(1), p_m2(1)], [p_m1(2), p_m2(2)], 'LineWidth', 3, 'Color',
'g');
h_mass2 = plot(ax, p_m2(1), p_m2(2), 'o', 'MarkerSize', 10, 'MarkerFaceColor','g');
xline(ax, mpc_obj.OV(1).Min, '--k', 'Min x');
xline(ax, mpc_obj.OV(1).Max, '--k', 'Max x');
```

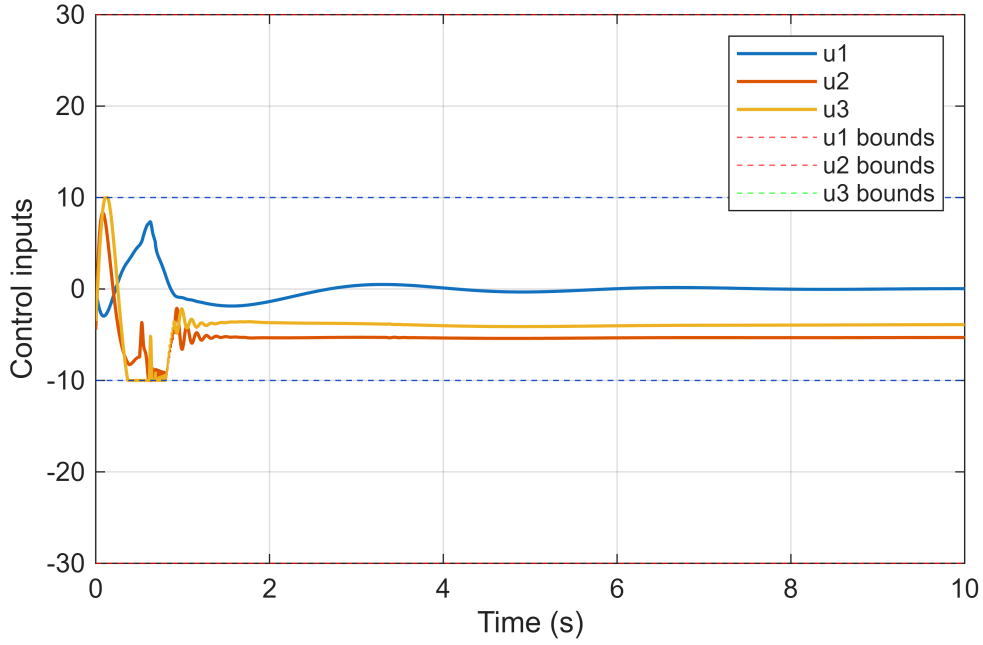
```
axis(ax, [-1.5 1.5 -1.5 1.5]);
drawnow;

for k = 1:Nsim-1
    y = C_new * x;
    try
        u = mpcmove(mpc_obj, mpc_state, y, r);
    catch ME
        warning('mpcmove threw an error at step %d: %s', k, ME.message);
        u = mpc_state.MV;
    end
    if isempty(u)
        u = u_e_new;
    end

    u_cmd = u(:);
    odefun = @(tt,xx) f_num(tt, xx, u_cmd);
    [~, XX] = ode45(odefun, [0 h], x);
    x_next = XX(end, :)';
    x = x_next;
    x_history(:, k+1) = x;
    u_history(:, k) = u;
    p_cart = [x(1), 0];
    p_m1   = [x(1) + l1_val*sin(x(2)), -l1_val*cos(x(2))];
    p_m2   = [p_m1(1) + l2_val*sin(x(3)), p_m1(2) - l2_val*cos(x(3))];

    set(h_cart, 'Position', [p_cart(1)-cart_w/2, -cart_h/2, cart_w, cart_h]);
    set(h_line1, 'XData', [p_cart(1), p_m1(1)], 'YData', [p_cart(2), p_m1(2)]);
    set(h_mass1, 'XData', p_m1(1), 'YData', p_m1(2));
    set(h_line2, 'XData', [p_m1(1), p_m2(1)], 'YData', [p_m1(2), p_m2(2)]);
    set(h_mass2, 'XData', p_m2(1), 'YData', p_m2(2));

    drawnow limitrate;
end
```




```
u_history(:, Nsim) = u_history(:, Nsim-1);

figure('Name','State Trajectories','Color','w');
subplot(3,1,1);
plot(t_history, x_history(1,:), 'LineWidth', 1.5); hold on;
yline(mpc_obj.OV(1).Min,'r--','Min x'); yline(mpc_obj.OV(1).Max,'r--','Max x');
plot(t_history, ones(1,Nsim)*x_eq(1), 'k--', 'LineWidth', 1);
ylabel('Cart pos x (m)'); grid on; legend('x','x_{ref}');

subplot(3,1,2);
plot(t_history, rad2deg(x_history(2,:)), 'LineWidth', 1.5); hold on;
yline(rad2deg(mpc_obj.OV(2).Min),'r--'); yline(rad2deg(mpc_obj.OV(2).Max),'r--');
plot(t_history, ones(1,Nsim)*rad2deg(x_eq(2)), 'k--', 'LineWidth', 1);
ylabel('\theta_1 (deg)'); grid on;

subplot(3,1,3);
plot(t_history, rad2deg(x_history(3,:)), 'LineWidth', 1.5); hold on;
yline(rad2deg(mpc_obj.OV(3).Min),'r--'); yline(rad2deg(mpc_obj.OV(3).Max),'r--');
plot(t_history, ones(1,Nsim)*rad2deg(x_eq(3)), 'k--', 'LineWidth', 1);
ylabel('\theta_2 (deg)'); grid on;
xlabel('Time (s)');
```




```
figure('Name','Control Effort (with MV bounds)','Color','w');
plot(t_history, u_history(1,:), 'LineWidth', 1.2); hold on;
plot(t_history, u_history(2,:), 'LineWidth', 1.2);
plot(t_history, u_history(3,:), 'LineWidth', 1.2);
yline(mpc_obj.MV(1).Min,'r--'); yline(mpc_obj.MV(1).Max,'r--');
yline(mpc_obj.MV(2).Min,'g--'); yline(mpc_obj.MV(2).Max,'g--');
yline(mpc_obj.MV(3).Min,'b--'); yline(mpc_obj.MV(3).Max,'b--');
ylabel('Control inputs'); xlabel('Time (s)');
legend('u1','u2','u3','u1 bounds','u2 bounds','u3 bounds'); grid on;
```



```
obs_poles = [0.2 0.3 0.4 0.1 0.6 0.5];
L = place(phi', C_new', obs_poles)';
fprintf('Luenberger observer poles: \n');
```

```
Luenberger observer poles:
```

```
disp(eig(phi - L*C_new));
```

```
    0.1000
    0.6000
    0.5000
    0.2000
    0.3000
    0.4000
```

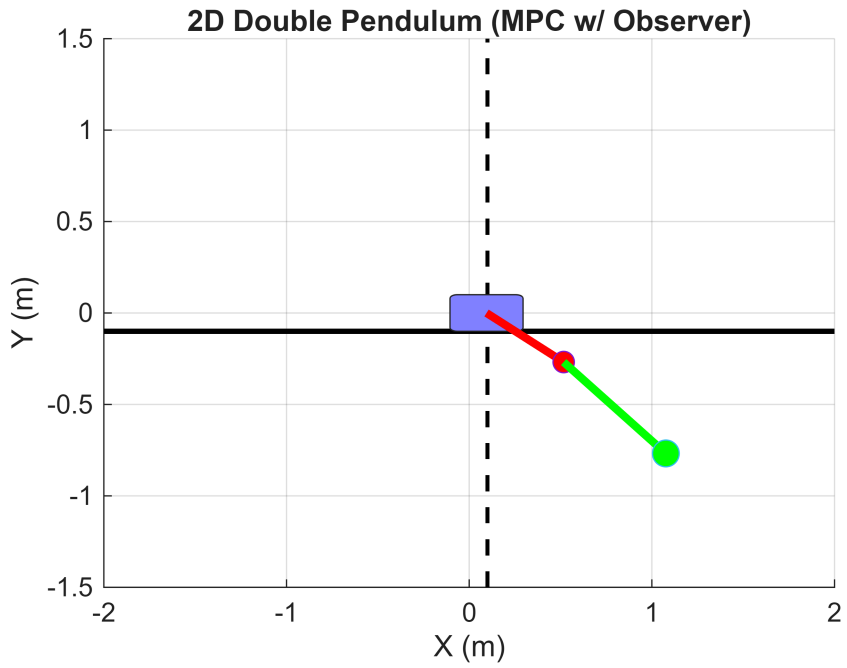
```
disp('Running MPC + Luenberger Observer simulation...');
```

```
Running MPC + Luenberger Observer simulation...
```

```
tf = 6;
t_history = 0:h:tf;
Nsim = length(t_history);

x0_true = [0; deg2rad(35); deg2rad(50); 0; 0; 0];
x0_hat = x_eq;
x_true = x0_true;
x_hat = x0_hat;

r = C_new * x_eq;

x_history = zeros(length(x_eq), Nsim);
xhat_history = zeros(length(x_eq), Nsim);
u_history = zeros(size(B_lin,2), Nsim);
x_history(:,1) = x_true;
xhat_history(:,1) = x_hat;
u_history(:,1) = u_e;

mpc_state = mpcstate(mpc_obj);
ode_opts = odeset('RelTol', 1e-6, 'AbsTol', 1e-8);

fig2D = figure('Name','2D Animation: MPC w/ Luenberger Observer','Color','w');
ax2D = axes('Parent',fig2D);
hold(ax2D,'on'); grid(ax2D,'on'); axis(ax2D,'equal');
xlabel(ax2D,'X (m)'); ylabel(ax2D,'Y (m)');
title(ax2D,'2D Double Pendulum (MPC w/ Observer)');
axis(ax2D, [-2 2 -1.5 1.5]);
plot(ax2D, [-3 3], [-cart_h/2 -cart_h/2], 'k-', 'LineWidth', 2);

plot(ax2D, [x_eq(1) x_eq(1)], [-1.5 1.5], 'k--', 'LineWidth', 1.5);

x_k = x_history(:,1);
```

```
p_cart = [x_k(1), 0];
p_m1   = [x_k(1) + l1_val*sin(x_k(2)), -l1_val*cos(x_k(2))];
p_m2   = [p_m1(1) + l2_val*sin(x_k(3)), p_m1(2) - l2_val*cos(x_k(3))];

h_cart = rectangle(ax2D, 'Position', [p_cart(1)-cart_w/2, -cart_h/2, cart_w,
cart_h], ...
                   'FaceColor',[0.5 0.5 1], 'Curvature',[0.2 0.2]);
h_line1 = line(ax2D, [p_cart(1), p_m1(1)], [p_cart(2), p_m1(2)], 'LineWidth', 3,
'Color', 'r');
h_mass1 = plot(ax2D, p_m1(1), p_m1(2), 'o', 'MarkerSize', 8, 'MarkerFaceColor','r');
h_line2 = line(ax2D, [p_m1(1), p_m2(1)], [p_m1(2), p_m2(2)], 'LineWidth', 3,
'Color', 'g');
h_mass2 = plot(ax2D, p_m2(1), p_m2(2), 'o', 'MarkerSize', 10,
'MarkerFaceColor','g');
drawnow;

for k = 1:Nsim-1
    y_meas = C_new * x_true;

    x_hat = x_hat + L * (y_meas - C_new * x_hat);
    xhat_history(:, k) = x_hat;

    mpc_state.Plant = x_hat;
    try
        u = mpcmove(mpc_obj, mpc_state, y_meas, r);
    catch ME
        warning('mpcmove threw an error at step %d: %s', k, ME.message);
        u = mpc_state.LastMove;
    end
    if isempty(u)
        u = u_e;
    end
    u_cmd = u(:);
    u_history(:, k) = u_cmd;
    odefun = @(tt,xx) f_num(tt, xx, u_cmd);
    [~, XX] = ode45(odefun, [t_history(k) t_history(k+1)], x_true, ode_opts);
    x_true = XX(end, :)';
    x_history(:, k+1) = x_true;

    x_hat = phi * x_hat + gamma * u_cmd;
    xhat_history(:, k+1) = x_hat;
    p_cart = [x_true(1), 0];
    p_m1   = [x_true(1) + l1_val*sin(x_true(2)), -l1_val*cos(x_true(2))];
    p_m2   = [p_m1(1) + l2_val*sin(x_true(3)), p_m1(2) - l2_val*cos(x_true(3))];

    set(h_cart, 'Position', [p_cart(1)-cart_w/2, -cart_h/2, cart_w, cart_h]);
    set(h_line1, 'XData', [p_cart(1), p_m1(1)], 'YData', [p_cart(2), p_m1(2)]);
    set(h_mass1, 'XData', p_m1(1), 'YData', p_m1(2));
    set(h_line2, 'XData', [p_m1(1), p_m2(1)], 'YData', [p_m1(2), p_m2(2)]);
    set(h_mass2, 'XData', p_m2(1), 'YData', p_m2(2));
```

```
    drawnow limitrate;
end
```



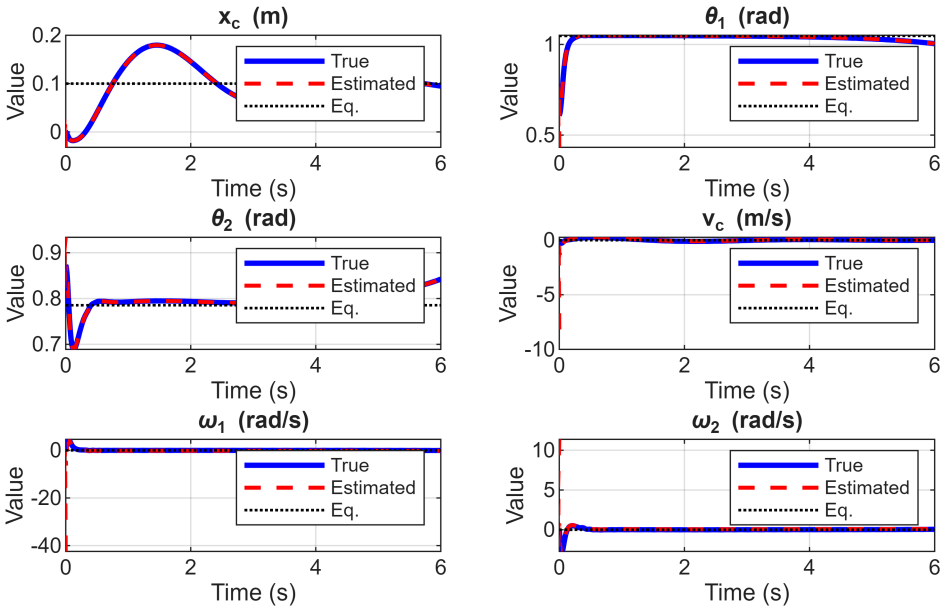

```
u_history(:, Nsim) = u_history(:, Nsim-1);
disp('Simulation complete.');
```

```
Simulation complete.
```

```
disp('Plotting simulation results...');
```

```
Plotting simulation results...
```

```
t_plot = t_history;

figure('Name', 'Observer Performance (True vs. Estimated States)');
state_names = {'x_c (m)', '\theta_1 (rad)', '\theta_2 (rad)', 'v_c (m/s)',
'\omega_1 (rad/s)', '\omega_2 (rad/s)'};
for i = 1:6
    subplot(3, 2, i);
    plot(t_plot, x_history(i,:), 'b-', 'LineWidth', 2); hold on;
    plot(t_plot, xhat_history(i,:), 'r--', 'LineWidth', 1.5);
    plot(t_plot, x_eq(i)*ones(size(t_plot)), 'k:', 'LineWidth', 1);
    title(state_names{i});
    ylabel('Value');
    xlabel('Time (s)');
    legend('True', 'Estimated', 'Eq.');
    grid on;
end
```


x_c (m)
θ_1 (rad)
θ_2 (rad)
v_c (m/s)
ω_1 (rad/s)
ω_2 (rad/s)
Value
Time (s)
True
Estimated
Eq.