```
syms x1 x2 x3 x4 x5 x6 u
```

```
Warning: Class 'sym' is defined in a class folder and takes precedence over a function with the same name
that is earlier on the MATLAB path. In a future release, class 'sym' will no longer be given precedence.

Click here for the locations of the conflicting items.
Click here for guidelines to avoid this warning.
```

```
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 =

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

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 =

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

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 =

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

Funwork 3

Instructions

The objective of this assignment is to apply Linear Matrix Inequalities (LMI) to the design of linear controllers stabilizing a nonlinear system about a nonzero equilibrium state. Your controllers are to be tested on the nonlinear model.

Problem 1. (4 pts) Start with the non-linear model of the double inverted pendulum on a cart (DIPC) from FunWork 1. Show that there does not exist u_e that would make

$$x_e = \begin{bmatrix} 0.1 & 60^\circ & 45^\circ & 0 & 0 & 0 \end{bmatrix}^\top$$

an equilibrium state.

Intuitively we can say the system won't be stable as only one control input is provided and there are three degrees of freedom which are initialized. I will show this by considering the equation for qdd which says that:

$\mathrm{qdd} = D(q)^{-1}(H.u - C.\mathrm{qd} - G(q))$

For equilibrium for some U_e and X_e we have:

$\mathrm{qdd} = 0$ As at equilibrium, all derivates should be 0

qd = 0 (Given)

$D(q)^{-1}(H.u_e - C.(0) - G(q)) = 0$

D(q) is invertible and also at the given states it is a non zero matrix. So to find the solution of the equation is to find solution for:

$H.u_e - G = 0$

```
syms u_e 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*u_e == G_val;
sol_u = solve(eq, u_e, 'Real', true)
```

sol_u =

```
Empty sym: 0-by-1
```

There is no solution for U that will take the state to equilibrium

Problem 2. (4 pts) Add an extra input in the non-linear DIPC model, namely, the torque at the first joint. Thus the system's in

$$u = \begin{bmatrix} u_1 & u_2 \end{bmatrix}^\top,$$

where u_1 is the force applied to the cart and u_2 is the torque applied at the first joint. In summary, we now have a three-c two-input system.

Show that there is no $u_e = \begin{bmatrix} u_{1e} & u_{2e} \end{bmatrix}^\top$ that would make $x_e = \begin{bmatrix} 0.1 & 60^\circ & 45^\circ & 0 & 0 & 0 \end{bmatrix}^\top$ ar equilibrium state.

```
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 =
```

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

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]
```

Problem 3. (4 pts) Add the third extra input in the non-linear DIPC model, namely, the torque at the second joint. Thus the system's input is

$$u = [u_1 \quad u_2 \quad u_3]^\top,$$

where u_1 is the force applied to the cart, u_2 is the torque applied at the first joint, and u_3 is the torque applied at the second joint. In summary, we now have a three-output three-input system.

Find $u_e = [u_{1e} \quad u_{2e} \quad u_{3e}]^\top$ that makes

$$x_e = [0.1 \quad 60^\circ \quad 45^\circ \quad 0 \quad 0 \quad 0]^\top$$

an equilibrium state.

```
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 =
```

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

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
```

Problem 4. (4 pts) Perform Taylor's linearization about (x_e, u_e).

```
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
```

Problem 5. (8 pts) Design a state-feedback controller, $u = -K_x x$, using LMIs and test it on the non-linear model.

Generate plots of the state variables versus time on the time interval $[0,\ 3]$ secs. When performing your simulations you h use one of MATLAB's *ode* functions, for example, *ode23* or *ode45*. Compare their performance and see if you can notice any differences in your plots.

$x_{\mathrm{dot}} = \mathrm{Ax} + \mathrm{Bu}$

$x_{\mathrm{dot}} = \mathrm{Ax} + B(-K_x x)$

$x_{\mathrm{dot}} = (A - \mathrm{BK}_x)x$

We need to create lyapunov equation with this and then convert it into a format that can be solved using LMI

$(A - \mathrm{BK})^T P + P(A - \mathrm{BK}) < 0,\ P > 0$

$S = P^{-1}$

$Z = \mathrm{KS}$

$\mathrm{SA}^T + \mathrm{AS} - Z^T B^T - \mathrm{BZ} < 0,\ P > 0$

We need to use the linearized matrices from the previous part

We need to solve the above equation as our LMI to get the state feedback controller design n = 6 (size of A is (6,6))

m = 3 (size of B is (6,3))

```
cvx_begin sdp
n = 6
```

```
n =
6
```

```
m = 3
```

```
m =
3
```

```
ep = 1e-6
```

```
ep =
1.0000e-06
```

```
variable S(n,n) symmetric
variable Z(m,n)
A_lin_syms_new*S + S*A_lin_syms_new' - Z'*B_lin_syms_new' - B_lin_syms_new*Z <=
-ep*eye(n)
S >= ep*eye(n)
cvx_end
```

```
Calling SDPT3 4.0: 60 variables, 21 equality constraints
------------------------------------------------------------

 num. of constraints = 21
 dim. of sdp    var  = 12,   num. of sdp  blk  =  2
 dim. of free   var  = 18 *** convert ublk to lblk
*******************************************************************
   SDPT3: Infeasible path-following algorithms
*******************************************************************
 version  predcorr  gam  expon  scale_data
   HKM      1      0.000   1        0
it pstep dstep pinfeas dinfeas  gap      prim-obj      dual-obj    cputime
-------------------------------------------------------------------
 0|0.000|0.000|5.2e+02|9.2e+02|2.0e+05| 0.000000e+00  0.000000e+00| 0:0:00| chol  1  1
 1|1.000|0.884|6.0e-04|1.1e+02|5.6e+03| 0.000000e+00  1.058790e-04| 0:0:00| chol  1  1
 2|1.000|0.988|2.0e-04|1.3e+00|2.4e+01| 0.000000e+00  1.260239e-06| 0:0:01| chol  1  1
 3|1.000|0.989|5.0e-06|1.5e-02|1.8e-01| 0.000000e+00  1.442981e-08| 0:0:01| chol  1  1
 4|1.000|0.990|1.2e-06|2.5e-04|3.6e-03| 0.000000e+00  2.250779e-10| 0:0:01| chol  1  1
 5|1.000|1.000|5.4e-08|2.0e-04|5.7e-04| 0.000000e+00  7.070191e-12| 0:0:01| chol  1  1
 6|1.000|0.978|1.3e-09|3.2e-05|8.2e-05| 0.000000e+00  1.066474e-12| 0:0:01| chol  1  1
 7|1.000|0.989|9.7e-11|4.6e-06|1.0e-05| 0.000000e+00  1.182627e-14| 0:0:01| chol  1  1
 8|1.000|0.962|3.1e-13|5.7e-07|1.3e-06| 0.000000e+00  8.990980e-15| 0:0:01| chol  1  1
 9|1.000|0.828|3.0e-13|7.1e-08|1.6e-07| 0.000000e+00  2.503882e-15| 0:0:01| chol  1  1
10|1.000|0.863|2.0e-12|8.9e-09|2.1e-08| 0.000000e+00  5.397701e-16| 0:0:01| chol  1  1
11|1.000|0.890|4.9e-13|1.2e-09|2.8e-09| 0.000000e+00  8.372175e-17| 0:0:01|
  stop: max(relative gap, infeasibilities) < 1.49e-08
-------------------------------------------------------------------
 number of iterations   = 11
 primal objective value =  0.00000000e+00
 dual   objective value =  8.37217523e-17
 gap := trace(XZ)       = 2.76e-09
 relative gap           = 2.76e-09
```

```
 actual relative gap    = -8.37e-17
 rel. primal infeas (scaled problem)   = 4.94e-13
 rel. dual     "        "       "      = 1.16e-09
 rel. primal infeas (unscaled problem) = 0.00e+00
 rel. dual     "        "       "      = 0.00e+00
 norm(X), norm(y), norm(Z) = 7.1e+01, 2.3e-11, 7.8e-10
 norm(A), norm(b), norm(C) = 9.4e+01, 1.0e+00, 1.0e+00
 Total CPU time (secs)  = 0.77
 CPU time per iteration = 0.07
 termination code       =  0
 DIMACS: 4.9e-13  0.0e+00  1.2e-09  0.0e+00  -8.4e-17  2.8e-09
-------------------------------------------------------------------

------------------------------------------------------------
Status: Solved
Optimal value (cvx_optval): +0
```

```
K_cvx = Z/S
```

```
K_cvx = 3×6
    0.5059    0.7096    1.4190    1.9847    0.5756    0.7611
    0.0858    5.2626    2.4848    0.4269    1.1523    1.1960
    0.1171    1.5430    8.7194    0.5831    1.2254    1.8557
```

Controller gain matrix K has been computed using the CVX solver. I will proceed to plot the simulation graphs.

```
A_cvx = A_lin_syms_new-B_lin_syms_new*K_cvx
```

```
A_cvx = 6×6
         0         0         0    1.0000         0         0
         0         0         0         0    1.0000         0
         0         0         0         0         0    1.0000
   -0.1664    0.5265    1.2203   -0.6012    0.3097    0.4205
   -0.0074   -8.9635    3.9880   -0.1271   -2.7669   -0.1801
   -0.1158    1.6182  -15.1377   -0.7336   -1.4150   -4.6790
```

```
%Checking the poles of the new system
poles = eig(A_cvx);
disp('Poles of the new system:');
```

```
Poles of the new system:
```

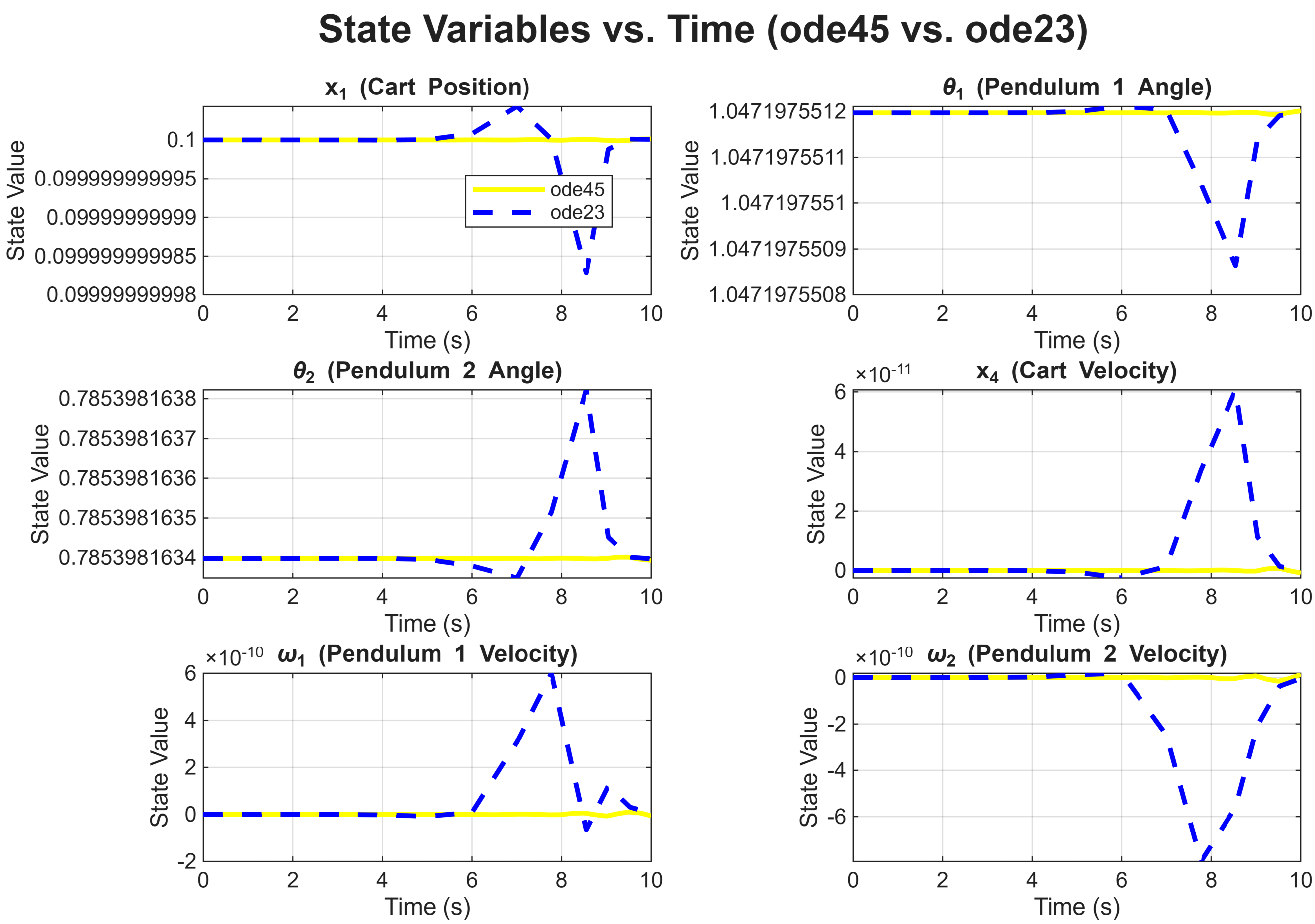
```
disp(poles);
```

```
  -0.3433 + 0.2471i
  -0.3433 - 0.2471i
  -1.5654 + 3.4358i
  -1.5654 - 3.4358i
  -2.1149 + 2.1402i
  -2.1149 - 2.1402i
```

Poles are all in open left hand plane. It is a stable design

```

% Define the state variables for the function handles
X_vars = [x1; x2; x3; x4; x5; x6];

% Create the required .m files in your current folder
matlabFunction(D, 'File', 'D_func', 'Vars', {X_vars.'}, 'Optimize', false);
```

```
matlabFunction(C, 'File', 'C_func', 'Vars', {X_vars.'}, 'Optimize', false);
matlabFunction(G, 'File', 'G_func', 'Vars', {X_vars.'}, 'Optimize', false);

tspannew = [0 10];
x0 = [0.1; deg2rad(60); deg2rad(45); 0; 0; 0]; % Initial conditions

odefun = @(t,x) nonlinear_closed_loop(t, x, K_cvx, u_e_new', x0);

opts = odeset('RelTol', 1e-6, 'AbsTol', 1e-9);
[t45, x45] = ode45(odefun, tspannew, x0, opts);
[t23, x23] = ode23(odefun, tspannew, x0, opts);

% --- Plotting Results ---
% --- Plotting Results ---
figure('Position', [100, 100, 1200, 800]);

sgtitle('State Variables vs. Time (ode45 vs. ode23)', 'FontSize', 16, 'FontWeight',
'bold');

state_names = {'x_1 (Cart Position)', '\theta_1 (Pendulum 1 Angle)', '\theta_2
(Pendulum 2 Angle)', ...
               'x_4 (Cart Velocity)', '\omega_1 (Pendulum 1 Velocity)', '\omega_2
(Pendulum 2 Velocity)'};

for i = 1:6
    subplot(3, 2, i);
    % Changed plot colors to yellow ('y-') and blue ('b--')
    plot(t45, x45(:,i), 'y-', 'LineWidth', 2);
    hold on;
    plot(t23, x23(:,i), 'b--', 'LineWidth', 2);
    grid on;
    title(state_names{i});
    xlabel('Time (s)');
    ylabel('State Value');
    if i == 1
        legend('ode45', 'ode23', 'Location', 'best');
    end
end
```




```
function dx = nonlinear_closed_loop(t, x, K, u_e, x_e)

    % Decompose state vector
    q   = x(1:3);
    qd  = x(4:6);

    % Evaluate dynamics matrices at the current state.
    % These call the .m files we generated with matlabFunction.
    D_mat = D_func(x.'); % Pass state as a row vector
    C_mat = C_func(x.');
    G_vec = G_func(x.');

    % Calculate the state-feedback control input
    u = u_e-K * (x-x_e); % u is 3x1

    % Input matrix H is the identity matrix
    H_mat = eye(3);

    qdd = D_mat \ (H_mat * u - C_mat * qd - G_vec); % Ensure G is a column vector

    % Return the state derivative vector
    dx = [qd; qdd];
```

```
end
```

Problem 6. (4 pts) Use LMIs to design an output-feedback controller, $u = -K_o y$. Generate plots of state variables versus time on the time interval $[0,\ 3]$ secs. This can be one figure with subplots.

$x_{\text{dot}} = \text{Ax} + \text{Bu}$

$y = \text{Cx}$

$u = -K_o y$

$u = -K_o\text{Cx}$

$x_{\text{dot}} = \text{Ax} + B(-K_o\text{Cx})$

$x_{\text{dot}} = (A - \text{BK}_o C)x$

$(A - \text{BKC})^T P + P(A - \text{BKC}) < 0, P > 0$

$A^T P + \text{PA} - C^T K^T \text{MB}^T - \text{BMKC} < 0$

$A^T P + \text{PA} - C^T N^T B^T - \text{BNC} < 0, \text{BM} = \text{PB}, P > 0$

$K = M^{-1} N$

```
p = size(C,1);
C_new = [1 0 0 0 0 0;
    0 1 0 0 0 0;
    0 0 1 0 0 0];
cvx_begin sdp quiet
variable P(n,n) symmetric
variable N(m,p)
variable M(m,m)
P*A_lin_syms_new+A_lin_syms_new'*P-C_new'*N'*B_lin_syms_new'-
B_lin_syms_new*N*C_new<-ep*eye(n)
```

```
Warning: The use of strict inequalities in CVX is strongly discouraged,
    because solvers treat them as non-strict inequalities. Please
    consider using "<=" instead.
```

```
B_lin_syms_new*M == P*B_lin_syms_new
P >= ep*eye(n);
cvx_end
K_o = M\N
```

```
K_o = 3×3
    1.1331    0.3146    0.4202
   -0.0227    3.9816    0.8455
   -0.0213    0.7901    5.2143
```

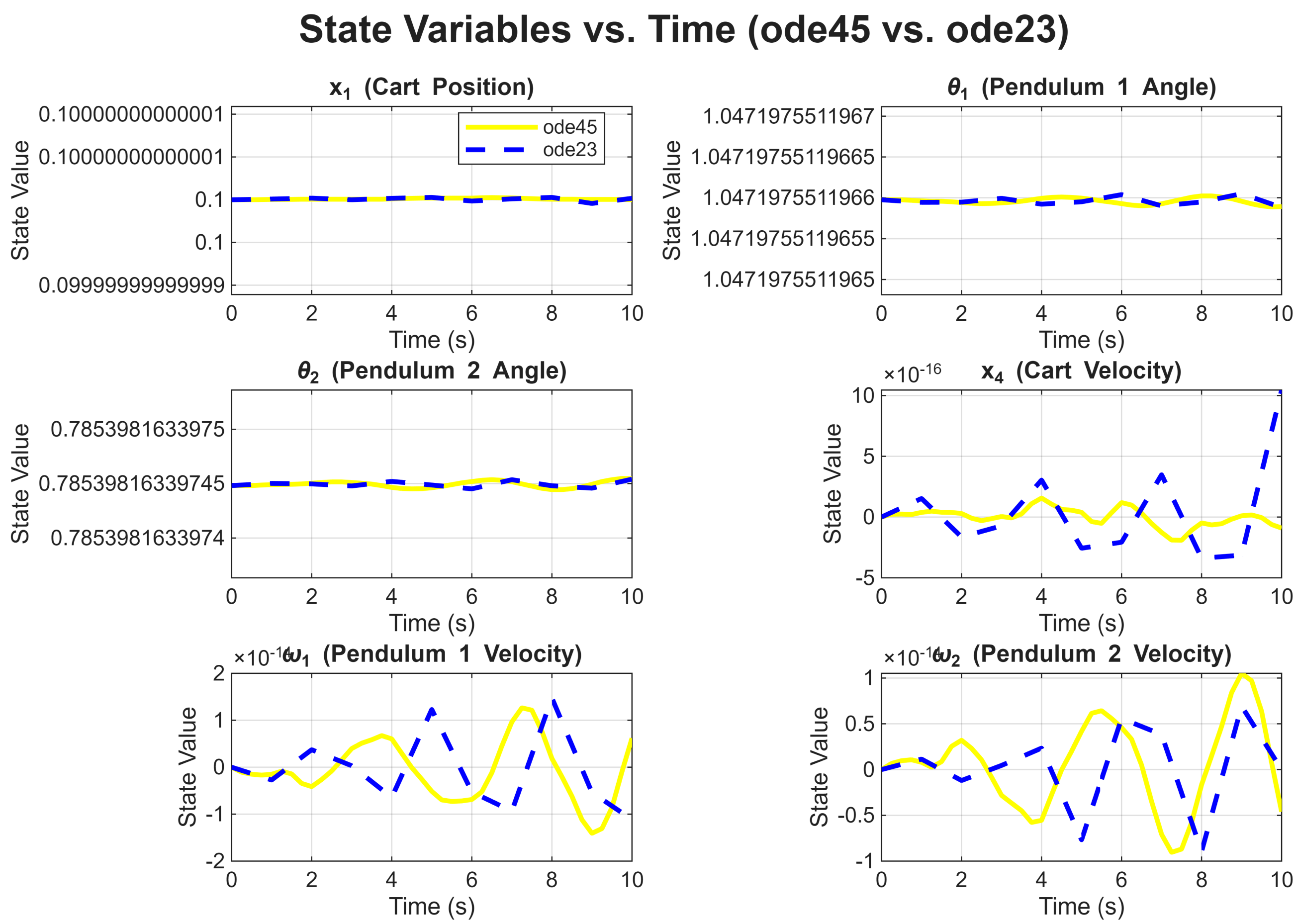
```
A_o = A_lin_syms_new - B_lin_syms_new*K_o*C_new;
eig(A_o)
```

```
ans = 6×1 complex
   0.0000 + 0.6546i
```

```
    0.0000 - 0.6546i
    0.0000 + 1.8611i
    0.0000 - 1.8611i
   -0.0000 + 1.7345i
   -0.0000 - 1.7345i
```

$n \leq m + p - 1$ (where n = 6, m = 3, p = 3) is not satisfied in this case and we can see that the system is not stable in this case.

```
odefun2 = @(t,x) nonlinear_closed_loop2(t, x, K_o, u_e_new', x0,C_new);

opts = odeset('RelTol', 1e-6, 'AbsTol', 1e-9);
[t45_o, x45_o] = ode45(odefun2, tspannew, x0, opts);
[t23_o, x23_o] = ode23(odefun2, tspannew, x0, opts);
figure('Position', [100, 100, 1200, 800]);

sgtitle('State Variables vs. Time (ode45 vs. ode23)', 'FontSize', 16, 'FontWeight',
'bold');

state_names = {'x_1 (Cart Position)', '\theta_1 (Pendulum 1 Angle)', '\theta_2
(Pendulum 2 Angle)', ...
               'x_4 (Cart Velocity)', '\omega_1 (Pendulum 1 Velocity)', '\omega_2
(Pendulum 2 Velocity)'};

for i = 1:6
    subplot(3, 2, i);
    plot(t45_o, x45_o(:,i), 'y-', 'LineWidth', 2);
    hold on;
    plot(t23_o, x23_o(:,i), 'b--', 'LineWidth', 2);
    grid on;
    title(state_names{i});
    xlabel('Time (s)');
    ylabel('State Value');
    if i == 1
        legend('ode45', 'ode23', 'Location', 'best');
    end
end
```




```
function dx = nonlinear_closed_loop2(t, x, K, u_e, x_e,C_k)


    q   = x(1:3);
    qd  = x(4:6);

    D_mat = D_func(x.');
    C_mat = C_func(x.');
    G_vec = G_func(x.');

    u = u_e-K*C_k*(x-x_e);

    H_mat = eye(3);

    qdd = D_mat \ (H_mat * u - C_mat * qd - G_vec);

    dx = [qd; qdd];
end
```

Problem 7. (4 pts) Simulate the combined state-feedback controller-observer compensator and compare its performance against the output feedback controller.

Generate plots of state variables versus time, where $\tilde{x}_i$ is an estimate of x_i. Note that you can always set your observer's initial conditions to zero, if you wish, since you have complete access to your design.

For combined state feedback controller-observer compensator we have:

We already checked the eigenvalues of A-BK which were in open left hand plane, for the observer we have:

$$(A - \mathrm{LC})^T P + P(A - \mathrm{LC}) < 0$$
$$P = P^T, P > 0$$
$$A^T P + \mathrm{PA} - C^T L^T P - \mathrm{PLC} < 0$$
$$A^T P + \mathrm{PA} - C^T Y^T - \mathrm{YC} + 2\alpha P \leq 0, P > 0$$
$$L = P^{-1} Y$$

```
cvx_begin sdp
variable P(n,n) symmetric
variable Y(n,p)
P*A_lin_syms_new + A_lin_syms_new'*P - C_new'*Y'-Y*C_new + P <= 0
P >= eps*eye(n)
cvx_end
```

```
Calling SDPT3 4.0: 56 variables, 17 equality constraints
------------------------------------------------------------

 num. of constraints = 17
 dim. of sdp    var  = 12,   num. of sdp  blk  =  2
 dim. of free   var  = 14 *** convert ublk to lblk
*******************************************************************
   SDPT3: Infeasible path-following algorithms
*******************************************************************
 version  predcorr  gam  expon  scale_data
   HKM      1      0.000   1        0
it pstep dstep pinfeas dinfeas  gap      prim-obj      dual-obj    cputime
-------------------------------------------------------------------
 0|0.000|0.000|4.0e+02|1.8e+02|2.5e+04| 0.000000e+00  0.000000e+00| 0:0:00| chol  1  1
 1|0.961|0.844|1.6e+01|2.8e+01|1.2e+03| 0.000000e+00  1.688997e-14| 0:0:00| chol  1  1
 2|1.000|0.896|1.5e-05|3.0e+00|7.3e+01| 0.000000e+00  6.705122e-15| 0:0:00| chol  1  1
 3|1.000|0.988|6.6e-06|3.5e-02|7.3e-01| 0.000000e+00  8.077666e-17| 0:0:00| chol  1  1
 4|1.000|0.990|9.2e-07|4.6e-04|9.4e-03| 0.000000e+00  1.099275e-18| 0:0:00| chol  1  1
 5|1.000|0.998|4.5e-08|1.1e-05|2.2e-04| 0.000000e+00  2.790843e-20| 0:0:00| chol  1  1
 6|1.000|1.000|2.0e-09|1.3e-05|4.3e-05| 0.000000e+00  2.544383e-21| 0:0:00| chol  1  1
 7|1.000|0.989|1.0e-10|2.5e-06|5.6e-06| 0.000000e+00  2.773511e-23| 0:0:00| chol  1  1
 8|1.000|0.969|4.9e-13|3.3e-07|7.3e-07| 0.000000e+00  3.683199e-24| 0:0:00| chol  1  1
 9|1.000|0.774|1.0e-13|4.2e-08|9.6e-08| 0.000000e+00  1.067711e-24| 0:0:00| chol  1  1
10|1.000|0.846|4.4e-13|5.6e-09|1.3e-08| 0.000000e+00  2.695667e-25| 0:0:00|
  stop: max(relative gap, infeasibilities) < 1.49e-08
-------------------------------------------------------------------
 number of iterations   = 10
 primal objective value =  0.00000000e+00
 dual   objective value =  2.69566748e-25
 gap := trace(XZ)       = 1.32e-08
```

```
  relative gap           = 1.32e-08
  actual relative gap    = -2.70e-25
  rel. primal infeas (scaled problem)   = 4.35e-13
  rel. dual     "        "       "      = 5.60e-09
  rel. primal infeas (unscaled problem) = 0.00e+00
  rel. dual     "        "       "      = 0.00e+00
  norm(X), norm(y), norm(Z) = 6.4e+01, 1.2e-10, 4.0e-09
  norm(A), norm(b), norm(C) = 8.8e+01, 1.0e+00, 1.0e+00
  Total CPU time (secs)  = 0.20
  CPU time per iteration = 0.02
  termination code       =  0
  DIMACS: 4.4e-13  0.0e+00  5.6e-09  0.0e+00  -2.7e-25  1.3e-08
-------------------------------------------------------------------

------------------------------------------------------------
Status: Solved
Optimal value (cvx_optval): +0
```

```
L = P\Y
```

```
L = 6×3
    2.0286   -0.0808   -0.0754
   -0.1215    4.7416   -1.6842
   -0.2884   -1.3439    4.4698
    2.8472   -0.4678   -1.9145
    0.2812   52.9516  -40.3881
   -1.9718  -36.9974   47.1164
```

```
A_l = A_lin_syms_new - L*C_new;
eigenvalues = eig(A_l)
```

```
eigenvalues = 6×1 complex
  -3.0645 + 6.4422i
  -3.0645 - 6.4422i
  -1.0264 + 1.2420i
  -1.0264 - 1.2420i
  -1.5291 + 1.6701i
  -1.5291 - 1.6701i
```

Eigenvalues of the observer are in the open left hand plane.

We have

$\delta x_{\mathrm{dot}} = (A - \mathrm{LC})\delta x + B\delta u$

$\delta y = C\delta x + D\delta u,$

For combined observer controller compensator :

$z_{\mathrm{dot}} = \mathrm{Az} + B\delta u + L(\delta y - \delta y_{\mathrm{est}})$

$\delta y_{\mathrm{est}} = \mathrm{Cz} + D\delta u$

$\delta u = -\mathrm{Kz} + v(\text{will set } v \text{ to } 0)$

where z is estimator of x or x_{est}

$(\delta y - \delta y_{\mathrm{est}}) = C(\delta x - z)$

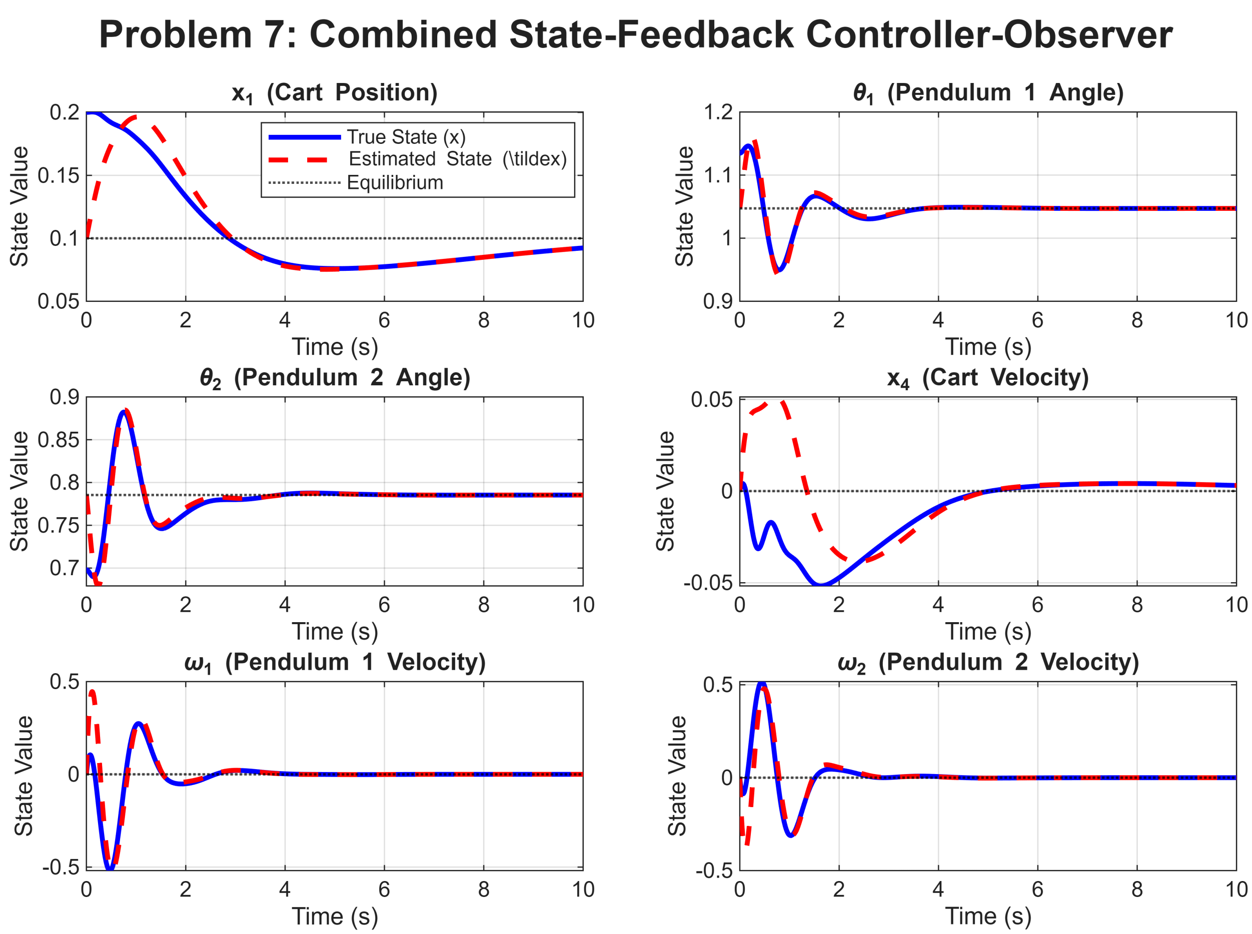
$z_{\mathrm{dot}} = (A - \mathrm{BK} - \mathrm{LC})z + \mathrm{LC}\delta x$

Here K is out K_cvx which we calculated for Problem 5, we calculated L above.

Let's combine them for Combined Observer-Controller Compensator

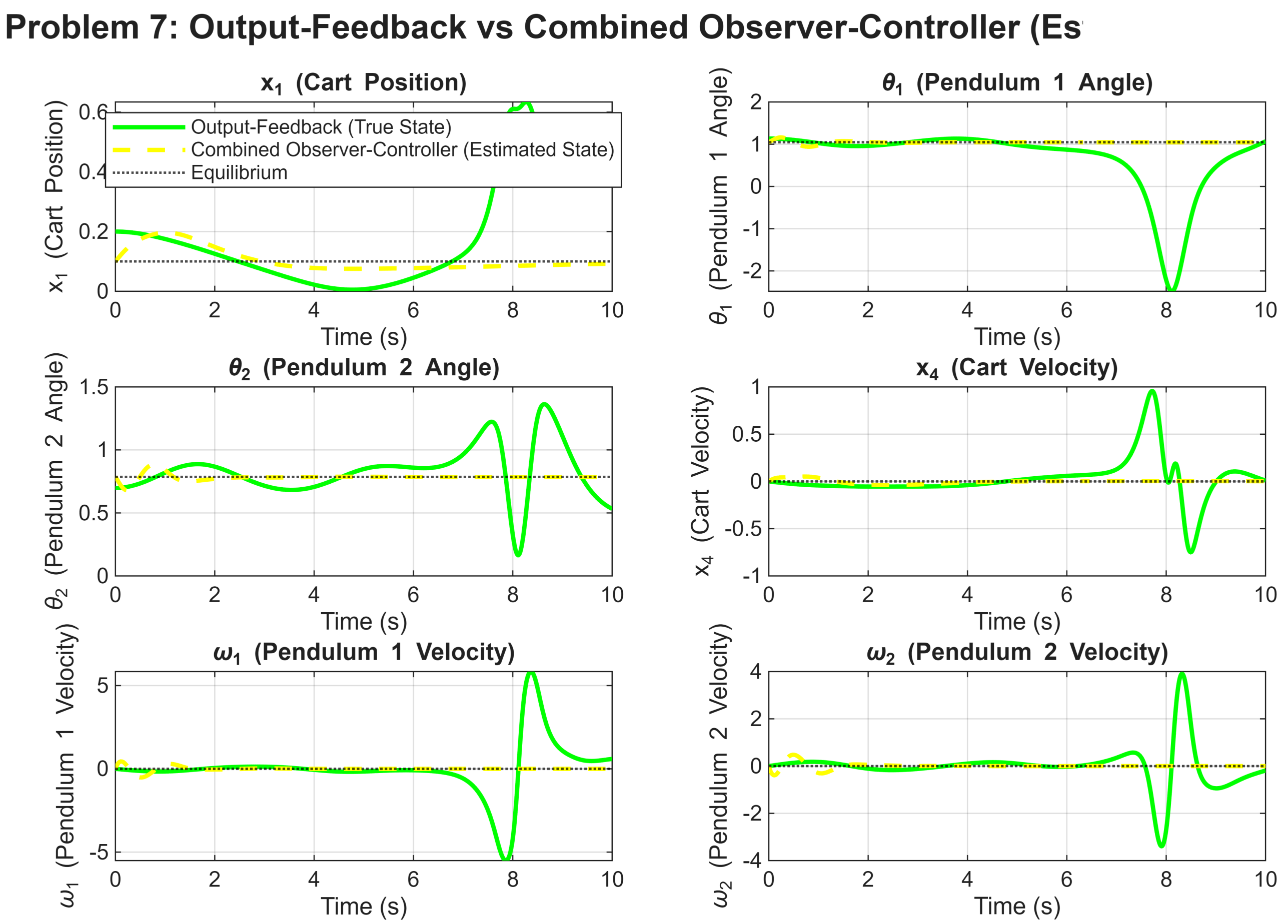
```
function dx_aug_dt = nonlinear_combined_loop(t, x_aug, K, L, A, B, C, x_e, u_e)
    x_real = x_aug(1:6);
    z = x_aug(7:12);
    u_dev = -K*z;
    u = u_e - K*z;
    q_real = x_real(1:3);
    qd_real = x_real(4:6);
    D_mat = D_func(x_real.');
    C_mat = C_func(x_real.');
    G_vec = G_func(x_real.');
    H_mat = eye(3);
    qdd_real = D_mat\(H_mat*u - C_mat*qd_real - G_vec);
    dx_real_dt = [qd_real; qdd_real];
    y_dev_real = C*(x_real-x_e);
    y_dev_est = C*z;
    dz_dt = A*z + B*u_dev + L * (y_dev_real - y_dev_est);
    dx_aug_dt = [dx_real_dt; dz_dt];
end

tspannew = [0 10];

x_e = [0.1; deg2rad(60); deg2rad(45); 0; 0; 0];
u_e = [0; -5.3098; -3.9019];

x0_real = [0.2; deg2rad(65); deg2rad(40); 0; 0; 0];

z0 = zeros(6,1);

x_aug_0 = [x0_real; z0];

% --- Run the Simulation ---
opts = odeset('RelTol', 1e-6, 'AbsTol', 1e-9);
odefun_obs = @(t,x_aug) nonlinear_combined_loop(t, x_aug, K_cvx, L, ...
                                 A_lin_syms_new, B_lin_syms_new, C_new, x_e, u_e);

[t_obs, x_obs_aug] = ode45(odefun_obs, tspannew, x_aug_0, opts);

x_real = x_obs_aug(:, 1:6);
z  = x_obs_aug(:, 7:12);
x_est  = z + x_e'; % Add x_e to each row

% --- Plotting Results for Problem 7 ---
figure('Position', [100, 100, 1200, 800]);
sgtitle('Problem 7: Combined State-Feedback Controller-Observer', 'FontSize', 16,
'FontWeight', 'bold');

state_names = {'x_1 (Cart Position)', '\theta_1 (Pendulum 1 Angle)', '\theta_2
(Pendulum 2 Angle)', ...
               'x_4 (Cart Velocity)', '\omega_1 (Pendulum 1 Velocity)', '\omega_2
(Pendulum 2 Velocity)'};
```

```
for i = 1:6
    subplot(3, 2, i);
    plot(t_obs, x_real(:,i), 'b-', 'LineWidth', 2);
    hold on;
    plot(t_obs, x_est(:,i), 'r--', 'LineWidth', 2);

    % Plot the equilibrium line
    yline(x_e(i), 'k:', 'LineWidth', 1);

    grid on;
    title(state_names{i});
    xlabel('Time (s)');
    ylabel('State Value');
    if i == 1
        legend('True State (x)', 'Estimated State (\tilde{x})', 'Equilibrium',
'Location', 'best');
    end
end
```



Now, checking the output-feedback vs combined state feedback controller-observer

```
figure('Position',[100 100 1200 800]);
sgtitle('Problem 7: Output-Feedback vs Combined Observer-Controller (Estimated
States)', ...
    'FontSize',14,'FontWeight','bold');

state_names = {'x_1 (Cart Position)', '\theta_1 (Pendulum 1 Angle)', '\theta_2
(Pendulum 2 Angle)', ...
               'x_4 (Cart Velocity)', '\omega_1 (Pendulum 1 Velocity)', '\omega_2
(Pendulum 2 Velocity)'};
[t45_oo, x45_oo] = ode45(odefun2, tspannew, x0_real, opts);

for i = 1:6
    subplot(3,2,i);

    % True state from output-feedback controller
    plot(t45_oo, x45_oo(:,i), 'g-', 'LineWidth', 1.8); hold on;

    % Estimated state from combined observer-controller
    plot(t_obs, x_est(:,i), 'y--', 'LineWidth', 1.8);

    % Equilibrium reference line
    yline(x_e(i), 'k:', 'LineWidth', 1);

    grid on;
    xlabel('Time (s)');
    ylabel(state_names{i});
    title(state_names{i});

    if i == 1
        legend('Output-Feedback (True State)', ...
               'Combined Observer-Controller (Estimated State)', ...
               'Equilibrium', 'Location', 'best');
    end
end
```



We can clearly see that the output feedback has failed to stabilitze. This was anyways bound to happen because the poles were placed at the imaginary axis. But we can see that in long run combined observer controller does better.

For problem 8:

We need our observer poles atleast 3-4 times behind hte controller poles in the open left hand plane, let's try to redesign by choosing sone value of α that will give us sufficient value for it.

```
alpha = 4
```

```
alpha =
4
```

```
cvx_begin sdp
variable P_new(n,n) symmetric
variable Y_new(n,p)
P_new*A_lin_syms_new + A_lin_syms_new'*P_new - C_new'*Y_new'-Y_new*C_new +
2*alpha*P_new <= 0
P_new >= eps*eye(n)
cvx_end
```

```
Calling SDPT3 4.0: 56 variables, 17 equality constraints
------------------------------------------------------------

 num. of constraints = 17
 dim. of sdp    var  = 12,   num. of sdp  blk  =  2
 dim. of free   var  = 14 *** convert ublk to lblk
*******************************************************************
   SDPT3: Infeasible path-following algorithms
*******************************************************************
 version  predcorr  gam  expon  scale_data
   HKM      1      0.000   1        0
it pstep dstep pinfeas dinfeas  gap      prim-obj      dual-obj    cputime
-------------------------------------------------------------------
 0|0.000|0.000|4.5e+02|1.8e+02|2.5e+04| 0.000000e+00  0.000000e+00| 0:0:00| chol  1  1
 1|0.931|0.941|3.1e+01|1.1e+01|5.2e+02| 0.000000e+00  2.876690e-14| 0:0:00| chol  1  1
 2|1.000|0.794|5.0e-05|2.2e+00|5.4e+01| 0.000000e+00  2.517729e-14| 0:0:00| chol  1  1
 3|1.000|0.988|9.5e-06|2.8e-02|5.8e-01| 0.000000e+00  3.177904e-16| 0:0:00| chol  1  1
 4|1.000|0.992|1.6e-06|3.3e-04|6.7e-03| 0.000000e+00  3.615910e-18| 0:0:00| chol  1  1
 5|1.000|1.000|5.5e-07|1.0e-05|2.0e-04| 0.000000e+00  9.691114e-20| 0:0:00| chol  1  1
 6|1.000|1.000|1.2e-08|1.1e-05|3.6e-05| 0.000000e+00  9.387425e-21| 0:0:00| chol  1  1
 7|1.000|0.987|6.5e-10|2.0e-06|4.2e-06| 0.000000e+00  2.047189e-22| 0:0:00| chol  1  1
 8|1.000|0.973|8.2e-12|2.4e-07|4.7e-07| 0.000000e+00  2.087649e-23| 0:0:00| chol  1  1
 9|1.000|0.974|5.3e-13|2.7e-08|5.3e-08| 0.000000e+00  2.385924e-24| 0:0:00| chol  1  1
10|1.000|0.854|9.1e-14|3.0e-09|6.1e-09| 0.000000e+00  5.052433e-25| 0:0:00|
  stop: max(relative gap, infeasibilities) < 1.49e-08
-------------------------------------------------------------------
 number of iterations   = 10
 primal objective value =  0.00000000e+00
 dual   objective value =  5.05243336e-25
 gap := trace(XZ)       = 6.12e-09
 relative gap           = 6.12e-09
 actual relative gap    = -5.05e-25
 rel. primal infeas (scaled problem)   = 9.13e-14
 rel. dual     "        "       "      = 3.00e-09
 rel. primal infeas (unscaled problem) = 0.00e+00
 rel. dual     "        "       "      = 0.00e+00
 norm(X), norm(y), norm(Z) = 5.3e+01, 1.9e-10, 2.5e-09
 norm(A), norm(b), norm(C) = 9.2e+01, 1.0e+00, 1.0e+00
 Total CPU time (secs)  = 0.22
 CPU time per iteration = 0.02
 termination code       =  0
 DIMACS: 9.1e-14  0.0e+00  3.0e-09  0.0e+00  -5.1e-25  6.1e-09
-------------------------------------------------------------------

------------------------------------------------------------
Status: Solved
Optimal value (cvx_optval): +0
```

```
L_new = P_new\Y_new
```

```
L_new = 6×3
   11.6989   -0.1113   -0.1233
   -0.8345   17.1915    1.3549
   -1.5255    0.4878   18.0611
   60.0531   -0.8879   -2.4854
    2.9333  104.5949  -25.1762
   -3.2046  -21.8558  104.2521
```

```
eig(A_lin_syms_new-L_new*C_new)
```

```
ans = 6×1 complex
```

```
 -12.6832 + 0.0000i
  -5.8284 + 5.0757i
  -5.8284 - 5.0757i
  -5.9172 + 0.0000i
  -8.3472 + 4.4935i
  -8.3472 - 4.4935i
```

```
eig(A_lin_syms_new - B_lin_syms_new*K_cvx)
```

```
ans = 6×1 complex
  -0.3433 + 0.2471i
  -0.3433 - 0.2471i
  -1.5654 + 3.4358i
  -1.5654 - 3.4358i
  -2.1149 + 2.1402i
  -2.1149 - 2.1402i
```

f_syn =

$$
\left(\begin{array}{c}
x_4 \\
x_5 \\
x_6 \\
-\frac{280\,u - 981\sin(2\,x_2) + 100\,x_5^2\sin(x_2) + 45\,x_6^2\sin(x_3) - 120\,u\cos(\sigma_3) + 45\,x_6^2\,\sigma_1}{20\,\sigma_2} \\
\frac{1575\,x_6^2\sin(x_2 - x_3) - 30411\sin(x_2) - 8829\sin(x_2 - 2\,x_3) + 250\,x_5^2\sin(2\,x_2) + 1400\,u\cos(x_2) + 450\,x_5^2\sin(\sigma_3) - 600\,u\cos(x_2 - 2\,x_3) + 225\,x_6^2\sin(x_2 + x_3)}{50\,\sigma_2} \\
-\frac{300\sin(x_2 - x_3)\,x_5^2 + 135\sin(\sigma_3)\,x_6^2 - 2943\,\sigma_1 + 2943\sin(x_3) + 200\,u\cos(2\,x_2 - x_3) - 200\,u\cos(x_3)}{75\cos(2\,x_2) + 135\cos(\sigma_3) - 390}
\end{array}\right)
$$

```
f_sym_new_1 =
```

$$
\begin{pmatrix}
x_4 \\
x_5 \\
x_6 \\
-\frac{280\,u_1 - 981\sin(2\,x_2) + 100\,x_5^2\sin(x_2) + 45\,x_6^2\sin(x_3) - 120\,u_1\cos(\sigma_4) - 560\,u_2\cos(x_2) + 45\,x_6^2\,\sigma_1 + 240\,u_2\,\sigma_2}{20\,\sigma_3} \\
\frac{1200\,u_2\cos(2\,x_3) - 8829\sin(x_2 - 2\,x_3) - 30411\sin(x_2) - 7600\,u_2 + 1575\,x_6^2\sin(x_2 - x_3) + 250\,x_5^2\sin(2\,x_2) + 1400\,u_1\cos(x_2) + 450\,x_5^2\sin(\sigma_4) - 600\,u_1\,\sigma_2 + 225\,x_6^2\sin(x_2 + x_3)}{50\,\sigma_3} \\
-\frac{300\sin(x_2 - x_3)\,x_5^2 + 135\sin(\sigma_4)\,x_6^2 - 2943\,\sigma_1 + 2943\sin(x_3) + 200\,u_1\cos(2\,x_2 - x_3) + 400\,u_2\cos(x_2 + x_3) - 200\,u_1\cos(x_3) - 1360\,u_2\cos(x_2 - x_3)}{75\cos(2\,x_2) + 135\cos(\sigma_4) - 390}
\end{pmatrix}
$$

where

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

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

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

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

f_sym_new_2 =

$$
\begin{pmatrix}
x_4 \\
x_5 \\
x_6 \\
-\frac{840\, u_1 - 2943 \sin(2\, x_2) + 300\, x_5^2 \sin(x_2) + 135\, x_6^2 \sin(x_3) - 360\, u_1 \cos(\sigma_5) + 800\, u_3\, \sigma_2 - 1680\, u_2 \cos(x_2) - 800\, u_3 \cos(x_3) + 135\, x_6^2\, \sigma_1 + 720\, u_2\, \sigma_3}{60\, \sigma_4} \\
\frac{3600\, u_2 \cos(2\, x_3) - 26487 \sin(x_2 - 2\, x_3) - 91233 \sin(x_2) - 22800\, u_2 + 4725\, x_6^2 \sin(x_2 - x_3) + 750\, x_5^2 \sin(2\, x_2) - 4000\, u_3 \cos(x_2 + x_3) + 4200\, u_1 \cos(x_2) + 1350\, x_5^2 \sin(\sigma_5) - 1800\, u_1\, \sigma_3 + 13600\, u_3 \cos(x_2 - x_3) + 675\, x_6^2 \sin(x_2 + x_3)}{150\, \sigma_4} \\
-\frac{2700 \sin(x_2 - x_3)\, x_5^2 + 1215 \sin(\sigma_5)\, x_6^2 + 13600\, u_3 - 26487\, \sigma_1 + 26487 \sin(x_3) - 4000\, u_3 \cos(2\, x_2) + 1800\, u_1\, \sigma_2 + 3600\, u_2 \cos(x_2 + x_3) - 1800\, u_1 \cos(x_3) - 12240\, u_2 \cos(x_2 - x_3)}{135\, \sigma_4}
\end{pmatrix}
$$

```
%% DIPC: Nonlinear sim + observer-controller + 3D animation (updated for 30s + video)
clear; clc; close all;

% -----------------------
% System parameters
% -----------------------
M = 1.5; m1 = 0.5; l1 = 0.5; m2 = 0.75; l2 = 0.75; g = 9.81;

% Equilibrium (from earlier)
x_e = [0.1; deg2rad(60); deg2rad(45); 0; 0; 0];
u_e = [0; -5.3098; -3.9019];

% Linearized matrices
A = [ 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 = [ 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 = [1 0 0 0 0 0;
     0 1 0 0 0 0;
     0 0 1 0 0 0];

% Controller K
K = [ 0.5059, 0.7096, 1.4190, 1.9847, 0.5756, 0.7611;
      0.0858, 5.2626, 2.4848, 0.4269, 1.1523, 1.1960;
      0.1171, 1.5430, 8.7194, 0.5831, 1.2254, 1.8557 ];

alpha = 4;
n = size(A,1); p = size(C,1); ep = 1e-6;
L = [];
try
    cvx_begin sdp quiet
        variable P(n,n) symmetric
        variable Y(n,p)
        A'*P + P*A + 2*alpha*P - C'*Y' - Y*C <= -ep*eye(n)
        P >= ep*eye(n)
    cvx_end
    if exist('cvx_status','var') && strcmpi(cvx_status,'Solved')
        L = P\Y;
    end
catch
end
if isempty(L)
    warning('CVX/L not available — using backup observer gain.');
    L = eye(n,p)*5;
end


tspan = [0 15];
x0_real = [3; 0; 0; 0; 0; 0];
z0 = zeros(6,1);
x_aug0 = [x0_real; z0];
params = struct('M',M,'m1',m1,'m2',m2,'l1',l1,'l2',l2,'g',g);
opts = odeset('RelTol',1e-6,'AbsTol',1e-9);

```

```
[t_out, x_aug_out] = ode45(@(t,xa) aug_dynamics(t,xa,K,L,A,B,C,x_e,u_e,params), tspan, x_aug0, opts);

frameRate = 30;
t_uniform = linspace(t_out(1), t_out(end), ceil(frameRate*(t_out(end)-t_out(1))));
x_real = interp1(t_out, x_aug_out(:,1:6), t_uniform, 'linear', 'extrap');
t = t_uniform;

xc = x_real(:,1);
theta1 = x_real(:,2);
theta2 = x_real(:,3);
z_offset1 = 0.03; z_offset2 = -0.03;
p1x = xc + l1*sin(theta1); p1y = l1*cos(theta1);
p2x = p1x + l2*sin(theta2); p2y = p1y + l2*cos(theta2);

fig = figure('Name','DIPC 3D Animation','Color','w','Units','normalized','Position',[0.2 0.2 0.6 0.6]);
ax = axes('Parent',fig);
axis(ax,[-10 10 -0.2 2 -1 1]);
view(0,120); grid on; hold on;
xlabel('X'); ylabel('Y'); zlabel('Z');
title('Double Inverted Pendulum on Cart (3D)');
axis equal;
set(gcf,'Renderer','opengl');

[Xg,Zg] = meshgrid(-10:0.2:10, -1:0.2:1);
Yg = -0.05 * ones(size(Xg));
surf(Xg,Yg,Zg,'FaceColor',[0.95 0.95 0.95],'EdgeColor','none','FaceAlpha',0.9);

CART_W = 0.5; CART_H = 0.2; CART_D = 0.3;
[XcCube, YcCube, ZcCube] = ndgrid([-CART_W/2, CART_W/2],[0, CART_H],[-CART_D/2, CART_D/2]);
cart_base_verts = [XcCube(:), YcCube(:), ZcCube(:)];
cart_faces = [1 2 4 3; 1 2 6 5; 2 4 8 6; 4 3 7 8; 3 7 5 1; 7 8 6 5];
init_x = xc(1);
cart_patch = patch('Vertices', cart_base_verts + [init_x 0 0], 'Faces', cart_faces, ...
                   'FaceColor',[0.8 0.2 0.2], 'EdgeColor','k');

hPend1 = plot3([0 0],[0 0],[0 0],'-','LineWidth',6,'Color',[0.2 0.8 0.2]);
hPend2 = plot3([0 0],[0 0],[0 0],'-','LineWidth',6,'Color',[0.2 0.2 0.8]);
hM1 = plot3(0,0,0,'o','MarkerSize',8,'MarkerFaceColor',[1 0 0],'MarkerEdgeColor','k');
hM2 = plot3(0,0,0,'o','MarkerSize',8,'MarkerFaceColor',[0 0 1],'MarkerEdgeColor','k');
camlight; lighting gouraud;


doSaveVideo = true;
videoFile = 'DIPC_3D.mp4';
if doSaveVideo
    v = VideoWriter(videoFile,'MPEG-4');
    v.FrameRate = frameRate;
    v.Quality = 100;
    open(v);
end

fprintf('Animating %d frames (%.1f s @ %d fps)\n', length(t), t(end)-t(1), frameRate);
hWait = waitbar(0,'Animating...','Name','DIPC Animation Progress');

for k = 1:length(t)
    xk = xc(k);
    p1 = [p1x(k), p1y(k), z_offset1];
    p2 = [p2x(k), p2y(k), z_offset2];
    hinge = [xk, CART_H, 0];
```

```
    set(cart_patch,'Vertices',cart_base_verts + [xk 0 0]);
    set(hPend1,'XData',[hinge(1),p1(1)],'YData',[hinge(2),p1(2)],'ZData',[hinge(3),p1(3)]);
    set(hPend2,'XData',[p1(1),p2(1)],'YData',[p1(2),p2(2)],'ZData',[p1(3),p2(3)]);
    set(hM1,'XData',p1(1),'YData',p1(2),'ZData',p1(3));
    set(hM2,'XData',p2(1),'YData',p2(2),'ZData',p2(3));
    xlim([xk-1.2, xk+1.2]);
    drawnow;
    frame = getframe(fig);

    if doSaveVideo
        writeVideo(v, frame);
    end

    if mod(k,ceil(length(t)/100))==0
        waitbar(k/length(t),hWait,sprintf('Animating... %d%%',round(100*k/length(t))));
    end
end


if doSaveVideo
    close(v);
    fprintf('� Video saved to: %s\n', fullfile(pwd, videoFile));
end
if ishandle(hWait), close(hWait); end
disp('Animation complete.');

function dx_aug = aug_dynamics(~,x_aug,K,L,A,B,C,x_e,u_e,params)
    x_real = x_aug(1:6); z = x_aug(7:12);
    u_dev = -K*z; u = u_e + u_dev;

    try
        if exist('compute_D','file')==2 && exist('compute_C','file')==2 && exist('compute_G','file')==2
            Dm = compute_D(x_real(1),x_real(2),x_real(3),params.M,params.m1,params.m2,params.l1,params.l2);
            Cm = compute_C(x_real(1),x_real(2),x_real(3),x_real(4),x_real(5),x_real(6),params.m1,params.m2,params.l1,params.l2);
            Gv = compute_G(x_real(1),x_real(2),x_real(3),params.m1,params.m2,params.l1,params.l2,params.g);
            qd = x_real(4:6);
            qdd = Dm\(eye(3)*u - Cm*qd - Gv);
            dx_real = [qd; qdd];
        else
            x_dev = x_real - x_e;
            dx_real = A*x_dev + B*(u - u_e);
        end
    catch
        x_dev = x_real - x_e;
        dx_real = A*x_dev + B*(u - u_e);
    end

    x_dev = x_real - x_e;
    y_real_dev = C*x_dev;
    y_est_dev = C*z;
    dz = A*z + B*u_dev + L*(y_real_dev - y_est_dev);
    dx_aug = [dx_real; dz];
end
```