HW2 Gabe Colangelo

clear

close all

warning off

clc

% Symbolic variables

syms x1 x2 x3 x4 x5 x6 u u1 u2 x(t) x\_dot(t) theta1(t) theta\_dot\_1(t)...

theta2(t) theta\_dot\_2(t) t theta\_ddot\_1 x\_ddot theta\_ddot\_2...

m1 m2 M L1 L2 g real

% state vector

% x1 = x

% x2 = theta1

% x3 = theta2

% x4 = x\_dot

% x5 = theta\_dot\_1

% x6 = theta\_dot\_2

# Problem 1 - Lyapunov Stability

% State vector

x\_state = [x1;x2;x3;x4;x5;x6];

% System Parameters

m1\_num = 0.5;

L1\_num = 0.5;

m2\_num = 0.75;

L2\_num = 0.75;

M\_num = 1.5;

g\_num = 9.81;

% Non-linear Model

xdot = DIPC([],x\_state,u,m1\_num,m2\_num,M\_num,L1\_num,L2\_num,g\_num);

% Ouputs

y = [x1;x2;x3];

% Equilibirum pair - origin

xe = zeros(6,1);

ue = 0;

% Jacobian Matrices/ Linearized Model about origin

A = double(subs(jacobian(xdot,[x1;x2;x3;x4;x5;x6]),[x\_state;u],[xe;ue]));

B = double(subs(jacobian(xdot,u),[x\_state;u],[xe;ue]));

C = double(jacobian(y,[x1;x2;x3;x4;x5;x6]));

D = double(jacobian(y,u));

% Lyapunov Equation: A'P + PA = -Q

Q = eye(6);

% Create symbolic symmetric matrix P

P = sym('P',6,'real');

P = tril(P,0) + tril(P,-1).';

% Define Symbolic Lyapunov

Lyap\_eqn= A'\*P + P\*A == -Q;

% Get system of equations

eqns = tril(Lyap\_eqn);

eqns = eqns(eqns~=0);

% Get vector of unknowns

vars = tril(P);

vars = vars(vars~=0);

% Solve lyapunov equation

sol = solve(eqns,vars);

% Extract solutions

fnames = fieldnames(sol);

for i = 1:length(fnames)

sol\_vec = double(sol.(fnames{i}));

end

% Check if solution to Lyapunov equation is empty & A's eigenvalues are in RHP

if (isempty(sol\_vec) == 1) && (max(eig(A) > 0) == 1)

disp('A solution to the continous time Lyapunov matrix equation does not exist');

disp('This is because the system is unstable as the eigenvalues of A are in the right hand plane');

disp('Thus the equilibrium state/open loop system is NOT asymptotically stable in the sense of Lyapunov')

end

A solution to the continous time Lyapunov matrix equation does not exist

This is because the system is unstable as the eigenvalues of A are in the right hand plane

Thus the equilibrium state/open loop system is NOT asymptotically stable in the sense of Lyapunov

# Problem 2 - Linear State Feedback Controller Design

% Check system controllability

co = ctrb(A,B);

if rank(co) == length(A)

disp('The pair (A,B) is controllable')

end

The pair (A,B) is controllable

% Get dimensions of B

[n, m] = size(B);

disp('The linear state-feedback controller for the linearized model is:')

The linear state-feedback controller for the linearized model is:



% Use CVX to solve matrix inequality and determine K

cvx\_begin sdp quiet

% Variable definition

variable S(n, n) symmetric

variable Z(m, n)

% LMIs with robustness term (all eigenvalues less than -1)

S\*A' + A\*S -Z'\*B'- B\*Z +2\*S <= -eps\*eye(n);

S >= eps\*eye(n);

cvx\_end

disp('The control gains for the control law del\_u = -K\*del\_x are:')

The control gains for the control law del\_u = -K\*del\_x are:

% compute K matrix

K = Z/S

K = 1×6

47.2677 -513.1457 922.3308 68.5735 10.9892 178.6912

# Problem 3 - Closed Loop Transfer Function for Linearized Model

% Laplace Variable

s = tf('s');

% Closed Loop transfer Function Matrix Equation

Y\_R = (C - D\*K)\*inv(s\*eye(size(A)) - A + B\*K)\*B + D;

disp('Closed loop Transfer function for X to r:')

Closed loop Transfer function for X to r:

minreal(Y\_R(1),1e-5)

ans =

0.6667 s^4 - 3.407e-12 s^3 - 54.5 s^2 + 2.458e-10 s + 427.7

-----------------------------------------------------------------------------

s^6 + 31.06 s^5 + 617.6 s^4 + 4533 s^3 + 1.644e04 s^2 + 2.933e04 s + 2.022e04

Continuous-time transfer function.

disp('Closed loop Transfer Function for theta\_1 to r:')

Closed loop Transfer Function for theta\_1 to r:

minreal(Y\_R(2),1e-5)

ans =

-1.333 s^4 + 9.179e-13 s^3 + 43.6 s^2 - 3.772e-11 s - 3.446e-11

-----------------------------------------------------------------------------

s^6 + 31.06 s^5 + 617.6 s^4 + 4533 s^3 + 1.644e04 s^2 + 2.933e04 s + 2.022e04

Continuous-time transfer function.

disp('Closed loop Transfer Function for theta\_2 to r:')

Closed loop Transfer Function for theta\_2 to r:

minreal(Y\_R(3),1e-5)

ans =

43.6 s^2 - 3.59e-11 s - 3.114e-11

-----------------------------------------------------------------------------

s^6 + 31.06 s^5 + 617.6 s^4 + 4533 s^3 + 1.644e04 s^2 + 2.933e04 s + 2.022e04

Continuous-time transfer function.

# Problem 4 - Closed Loop Lyapunov Function for Linearized Model

% Closed loop A matrix

A\_cl = (A - B\*K);

% Lyapunov Function: V = del\_x'\*P\*del\_x

disp('The Lyapunov function for the closed-loop system comprised of the linearized model is:')

The Lyapunov function for the closed-loop system comprised of the linearized model is:



disp('Where P is given by')

Where P is given by

% Solve closed Loop Lyapunov Matrix Equation:A\_cl'\*P\_cl + P\_cl\*A\_cl = -Q



P\_cl = lyap(A\_cl',Q)

P\_cl = 6×6

2.5654 -1.2330 11.5041 2.1694 1.0768 2.7206

-1.2330 32.5984 -56.3316 -2.6064 -0.8137 -11.2030

11.5041 -56.3316 166.4115 18.1827 8.3373 35.8444

2.1694 -2.6064 18.1827 3.1487 1.5451 4.2450

1.0768 -0.8137 8.3373 1.5451 0.7940 1.9700

2.7206 -11.2030 35.8444 4.2450 1.9700 7.9145

if min(eig(P\_cl) > 0) == 1 && issymmetric(P\_cl) == 1

disp('P is symmetric positive definite')

disp('Thus the equilibrium state of interest of the closed-loop system is asymptotically stable in the sense of Lyapunov')

end

P is symmetric positive definite

Thus the equilibrium state of interest of the closed-loop system is asymptotically stable in the sense of Lyapunov

# Problem 5 - State Feedback Controller for Two Input System

% Call Lagrangian for DIPC

L = DIPC\_Lagrangian(t,x,x\_dot, theta1, theta\_dot\_1, theta2, theta\_dot\_2, M, m1,m2, L1, L2, g);

% Solve Lagrange's Equations of Motion

% q = x, Q = u1

eqn\_x = subs(simplify(diff(diff(L,x\_dot),t) - diff(L,x)),[diff(x(t),t), diff(theta1(t),t)...

,diff(theta2(t),t), diff(x\_dot,t), diff(theta\_dot\_1(t), t), diff(theta\_dot\_2(t), t)],...

[x\_dot, theta\_dot\_1, theta\_dot\_2, x\_ddot, theta\_ddot\_1, theta\_ddot\_2]) == u1;

% q = theta1, Q = u2

eqn\_theta1 = subs(simplify(diff(diff(L,theta\_dot\_1),t) - diff(L,theta1)),[diff(x(t),t), diff(theta1(t),t)...

,diff(theta2(t),t), diff(x\_dot,t), diff(theta\_dot\_1(t), t), diff(theta\_dot\_2(t), t)],...

[x\_dot, theta\_dot\_1, theta\_dot\_2, x\_ddot, theta\_ddot\_1, theta\_ddot\_2]) == u2;

% q = theta2, Q = 0

eqn\_theta2 = subs(simplify(diff(diff(L,theta\_dot\_2),t) - diff(L,theta2)),[diff(x(t),t), diff(theta1(t),t)...

,diff(theta2(t),t), diff(x\_dot,t), diff(theta\_dot\_1(t), t), diff(theta\_dot\_2(t), t)],...

[x\_dot, theta\_dot\_1, theta\_dot\_2, x\_ddot, theta\_ddot\_1, theta\_ddot\_2]) == 0;

% Solve system of equations for 2nd derivative of states

sys\_eqn = solve([eqn\_x,eqn\_theta1,eqn\_theta2],[x\_ddot,theta\_ddot\_1,theta\_ddot\_2]);

% Put EOM into state space form

x1\_dot = x4;

x2\_dot = x5;

x3\_dot = x6;

x4\_dot = subs(simplify(sys\_eqn.x\_ddot),[x theta1 theta2 x\_dot theta\_dot\_1 theta\_dot\_2],[x1 x2 x3 x4 x5 x6]);

x5\_dot = subs(simplify(sys\_eqn.theta\_ddot\_1),[x theta1 theta2 x\_dot theta\_dot\_1 theta\_dot\_2],[x1 x2 x3 x4 x5 x6]);

x6\_dot = subs(simplify(sys\_eqn.theta\_ddot\_2),[x theta1 theta2 x\_dot theta\_dot\_1 theta\_dot\_2],[x1 x2 x3 x4 x5 x6]);

% Define Non-linear system

f = [x1\_dot;x2\_dot;x3\_dot;x4\_dot;x5\_dot;x6\_dot];

h = [x1;x2;x3];

% Jacobian Matrices

df\_dx = subs(jacobian(f,[x1;x2;x3;x4;x5;x6]),[m1 m2 M L1 L2 g],[m1\_num, m2\_num,M\_num,L1\_num, L2\_num, g\_num]);

df\_du = subs(jacobian(f,[u1;u2]),[m1 m2 M L1 L2 g],[m1\_num, m2\_num,M\_num,L1\_num, L2\_num, g\_num]);

% Input for equilibirum at origin

u1e = 0;

u2e = 0;

disp('The updated linearized model with two inputs is:')

The updated linearized model with two inputs is:

% Redefine State Matrices with extra input

A = double(subs(df\_dx,[x1;x2;x3;x4;x5;x6;u1;u2],[xe;u1e;u2e]))

A = 6×6

0 0 0 1.0000 0 0

0 0 0 0 1.0000 0

0 0 0 0 0 1.0000

0 -8.1750 0 0 0 0

0 65.4000 -29.4300 0 0 0

0 -32.7000 32.7000 0 0 0

B = double(subs(df\_du,[x1;x2;x3;x4;x5;x6;u1;u2],[xe;u1e;u2e]))

B = 6×2

0 0

0 0

0 0

0.6667 -1.3333

-1.3333 10.6667

0 -5.3333

C

C = 3×6

1 0 0 0 0 0

0 1 0 0 0 0

0 0 1 0 0 0

D = double(jacobian(h,[u1;u2]))

D = 3×2

0 0

0 0

0 0

% Check system controllability

co = ctrb(A,B);

if rank(co) == length(A)

disp('The pair (A,B) is controllable')

end

The pair (A,B) is controllable

% Get dimensions of new B

[n, m] = size(B);

% Use CVX to solve matrix inequality and determine new K

cvx\_begin sdp quiet

% Variable definition

variable S(n, n) symmetric

variable Z(m, n)

% LMIs with robustness term (all eigenvalues less than -1)

S\*A' + A\*S -Z'\*B'- B\*Z +2\*S <= -eps\*eye(n);

S >= eps\*eye(n);

cvx\_end

disp('The linear state-feedback controller for the new linearized model is: del\_u = -K\*del\_x')

The linear state-feedback controller for the new linearized model is: del\_u = -K\*del\_x



disp('The new control gains for the control law del\_u = -K\*del\_x are:')

The new control gains for the control law del\_u = -K\*del\_x are:

% compute new K matrix

K = Z/S

K = 2×6

-34.4541 -30.7791 -192.7857 -38.8292 -24.6616 -45.8890

-5.2096 3.5185 -30.2032 -5.6298 -3.0797 -6.3989

# Problem 6 - Luenberger Observer Design

% Check system observability

ob = obsv(A,C);

if rank(ob) == length(A)

disp('The pair (A,C) is observable')

end

The pair (A,C) is observable

% Dimensions of C matrix

[p, n] = size(C);

% Use CVX to solve matrix inequality and determine L

cvx\_begin sdp quiet

% Variable definition

variable P(n, n) symmetric

variable Y(n, p)

% LMI with robustness term (all eigenvalues less than -2)

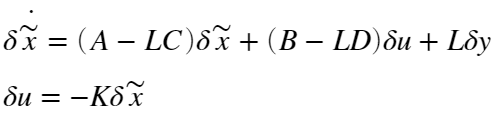
A'\*P + P\*A - C'\*Y' - Y\*C + 4\*P <= -eps\*eye(n);

P >= eps\*eye(n)

cvx\_end

disp('The Luenberger observer takes the form of:')

The Luenberger observer takes the form of:



disp('The control gains for the Luenberger observer are:')

The control gains for the Luenberger observer are:

% solver for observer gain matrix

L = P\Y

L = 6×3

6.3897 0.0002 0.1509

-8.3252 15.8391 -3.5881

-3.9365 -3.3157 11.9497

32.0119 -18.6302 2.4456

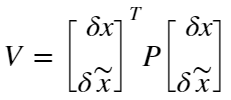
-66.3206 183.0734 -88.9352

-0.5446 -87.8092 97.5863

# Problem 7 - Lyapunov Function for combined observer controller compensator

disp('The Lyapunov function for the combined observer-controller compensator closed loop system is: ')

The Lyapunov function for the combined observer-controller compensator closed loop system is:



% A matrix for closed loop system driven by the combined observer controller compensator

A\_cl\_full = [A, -B\*K; L\*C, A - L\*C - B\*K];

% Solve Lyapunov Matrix Equation for combined observer controller compensator system :A\_cl'\*P\_cl + P\_cl\*A\_cl = -Q

P\_cl\_full = lyap(A\_cl\_full',eye(12))

P\_cl\_full = 12×12

51.7139 11.9401 22.0707 -0.5000 2.6561 7.1675 -49.4543 ⋯

11.9401 20.6948 12.6307 -2.6561 -0.5000 1.7712 -10.6477

22.0707 12.6307 114.2623 -7.1675 -1.7712 -0.5000 -8.7921

-0.5000 -2.6561 -7.1675 4.8737 2.1619 3.5763 0.1885

2.6561 -0.5000 -1.7712 2.1619 1.6746 1.9689 -2.8448

7.1675 1.7712 -0.5000 3.5763 1.9689 4.7547 -7.7321

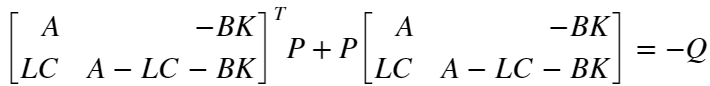
-49.4543 -10.6477 -8.7921 0.1885 -2.8448 -7.7321 51.7781

-12.1827 -14.0938 -6.8286 2.2758 0.4461 -1.6701 11.9324

-13.5164 -4.9425 -55.9131 7.1460 2.7347 1.9692 9.3485

2.0906 4.9453 21.9802 -4.7500 -2.0399 -3.3126 0.9682

⋮



if min(eig(P\_cl\_full) > 0) == 1 && issymmetric(P\_cl\_full) == 1

disp('P is symmetric positive definite')

disp('Thus the equilibrium state of interest of the closed-loop system is asymptotically stable in the sense of Lyapunov')

end

P is symmetric positive definite

Thus the equilibrium state of interest of the closed-loop system is asymptotically stable in the sense of Lyapunov

# Problem 8 - Transfer Function for combined observer controller compensator

% Closed Loop transfer Function Matrix Equation

Y\_R = (C - D\*K)\*inv(s\*eye(size(A)) - A + B\*K)\*B + D;

disp('Observer - Controller Closed loop Transfer function for X to r\_1:')

Observer - Controller Closed loop Transfer function for X to r\_1:

minreal(Y\_R(1,1),1e-5)

ans =

0.6667 s^4 + 6.327 s^3 + 71.65 s^2 + 214.8 s + 182.3

------------------------------------------------------------------

s^6 + 15.78 s^5 + 175.2 s^4 + 992.2 s^3 + 3551 s^2 + 6571 s + 4906

Continuous-time transfer function.

disp('Observer - Controller Closed loop Transfer Function for theta\_1 to r\_1:')

Observer - Controller Closed loop Transfer Function for theta\_1 to r\_1:

minreal(Y\_R(2,1),1e-5)

ans =

-1.333 s^4 - 15.48 s^3 - 143.4 s^2 - 392.7 s - 363.4

------------------------------------------------------------------

s^6 + 15.78 s^5 + 175.2 s^4 + 992.2 s^3 + 3551 s^2 + 6571 s + 4906

Continuous-time transfer function.

disp('Observer - Controller Closed loop Transfer Function for theta\_2 to r\_1:')

Observer - Controller Closed loop Transfer Function for theta\_2 to r\_1:

minreal(Y\_R(3,1),1e-5)

ans =

1.883 s^3 + 0.05623 s^2 - 1.203e-12 s + 4.746e-14

------------------------------------------------------------------

s^6 + 15.78 s^5 + 175.2 s^4 + 992.2 s^3 + 3551 s^2 + 6571 s + 4906

Continuous-time transfer function.

disp('Observer - Controller Closed loop Transfer function for X to r\_2:')

Observer - Controller Closed loop Transfer function for X to r\_2:

minreal(Y\_R(1,2),1e-3)

ans =

-1.333 s^4 - 31.63 s^3 - 477.7 s^2 - 1720 s - 2147

------------------------------------------------------------------

s^6 + 15.78 s^5 + 175.2 s^4 + 992.3 s^3 + 3552 s^2 + 6572 s + 4906

Continuous-time transfer function.

disp('Observer - Controller Closed loop Transfer function for theta\_1 to r\_2:')

Observer - Controller Closed loop Transfer function for theta\_1 to r\_2:

minreal(Y\_R(2,2),5e-3)

ans =

10.67 s^4 + 119.2 s^3 + 995.3 s^2 + 2709 s + 2404

------------------------------------------------------------------

s^6 + 15.78 s^5 + 175.2 s^4 + 992.3 s^3 + 3552 s^2 + 6571 s + 4905

Continuous-time transfer function.

disp('Observer - Controller Closed loop Transfer function for theta\_2 to r\_2:')

Observer - Controller Closed loop Transfer function for theta\_2 to r\_2:

minreal(Y\_R(3,2),1e-3)

ans =

-5.333 s^4 - 37.31 s^3 - 96.37 s^2 + 1.008e-11 s + 4.754e-12

------------------------------------------------------------------

s^6 + 15.78 s^5 + 175.2 s^4 + 992.1 s^3 + 3551 s^2 + 6571 s + 4905

Continuous-time transfer function.

# Problem 9 Part 1 - Simulation

% Observer IC

z0 = zeros(6,1);

% State IC [m, rad, rad, m/s, rad/s, rad/s]

x0 = [.25 .08 .1 0 0 0]';

% Time interval and vector

dt = 1/200;

time = (0:dt:5)';

% equilibirum ouput for observer

ye = subs(h,[x1; x2; x3],xe(1:3));

% ODE45 solver options

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

% ODE45 Function call

[~, X] = ode45(@(t,x) ControlledDIPC(t, x, A, B, C, D, K, L, [u1e;u2e], ye, M\_num, m1\_num ,m2\_num, L1\_num, L2\_num, g\_num),...

time,[x0;z0], options);

# Problem 9 Part - 2 Animation

% Cart width and height

w = 1;

h = .5;

% Graphics handle - cart

figure

cart = rectangle('position',[X(1,1) - w/2, -h, w, h]);

% Graphics handle - hinge

hinge = line('xdata', X(1,1),'ydata',0,'marker','o','markersize',7);

% Graphics handle - mass 1

mass1 = line('xdata', X(1,1) + L1\_num\*sin(X(1,2)), 'ydata', L1\_num\*cos(X(1,2)),...

'marker','o','markersize',10,'MarkerFaceColor','k');

% Graphics handle - bar 1

bar1 = line('xdata', [X(1,1) X(1,1) + L1\_num\*sin(X(1,2))],'ydata',...

[0 L1\_num\*cos(X(1,2))],'linewidth',3);

% Graphics handle - mass 2

mass2 = line('xdata', X(1,1) + L1\_num\*sin(X(1,2)) + L2\_num\*sin(X(1,3)), 'ydata',...

L1\_num\*cos(X(1,2))+L2\_num\*cos(X(1,3)),'marker','o','markersize',10,'MarkerFaceColor','k');

% Graphics handle - bar 2

bar2 = line('xdata', [(X(1,1) + L1\_num\*sin(X(1,2))), (X(1,1) + L1\_num\*sin(X(1,2)) + L2\_num\*sin(X(1,3)))],'ydata',...

[(L1\_num\*cos(X(1,2))) (L1\_num\*cos(X(1,2))+L2\_num\*cos(X(1,3)))],'linewidth',3);

h\_txt = text(-1.1,1.1,strcat(['Time = ',' ', num2str(time(1)), ' [s]']));

% Define axis limits

axis([-1.1\*(L1\_num + L2\_num), 1.1\*(L1\_num + L2\_num),-1.1\*(L1\_num + L2\_num), 1.1\*(L1\_num + L2\_num)]);

grid on

xlabel('X [m]')

ylabel('Y [m]')

title('Controlled Double Inverted Pendulum')

% Video stuff

vidobj = VideoWriter('DIPC.avi');

open(vidobj);

nframes = length(X);

frames = moviein(nframes);

for i = 2:nframes

% Update handles

set(cart,'position',[X(i,1) - w/2, -h, w, h]);

set(hinge,'xdata', X(i,1),'ydata',0,'marker','o','markersize',7);

set(mass1,'xdata', X(i,1) + L1\_num\*sin(X(i,2)), 'ydata', L1\_num\*cos(X(i,2)),...

'marker','o','markersize',10,'MarkerFaceColor','k');

set(bar1,'xdata', [X(i,1) X(i,1) + L1\_num\*sin(X(i,2))],'ydata',...

[0 L1\_num\*cos(X(i,2))],'linewidth',3);

set(mass2,'xdata', X(i,1) + L1\_num\*sin(X(i,2)) + L2\_num\*sin(X(i,3)), 'ydata',...

L1\_num\*cos(X(i,2))+L2\_num\*cos(X(i,3)),'marker','o','markersize',10,'MarkerFaceColor','k');

set(bar2,'xdata', [(X(i,1) + L1\_num\*sin(X(i,2))), (X(i,1) + L1\_num\*sin(X(i,2)) + L2\_num\*sin(X(i,3)))],'ydata',...

[(L1\_num\*cos(X(i,2))) (L1\_num\*cos(X(i,2))+L2\_num\*cos(X(i,3)))],'linewidth',3);

set(h\_txt,'String',strcat(['Time = ',' ', num2str(time(i)), ' [s]']));

drawnow;

frames(:,i) = getframe(gcf);

writeVideo(vidobj,frames(:,i));

end

close(vidobj);

# Functions

% DIPC equations of motion in state space

function xdot = DIPC(t,x,u,m1,m2,M,L1,L2,g)

% States and inputs

x1 = x(1,1); % x

x2 = x(2,1); % theta\_1

x3 = x(3,1); % theta\_2

x4 = x(4,1); % xdot

x5 = x(5,1); % theta\_1\_dot

x6 = x(6,1); % theta\_2\_dot

% Equations of Motion

x1dot = x4; % xdot

x2dot = x5; % theta\_1\_dot

x3dot = x6; % theta\_2\_dot

% x\_ddot

x4dot = (2\*m1\*u + m2\*u - m2\*u\*cos(2\*x2 - 2\*x3) - g\*m1^2\*sin(2\*x2) +...

2\*L1\*m1^2\*x5^2\*sin(x2) - g\*m1\*m2\*sin(2\*x2) +...

2\*L1\*m1\*m2\*x5^2\*sin(x2) + L2\*m1\*m2\*x6^2\*sin(x3) +...

L2\*m1\*m2\*x6^2\*sin(2\*x2 - x3))/(2\*M\*m1 + M\*m2 + m1\*m2 -...

m1^2\*cos(2\*x2) + m1^2 - m1\*m2\*cos(2\*x2) - M\*m2\*cos(2\*x2 - 2\*x3));

% theta\_1\_ddot

x5dot = -(m1\*u\*cos(x2) + (m2\*u\*cos(x2))/2 - (m2\*u\*cos(x2 - 2\*x3))/2 -...

g\*m1^2\*sin(x2) - M\*g\*m1\*sin(x2) - (M\*g\*m2\*sin(x2))/2 -...

g\*m1\*m2\*sin(x2) + (L1\*m1^2\*x5^2\*sin(2\*x2))/2 - (M\*g\*m2\*...

sin(x2 - 2\*x3))/2 + (L2\*m1\*m2\*x6^2\*sin(x2 + x3))/2 + ...

L2\*M\*m2\*x6^2\*sin(x2 - x3) + (L2\*m1\*m2\*x6^2\*sin(x2 - x3))/2 +...

(L1\*m1\*m2\*x5^2\*sin(2\*x2))/2 + (L1\*M\*m2\*x5^2\*sin(2\*x2 - 2\*x3))/2)...

/(L1\*(M\*m1 + (M\*m2)/2 + (m1\*m2)/2 - (m1^2\*cos(2\*x2))/2 + m1^2/2 ...

- (m1\*m2\*cos(2\*x2))/2 - (M\*m2\*cos(2\*x2 - 2\*x3))/2));

% theta\_2\_ddot

x6dot = ((m1\*u\*cos(2\*x2 - x3))/2 - (m2\*u\*cos(x3))/2 - (m1\*u\*cos(x3))/2 +...

(m2\*u\*cos(2\*x2 - x3))/2 - (M\*g\*m1\*sin(2\*x2 - x3))/2 - ...

(M\*g\*m2\*sin(2\*x2 - x3))/2 + (M\*g\*m1\*sin(x3))/2 + ...

(M\*g\*m2\*sin(x3))/2 + L1\*M\*m1\*x5^2\*sin(x2 - x3) +...

L1\*M\*m2\*x5^2\*sin(x2 - x3) + (L2\*M\*m2\*x6^2\*sin(2\*x2 - 2\*x3))/2)/...

(L2\*(M\*m1 + (M\*m2)/2 + (m1\*m2)/2 - (m1^2\*cos(2\*x2))/2 + m1^2/2 -...

(m1\*m2\*cos(2\*x2))/2 - (M\*m2\*cos(2\*x2 - 2\*x3))/2));

xdot = [x1dot;x2dot;x3dot;x4dot;x5dot;x6dot];

end

% DIPC Lagrangian

function L = DIPC\_Lagrangian(t,x, x\_dot, theta1, theta\_dot\_1, theta2, theta\_dot\_2, M, m1,m2, L1, L2, g)

% Lagrangian for DIPC from HW1

L = (m2\*(x\_dot(t) + L1\*cos(theta1(t))\*theta\_dot\_1(t) + L2\*cos(theta2(t))\*...

theta\_dot\_2(t))^2)/2 + (m1\*(x\_dot(t) + L1\*cos(theta1(t))\*...

theta\_dot\_1(t))^2)/2 + (m2\*(L1\*sin(theta1(t))\*theta\_dot\_1(t) +...

L2\*sin(theta2(t))\*theta\_dot\_2(t))^2)/2 + (M\*x\_dot(t)^2)/2 + ...

(L1^2\*m1\*sin(theta1(t))^2\*theta\_dot\_1(t)^2)/2 - ...

L1\*g\*m1\*cos(theta1(t)) - L1\*g\*m2\*cos(theta1(t)) - L2\*g\*m2\*cos(theta2(t));

end

% Combined Controller-Observer Compensator applied to DIPC

function xdot = ControlledDIPC(t, x, A, B, C, D, K, L, ue, ye, M, m1,m2, L1, L2, g)

% Define State Vector

x1 = x(1,1); % x

x2 = x(2,1); % theta\_1

x3 = x(3,1); % theta\_2

x4 = x(4,1); % xdot

x5 = x(5,1); % theta\_1\_dot

x6 = x(6,1); % theta\_2\_dot

z1 = x(7,1); % delta\_x1\_tilde - estimate of change in x

z2 = x(8,1); % delta\_x2\_tilde - estimate of change in theta\_1

z3 = x(9,1); % delta\_x3\_tilde - estimate of change in theta\_2

z4 = x(10,1); % delta\_x4\_tilde - estimate of change in xdot

z5 = x(11,1); % delta\_x5\_tilde - estimate of change in theta\_1\_dot

z6 = x(12,1); % delta\_x6\_tilde - estimate of change in tbeta\_2\_dot

% delta\_tilde\_x - vector of state pertubation estimates

z = [z1;z2;z3;z4;z5;z6];

% Output vector - x, theta\_1, theta\_2

y = [x1;x2;x3];

% Output pertubation vector

del\_y = y - ye;

% Control law

del\_u = - K\*z;

u = del\_u + ue;

u1 = u(1);

u2 = u(2);

% State Dynamics

x1dot = x4; % xdot

x2dot = x5; % theta\_1\_dot

x3dot = x6; % theta\_2\_dot

x4dot = ((m2\*u2\*cos(x2 - 2\*x3))/2 - (m2\*u2\*cos(x2))/2 - m1\*u2\*cos(x2) +...

L1\*m1\*u1 + (L1\*m2\*u1)/2 - (L1\*g\*m1^2\*sin(2\*x2))/2 + L1^2\*m1^2\*x5^2\*sin(x2)...

- (L1\*m2\*u1\*cos(2\*x2 - 2\*x3))/2 - (L1\*g\*m1\*m2\*sin(2\*x2))/2 +...

L1^2\*m1\*m2\*x5^2\*sin(x2) + (L1\*L2\*m1\*m2\*x6^2\*sin(2\*x2 - x3))/2 + ...

(L1\*L2\*m1\*m2\*x6^2\*sin(x3))/2)/(L1\*(M\*m1 + (M\*m2)/2 + (m1\*m2)/2 -...

(m1^2\*cos(2\*x2))/2 + m1^2/2 - (m1\*m2\*cos(2\*x2))/2 - (M\*m2\*cos(2\*x2 - 2\*x3))/2));

x5dot = (M\*u2 + m1\*u2 + (m2\*u2)/2 - (m2\*u2\*cos(2\*x3))/2 - L1\*m1\*u1\*cos(x2) -...

(L1\*m2\*u1\*cos(x2))/2 + (L1\*m2\*u1\*cos(x2 - 2\*x3))/2 + L1\*g\*m1^2\*sin(x2) -...

(L1^2\*m1^2\*x5^2\*sin(2\*x2))/2 - (L1^2\*m1\*m2\*x5^2\*sin(2\*x2))/2 -...

(L1^2\*M\*m2\*x5^2\*sin(2\*x2 - 2\*x3))/2 + L1\*M\*g\*m1\*sin(x2) +...

(L1\*M\*g\*m2\*sin(x2))/2 + L1\*g\*m1\*m2\*sin(x2) +...

(L1\*M\*g\*m2\*sin(x2 - 2\*x3))/2 - (L1\*L2\*m1\*m2\*x6^2\*sin(x2 + x3))/2 -...

L1\*L2\*M\*m2\*x6^2\*sin(x2 - x3) - (L1\*L2\*m1\*m2\*x6^2\*sin(x2 - x3))/2)...

/(L1^2\*(M\*m1 + (M\*m2)/2 + (m1\*m2)/2 - (m1^2\*cos(2\*x2))/2 + m1^2/2 -...

(m1\*m2\*cos(2\*x2))/2 - (M\*m2\*cos(2\*x2 - 2\*x3))/2));

x6dot = (m1\*u2\*cos(x2 + x3) - m1\*u2\*cos(x2 - x3) - m2\*u2\*cos(x2 - x3) -...

2\*M\*u2\*cos(x2 - x3) + m2\*u2\*cos(x2 + x3) - L1\*m1\*u1\*cos(x3) - ...

L1\*m2\*u1\*cos(x3) + L1\*m1\*u1\*cos(2\*x2 - x3) + L1\*m2\*u1\*cos(2\*x2 - x3) -...

L1\*M\*g\*m1\*sin(2\*x2 - x3) - L1\*M\*g\*m2\*sin(2\*x2 - x3) + L1\*M\*g\*m1\*sin(x3) +...

L1\*M\*g\*m2\*sin(x3) + 2\*L1^2\*M\*m1\*x5^2\*sin(x2 - x3) +...

2\*L1^2\*M\*m2\*x5^2\*sin(x2 - x3) + L1\*L2\*M\*m2\*x6^2\*sin(2\*x2 - 2\*x3))/...

(L1\*L2\*(2\*M\*m1 + M\*m2 + m1\*m2 - m1^2\*cos(2\*x2) + m1^2 ...

- m1\*m2\*cos(2\*x2) - M\*m2\*cos(2\*x2 - 2\*x3)));

xdot(1:6,1) = [x1dot;x2dot;x3dot;x4dot;x5dot;x6dot];

% Luenberger Observer Dynamics

del\_y\_tilde = C\*z + D\*del\_u;

zdot = A\*z + B\*del\_u + L\*(del\_y - del\_y\_tilde);

xdot(7:12,1)= zdot;

end