HW1 Gabe Colangelo

clear

close all

warning off

clc

# Problem 1 Lagrangian Derivation & E.O.M

% Lagrangian Derivation

% x1 = x

% x2 = theta1

% x3 = theta2

% x4 = x\_dot

% x5 = theta\_dot\_1

% x6 = theta\_dot\_2

syms x(t) x\_dot(t) theta1(t) theta\_dot\_1(t) theta2(t) theta\_dot\_2(t) M m1 m2...

L1 L2 x1 x2 x3 x4 x5 x6 g theta1\_int theta2\_int theta\_ddot\_1 x\_ddot ...

theta\_ddot\_2 u s real

% Position vector to cart - inertial coordinates

r\_M = [x;0;0];

% Rotation Matrix from body frame 1 to inertial frame

I\_DCM\_B = [sin(theta1), cos(theta1), 0;...

cos(theta1), -sin(theta1),0;...

0, 0 ,-1];

% Position vector to mass 1, inertial frame: r\_m1 = x ix + l br

r\_m1 = r\_M + I\_DCM\_B\*[L1;0;0];

% Define angle between body frame 1 and body frame 2

beta = theta2 - theta1;

% Rotation Matrix from body frame 1 to inertial frame

B\_DCM\_C = [cos(beta), -sin(beta), 0;...

sin(beta), cos(beta), 0;...

0, 0, 1];

% Position vector to mass 2, inertial frame: r\_m2 = x ix + l br + l cr

r\_m2 = r\_M + I\_DCM\_B\*[L1;0;0] + I\_DCM\_B\*B\_DCM\_C\*[L2;0;0];

% Inertial Velocities

v\_M = subs(diff(r\_M),diff(x(t),t),x\_dot);

v\_m1 = simplify(subs(diff(r\_m1),[diff(x(t),t), diff(theta1(t),t)]...

,[x\_dot, theta\_dot\_1]));

v\_m2 = simplify(subs(diff(r\_m2),[diff(x(t),t), diff(theta1(t),t)...

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

% Kinetic Energy

T = simplify((1/2)\*M\*transpose(v\_M)\*v\_M + (1/2)\*m1\*transpose(v\_m1)\*v\_m1...

+ (1/2)\*m2\*transpose(v\_m2)\*v\_m2);

% Gravitational Forces

W\_M = [0;-M\*g;0];

W\_m1 = [0;-m1\*g;0];

W\_m2 = [0;-m2\*g;0];

% Differential Displacements

dsM = subs(diff(r\_M),diff(x(t),t),1);

dsm1 = simplify(subs(diff(r\_m1),[diff(x(t),t), diff(theta1(t),t)]...

,[1 1]));

dsm2 = simplify(subs(diff(r\_m2),[diff(x(t),t), diff(theta1(t),t)...

,diff(theta2(t),t)],[1 1 1]));

% Potential Energies of mass M: V = -int(dot(F,ds))

V\_M = -int(transpose(W\_M)\*dsM);

% Potential Energy of mass 1

int\_m1 = subs(transpose(W\_m1)\*dsm1,theta1,theta1\_int);

V\_m1 = subs(-int(int\_m1,theta1\_int),theta1\_int,theta1);

% Potential Energy of mass 2

int\_m2 = subs(transpose(W\_m2)\*dsm2,[theta1 theta2],[theta1\_int, theta2\_int]);

V\_m2 = subs(-int(subs(int\_m2,theta2\_int,0),theta1\_int) +....

-int(subs(int\_m2,theta1\_int,0),theta2\_int),[theta1\_int,theta2\_int],...

[theta1,theta2]);

% Total potential energy

V = simplify(V\_M + V\_m1 + V\_m2);

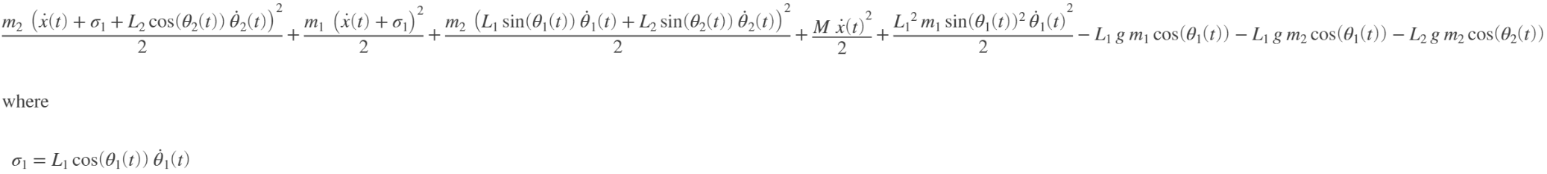
% Lagrangian

disp('The Lagrangian is')

The Lagrangian is

L = simplify(T - V)

L(t) =



% Lagrange Equations of Motions: d/dt(del\_L/del\_qdot) - del\_L/del\_q = Q

disp('Lagranges Equations of Motion are:')

Lagranges Equations of Motion are:

% : q = x, Q = u

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]) == u

eqn\_x(t) = 

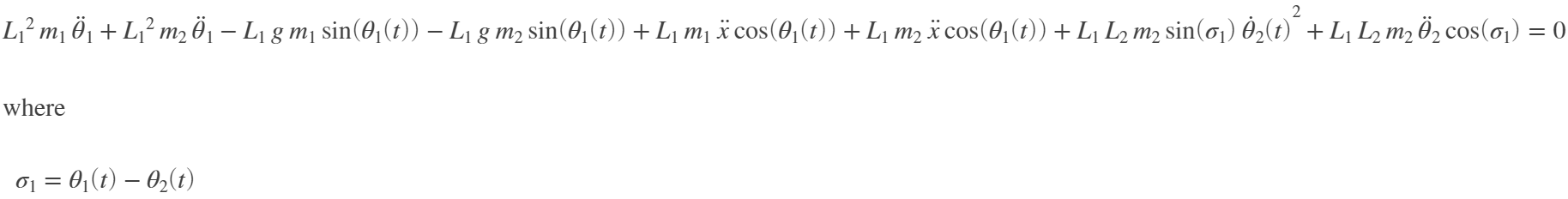
% : q = theta1, Q = 0

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]) == 0

eqn\_theta1(t) =



% : 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

eqn\_theta2(t) = 

% 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]);

% Lagrange Equation of motion, state space form

fprintf('\n')

disp('Lagranges Equation of motion in state space form are given by')

Lagranges Equation of motion in state space form are given by

x1\_dot = x4

x1\_dot = 

x2\_dot = x5

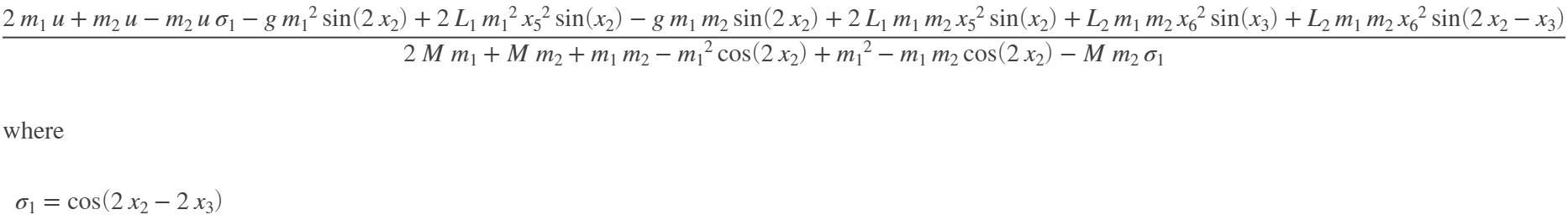
x2\_dot = 

x3\_dot = x6

x3\_dot = 

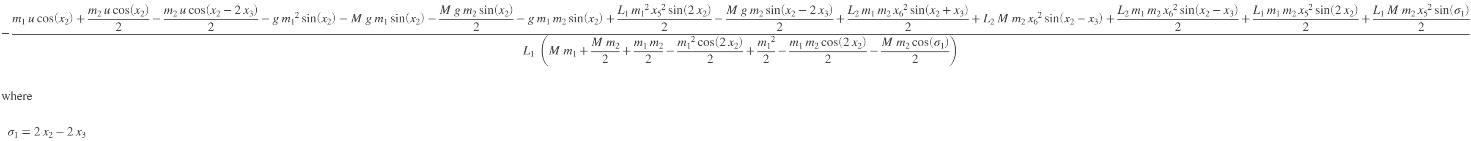
x4\_dot = subs(simplify(sys\_eqn.x\_ddot),[x theta1 theta2 x\_dot theta\_dot\_1 theta\_dot\_2],[x1 x2 x3 x4 x5 x6])

x4\_dot =



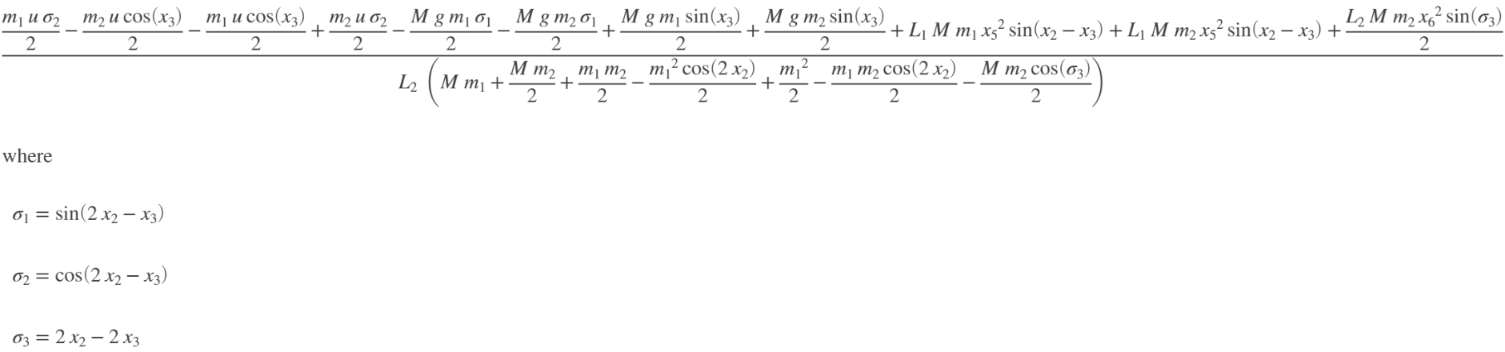
x5\_dot = subs(simplify(sys\_eqn.theta\_ddot\_1),[x theta1 theta2 x\_dot theta\_dot\_1 theta\_dot\_2],[x1 x2 x3 x4 x5 x6])

x5\_dot =



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])

x6\_dot =



# Part 2 Linearized Model Derivation

% Numerical 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;

% 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 = jacobian(f,[x1;x2;x3;x4;x5;x6]);

df\_du = jacobian(f,u);

dh\_dx = jacobian(h,[x1;x2;x3;x4;x5;x6]);

dh\_du = jacobian(h,u);

% Equilibirum pair - origin

xe = zeros(6,1);

ue = 0;

fprintf('\n')

disp('The linearized state space model about the origin is')

The linearized state space model about the origin is

% Linearized Model

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

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;ue;m1;L1;m2;L2;M;g],[xe;ue;m1\_num;L1\_num;m2\_num;L2\_num;M\_num;g\_num]))

B = 6×1

0

0

0

0.6667

-1.3333

0

C = double(dh\_dx)

C = 3×6

1 0 0 0 0 0

0 1 0 0 0 0

0 0 1 0 0 0

D = double(dh\_du)

D = 3×1

0

0

0

# Problem 3 Linearized Model

% Controllability Matrix

Co = ctrb(A,B);

% Observability Matrix

Ob = obsv(A,C);

% Number of states

n = length(A);

% Verify system is reachable & observable

if rank(Co) == n

disp('Pair (A,B) of the linearized model is reachable/controllable')

end

Pair (A,B) of the linearized model is reachable/controllable

if rank(Ob) == n

disp('Pair (A,C) of the linearized model is observable')

end

Pair (A,C) of the linearized model is observable

% Inverse of Controllability Matrix

Co\_inv = inv(Co);

% Last row of inverse of Controllability Matrix

q1 = Co\_inv(end,:);

% Transformation matrix used to get CCF

T\_ccf = [q1;q1\*A;q1\*A^2;q1\*A^3;q1\*A^4;q1\*A^5];

% Transform System into Controller Canonical Form

disp('The Linearized Model in Controller Form is')

The Linearized Model in Controller Form is

A\_ccf = T\_ccf\*A\*inv(T\_ccf)

A\_ccf = 6×6

103 ×

0 0.0010 0 -0.0000 0 -0.0000

0 0 0.0010 0 0.0000 0

0 0 0 0.0010 0 0.0000

0 0 0.0000 0 0.0010 0

0 0 0 0 0 0.0010

0 0 -1.1762 0 0.0981 0

B\_ccf = T\_ccf\*B

B\_ccf = 6×1

0

0

0

-0.0000

0

1.0000

C\_ccf = C\*inv(T\_ccf)

C\_ccf = 3×6

427.7160 0 -54.5000 0 0.6667 0

0 0 43.6000 0 -1.3333 0

0 0 43.6000 0 0.0000 0

D\_ccf = D

D\_ccf = 3×1

0

0

0

% Controllability Matrix of (A',C')

Co\_dual = ctrb(A',C');

% Row reduced echelon form of Controlability, get indice of pivots

[co\_ref,p] = rref(Co\_dual);

% Form L martix from inspection - L = [c1',A'\*c1', c2',A'\*c2', c3', A'\*c3']

L\_obs = [C(1,:)', A'\*C(1,:)',C(2,:)', A'\*C(2,:)',C(3,:)', A'\*C(3,:)'];

% Inverse of L

inv\_L = inv(L\_obs);

% Observability Matrices from inspection

d1\_obs = 2;

d2\_obs = 2;

d3\_obs = 2;

% Vectors needed for transformation matrix

q1 = inv\_L(d1\_obs,:);

q2 = inv\_L(d2\_obs + d1\_obs,:);

q3 = inv\_L(d3\_obs + d2\_obs + d1\_obs,:);

% Form Transformation Matrix

T\_obs = [q1; q1\*A'; q2; q2\*A'; q3; q3\*A'];

% Transform linear system into observer form

disp('The Linearized Model in Observer Form is')

The Linearized Model in Observer Form is

A\_obs = (T\_obs\*A'\*inv(T\_obs))'

A\_obs = 6×6

0 0 0 -8.1750 0 0

1.0000 0 0 0 0 0

0 0 0 65.4000 0 -29.4300

0 0 1.0000 0 0 0

0 0 0 -32.7000 0 32.7000

0 0 0 0 1.0000 0

C\_obs = (T\_obs\*C')'

C\_obs = 3×6

0 1 0 0 0 0

0 0 0 1 0 0

0 0 0 0 0 1

B\_obs = (B'\*inv(T\_obs))'

B\_obs = 6×1

0.6667

0

-1.3333

0

0

0

D\_obs = D

D\_obs = 3×1

0

0

0

# Problem 4 Transfer Function

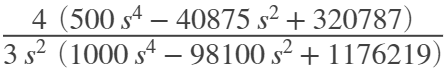
% Transfer Function Matrix Equation

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

disp('Transfer function for X to u:')

Transfer function for X to u:

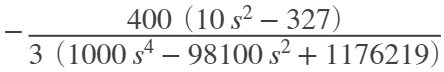
disp(simplify(Y\_U(1)))



disp('Trasnfer Function for theta\_1 to u:')

Transfer Function for theta\_1 to u:

disp(Y\_U(2))



disp('Transfer Function for theta\_2 to u:')

Transfer Function for theta\_2 to u:

disp(Y\_U(3))



# Problem 5 - Part 1 Pendulum Simulation

% Initial Conditions [m, rad, rad, m/s, rad/s, rad/s]

x0 = [0 .01 .02 0 0 0]';

% Time interval

dt = 1/200;

% time and input

time = (0:dt:10)';

u = 0;

% ODE45 Function Call

[~, X] = ode45(@(t,x) DIPC(t,x,u,m1\_num,m2\_num,M\_num,L1\_num,L2\_num,g\_num),time,x0);

# Problem 5 Part - 2 Animation

% Cart width and height

w = 1;

h = .5;

% Graphics handle - cart

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(-2,2,strcat(['Time = ',' ', num2str(time(1)), ' [s]']));

% Define axis limits

axis([-2\*(L1\_num + L2\_num), 2\*(L1\_num + L2\_num),-2\*(L1\_num + L2\_num), 2\*(L1\_num + L2\_num)]);

grid on

xlabel('X [m]')

ylabel('Y [m]')

title('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);

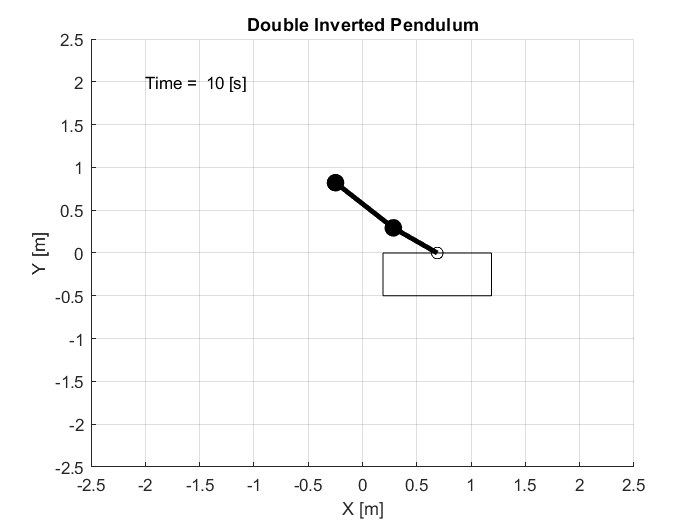
set(h\_txt,'String',strcat(['Time = ',' ', num2str(time(i)), ' [s]']));

drawnow;

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

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

end



close(vidobj);

# Function for Double Inverted Cart Pendulum

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