M6 - Dynamic 4-Bar Mechanism

Jakob Alsaker - 201808760

May 8, 2022

Contents

1 Introduction

A 4 bar mechanism is shown on figure 1 below.

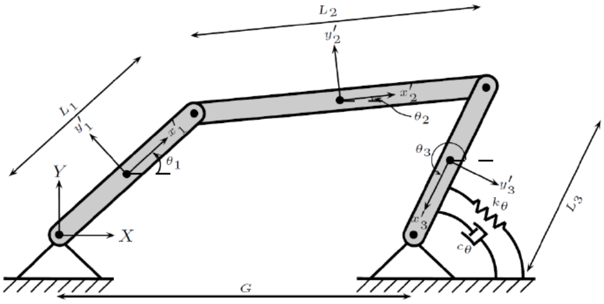

Figure 1: A 4 bar mechanism with a rotational spring damper element

The mechanis consists of an interconnected chain of 3 bars that are ttached to ground at the 2 'free' ends. The model is tied together by 4 ervolute join constraints. The mass of the left most lnk is negliglible, which means it can be replaced by an absolute distance constraint. A rotational spring-damper element is placed between ground and the right most body. The aim of the following set of exercises is to make a parametrix model of the mechanism in Matlab, which means that the model should be able to analyze mechanism with arbitraty bar lengths, and spring-damper properties, k_θ and c_{theta} respectively.

2 Exercises

a. Calculate the constraint jacobian Φ_q

b. Set up the acceleration equation γ

c. Set up a valid initial configuration and displace it to a configuration that gives substantial forces in the spring. Assume zero initial velocities for all links and run a simulation forward in time until the mechanism reaches rest.

d. plot position, velocities and acceleration vs time for all generalized coordinates

e. plot the spring and damper forces vs time

f. Plot the reaction forces in the revolute join between link 2 and link 3

3 Dynamics

3.1 Theory

Looking at the dynamics of the pendulum it is wished to solve for Newton's equation of motion for a differential mass which is described as

$$\ddot{r}^P \cdot dm(P) = F(P) \cdot dm(P) + [\int_m f(P,R)dm(R)]dm(P) \tag{3.1}$$

The equation is difficult to use as it explicitly involves internal forces which are unknown. The equation is also applied for every differential element of the mass which create too many equation of motion. Therefore to simplify the equation is to use the variations or virtual work approach. By using both theories the variational form of the equations of planar motion can be written as

$$\delta q^T[M\ddot{q} - Q] = 0 \tag{3.2}$$

where

$$M = diag(m,m,J') = \begin{bmatrix} m_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & m_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & J_1' & 0 & 0 & 0 \\ 0 & 0 & 0 & m_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & m_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & J_2' \end{bmatrix} \tag{3.3}$$

where J' is the polar mass moment of inertia. Q is only the applied force as the internal force is not known and removed by saying that the condition holds for any virtual displacement that is consistent with the set of constraint in the system. The condition is true if

$$\Phi_q \delta q = 0 \tag{3.4}$$

The last issue is to change the virtual work which can be done by applying Lagrange Multiplier Theorem which shows a unique linear correlation between A and b if $x^T b = 0$ as soon as $Ax = 0$. b is a linear combination of the rows of a and can be written as $b = -A^T\lambda$ The equation of motion can now be written as

$$M\ddot{q} + \Phi_q^T \lambda = Q_A \tag{3.5}$$

By combining it with the one of the kinematic constraints

$$\Phi(q,t) = 0 \tag{3.6}$$

$$\Phi_q \cdot \dot{q} = \nu$$

$$\Phi\ddot{q} = \gamma$$

a linear system can be made and solved by using a numerical integrator such as Runge-Kutta

$$\begin{bmatrix} M & \Phi_q^T \\ \Phi_q & 0 \end{bmatrix} \begin{bmatrix} \ddot{q} \\ \lambda \end{bmatrix} = \begin{bmatrix} Q^A \\ \gamma \end{bmatrix} \tag{3.7}$$

3.2 Constraints

The 4 bar mechanism seen on figure 1, is one body having 6 generalized cartesian coordinates. By applying absolute constraints on the x- and y-axis to the ground, the system have 1 DOF
By applying absolute constraint on body 3 in the x- and y-axis to ground with constant distance two DOF are constrained Applying an revolute joint between body 2 and 3 constrain 2 DOF. Furthermore to have 1 DOF, a absolute constraint between the origin of the global coordinate and body 2 is made.

$$DOF = nb - nh = 6 - 5 = 1 \tag{3.8}$$

The constraint is defined as

$$\Phi_k = \begin{bmatrix} \left(x_2 - \frac{L_2 \cos(\theta_2)}{2}\right)^2 - {L_1}^2 + \left(y_2 - \frac{L_2 \sin(\theta_2)}{2}\right)^2 \\ x_2 - x_3 + \frac{L_2 \cos(\theta_2)}{2} + \frac{L_3 \cos(\theta_3)}{2} \\ y_2 - y_3 + \frac{L_2 \sin(\theta_2)}{2} + \frac{L_3 \sin(\theta_3)}{2} \\ x_3 - G + \frac{L_3 \cos(\theta_3)}{2} \\ y_3 + \frac{L_3 \sin(\theta_3)}{2} \end{bmatrix} \tag{3.9}$$

In this case Φ_k is the same as the vector in the absolute frame pointing to CG of the body in the local frame. To find the Jacobian matrix, the following equation and the generalized coordinates can be used

$$\Phi_q = \frac{\partial \Phi_i(\mathbf{q})}{\partial q_j} = \begin{bmatrix} 2\,x_2 - L_2\,\cos(\theta_2) & 2\,y_2 - L_2\,\sin(\theta_2) & L_2\,\sin(\theta_2)\left(x_2 - \frac{L_2\,\cos(\theta_2)}{2}\right) - L_2\,\cos(\theta_2)\left(y_2 - \frac{L_2\,\sin(\theta_2)}{2}\right) & 0 & 0 & 0 \\ 1 & 0 & -\frac{L_2\,\sin(\theta_2)}{2} & -1 & 0 & -\frac{L_3\,\sin(\theta_3)}{2} \\ 0 & 1 & \frac{L_2\,\cos(\theta_2)}{2} & 0 & -1 & \frac{L_3\,\cos(\theta_3)}{2} \\ 0 & 0 & 0 & 1 & 0 & -\frac{L_3\,\sin(\theta_3)}{2} \\ 0 & 0 & 0 & 0 & 1 & \frac{L_3\,\cos(\theta_3)}{2} \end{bmatrix} \tag{3.10}$$

As there is no driving constraint, ν is a zero matrix. γ can be found by applying the chain rule on ν the equation will be derived as

$$\gamma = \Phi_q \cdot \ddot{q} = -(\Phi_q \cdot \dot{q})_q \cdot \dot{q} + \Phi_{qt} \cdot \dot{q} + \Phi_{tq} \cdot \dot{q} - \Phi_{tt} = -(\Phi_q \cdot \dot{q})_q \cdot \dot{q} - \Phi_{tt} = \tag{3.11}$$

$$\begin{bmatrix} -\dot{y}_2\left(2\,\dot{y}_2 - L_2\,\dot{\theta}_2\,\cos(\theta_2)\right) - \dot{x}_2\left(2\,\dot{x}_2 + L_2\,\dot{\theta}_2\,\sin(\theta_2)\right) - \dot{\theta}_2\left(\dot{\theta}_2\left(\frac{{L_2}^2\cos(\theta_2)^2}{2} + \frac{{L_2}^2\sin(\theta_2)^2}{2} + L_2\,\sin(\theta_2)\left(y_2 - \frac{L_2\,\sin(\theta_2)}{2}\right) + L_2\,\cos(\theta_2)\left(x_2 - \frac{L_2\,\cos(\theta_2)}{2}\right)\right) + L_2\,\dot{x}_2\,\sin(\theta_2) - L_2\,\dot{y}_2\,\cos(\theta_2)\right) \\ \frac{L_2\,\cos(\theta_2)\,\dot{\theta}_2^2}{2} + \frac{L_3\,\cos(\theta_3)\,\dot{\theta}_3^2}{2} \\ \frac{L_2\,\sin(\theta_2)\,\dot{\theta}_2^2}{2} + \frac{L_3\,\sin(\theta_3)\,\dot{\theta}_3^2}{2} \\ \frac{L_3\,\dot{\theta}_3^2\,\cos(\theta_3)}{2} \\ \frac{L_3\,\dot{\theta}_3^2\,\sin(\theta_3)}{2} \end{bmatrix}$$

The kinematic constraint are now defined.

To find the mass matrix, equation (3.3) can be applied for one body. As the body is a thin rectangular rod where it is constrained in the end, the polar mass moment of inertia can be defined as $J' = \frac{m \cdot l^2}{12}$ The applied force in the equation (3.7) is gravity in the y-axis and a rotational spring-damper force in the angle as no other force is applied.

$$Q^A = \begin{bmatrix} 0 \\ -m_1 \cdot g \\ 0 \\ 0 \\ -m_2 \cdot g \\ -n \end{bmatrix} \tag{3.12}$$

where $n = K_0 \cdot (q(6) - q(6)_{start}) + C0 \cdot \dot{q}(6)$

To solve for the acceleration in the equation of motion it can be rewritten as

$$\begin{bmatrix} \ddot{q}_{6x1} \\ \lambda_{5x1} \end{bmatrix} = \begin{bmatrix} M_{6x6} & \Phi^T_{q(6x5)} \\ \Phi_{q(5x6)} & 0_{5x5} \end{bmatrix}^{-1}_{11x11} \begin{bmatrix} Q^A \\ \gamma \end{bmatrix}_{11x1} \tag{3.13}$$

Furthermore as the constraint forces using the Lagrange multiplier theorem is neglected, they can be found by following equation

$$F_{constraint(6x1)} = -\Phi^T_{q(6x5)} \cdot \lambda_{(5x1)} \tag{3.14}$$

Equation (3.14) can be used for the constraint forces for the generalized coordinated but if wanting to look at the constraint forces in a connection, the following equation can be written as

$$F_i^{nk} = -C_i^T A_i^T \Phi_{ri}^{k\ T} \lambda^k \tag{3.15}$$

It should be noted at some situation such as this, the C- and A-matrix will be the identity matrix.

To find the torsion the equation can be written as

$$T_i^{nk} = (s_i'^{PT} B_i^T \Phi_{ri}^{k\ T} - \Phi_{\phi i}^{k\ T})\lambda^k \tag{3.16}$$

4 Numerical Integration

To solve the equation of motion, a numerical integration solver has to be used. A option is Runge-Kutta which finds an approximation of non linear equations. By applying an initial problem such as:

$$\frac{dy}{dt} = f(t,y) \tag{4.1}$$

using an initial start guesses $y(t_0) = y_0$ where y_0 represent all unknown variables. The integrator uses Euler's method 4 times and weight averaging the guesses.

$$y_{i+1} = y_i + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4) \tag{4.2}$$

where h is the step which can be defined by δt. Each k guess is defined as:

$$\begin{aligned} k_1 &= f(t_i, y_i) \\ k_2 &= f(t_i + \frac{h}{2}, y_i + h\frac{k_1}{2} \\ k_3 &= f(t_i + \frac{h}{2}, y_i + h\frac{k_2}{2} \\ k_4 &= f(t_i + h, y_i + hk_3) \end{aligned} \tag{4.3}$$

5 Plots

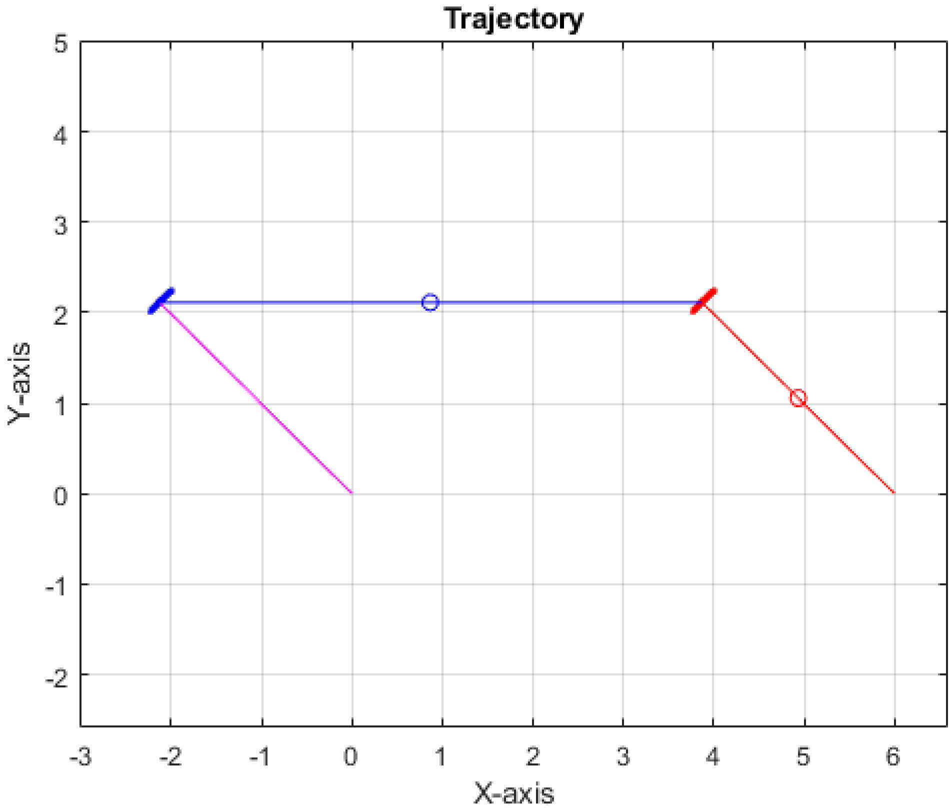

Figure 2: Trajectory

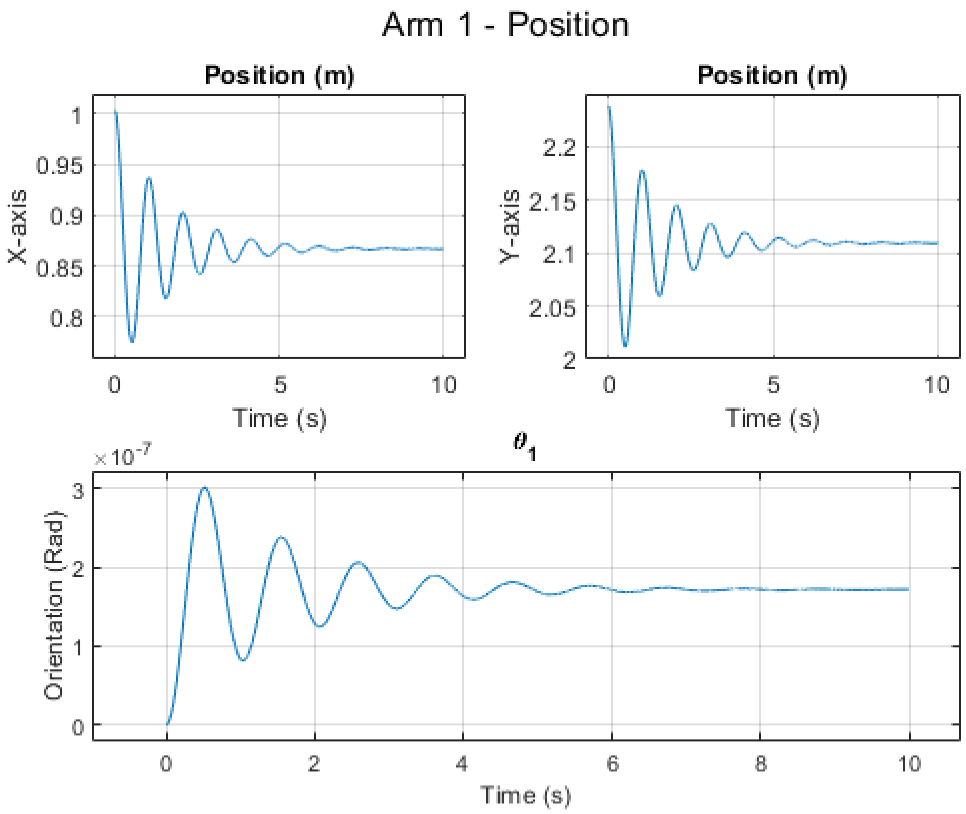

Figure 3: Position of arm 1

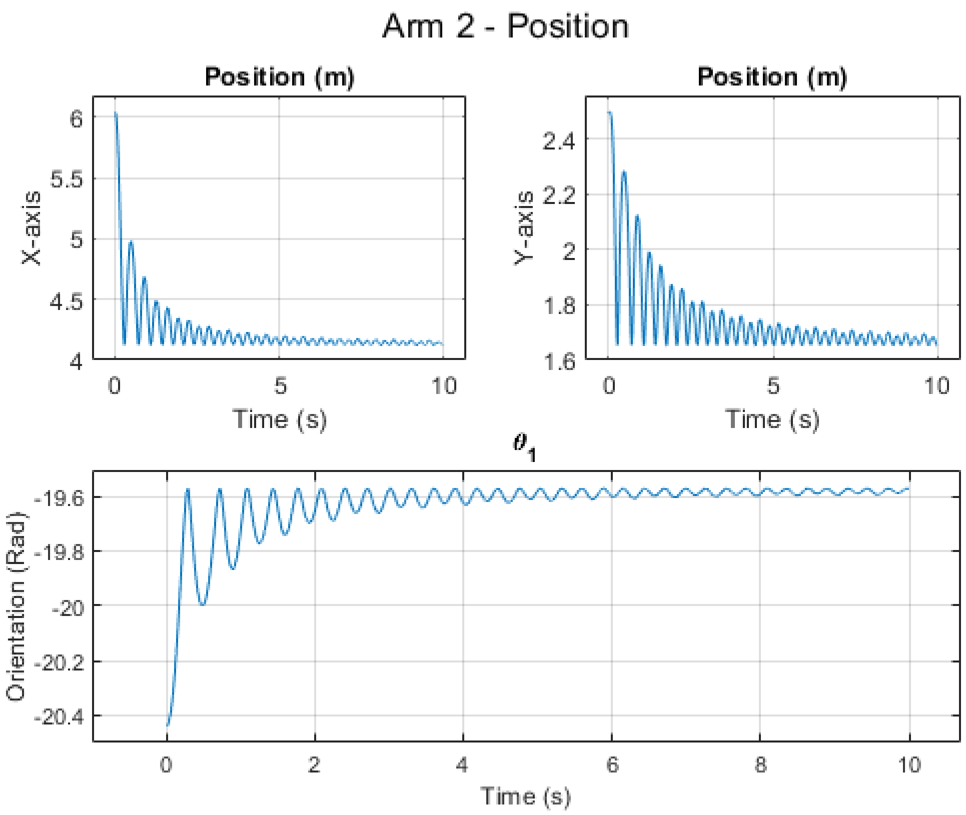

Figure 4: Position of arm 2

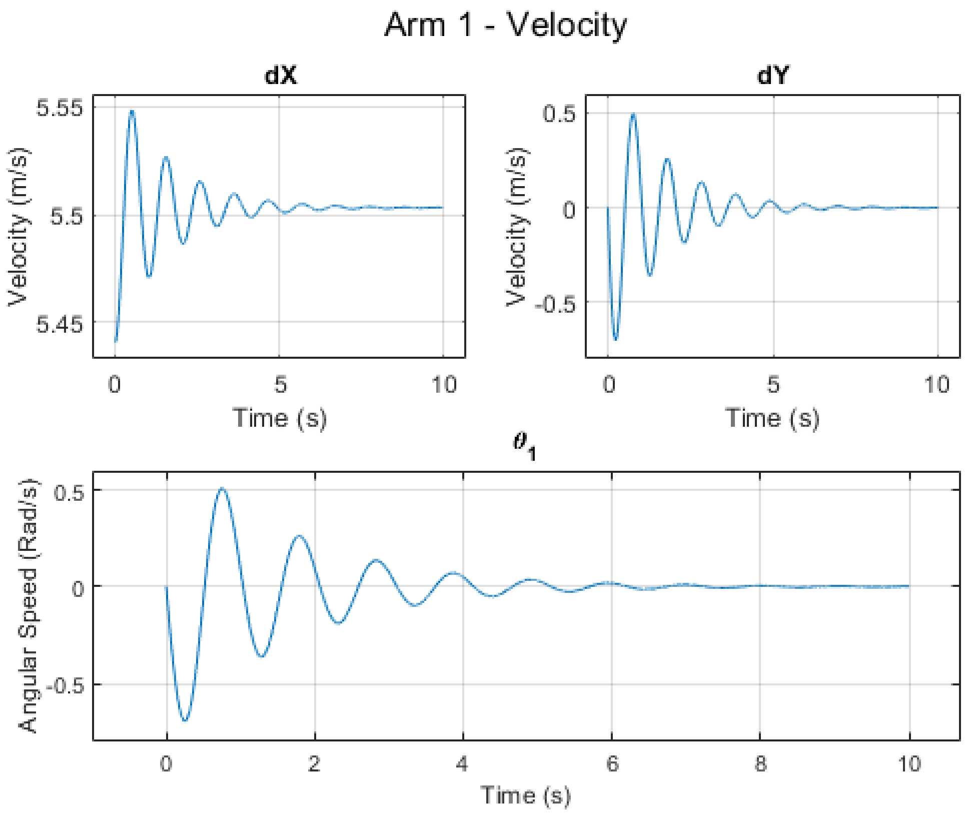

Figure 5: Velocity of arm 1

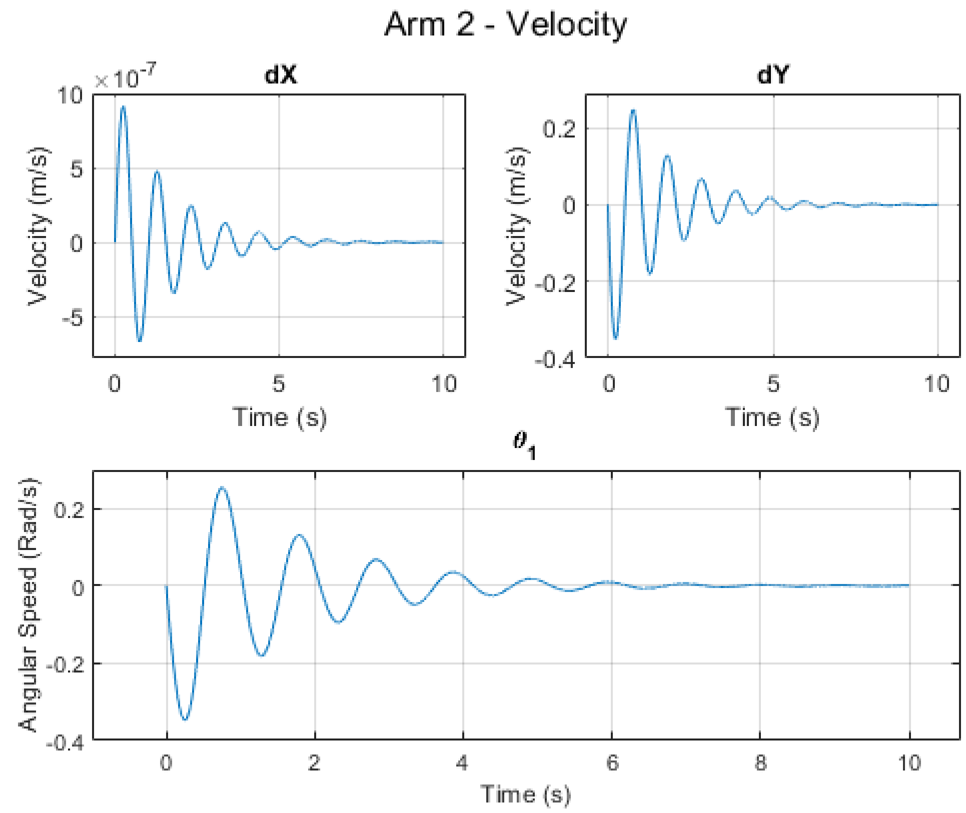

Figure 6: Velocity of arm 2

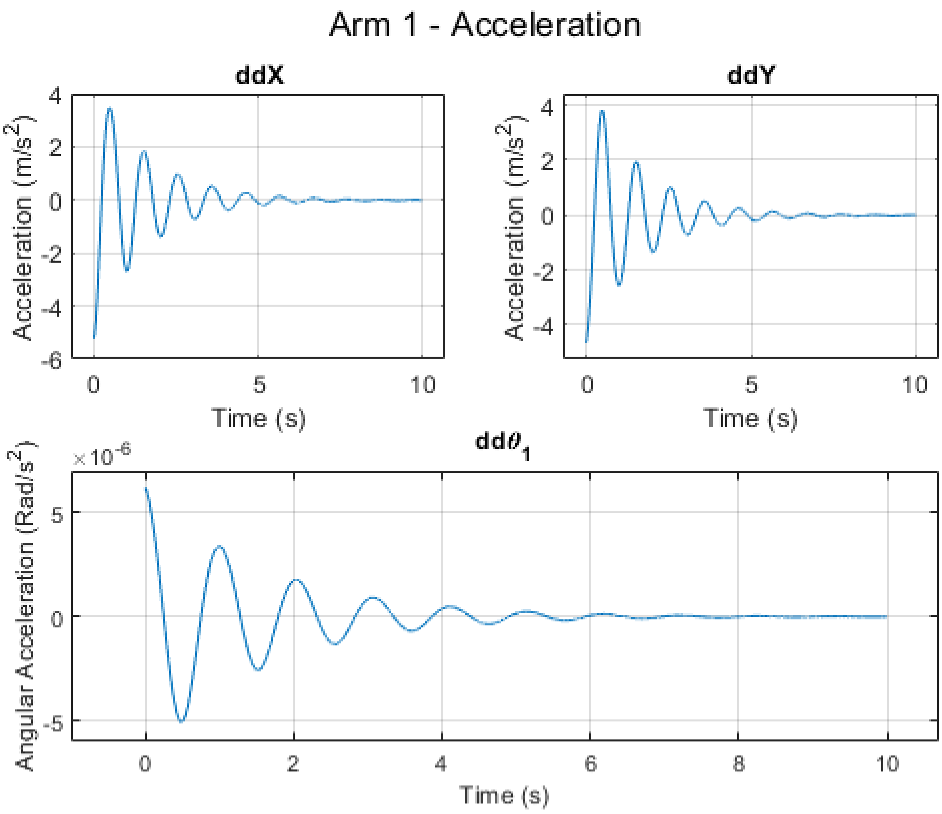

Figure 7: Acceleration of arm 1

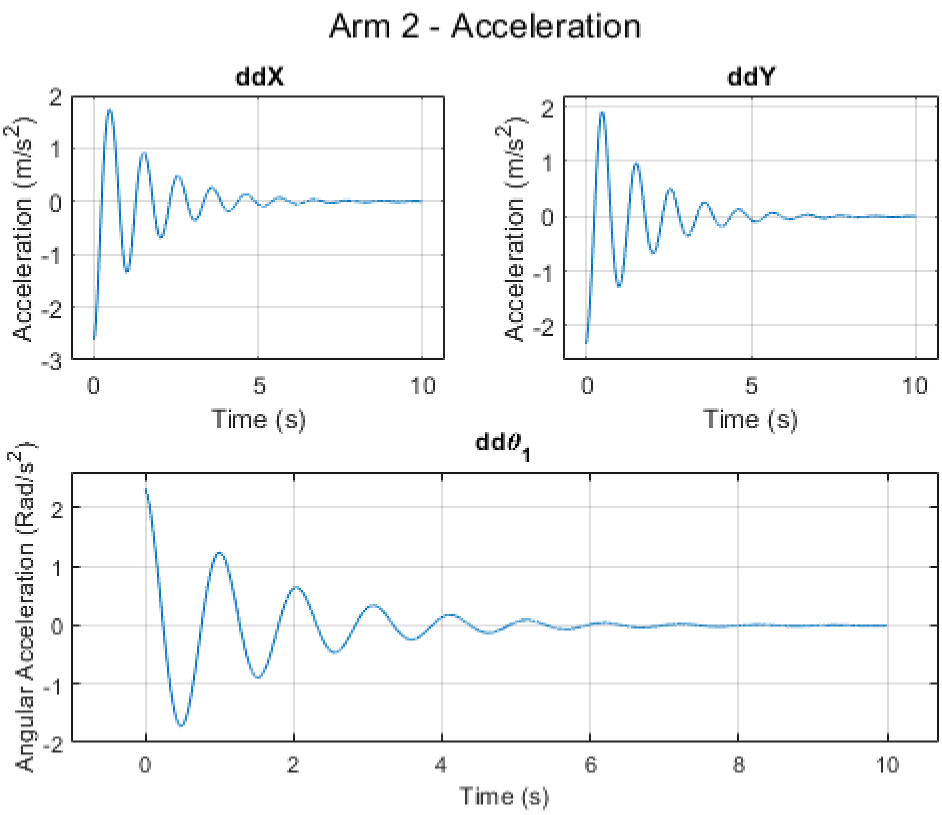

Figure 8: Acceleration of arm 2

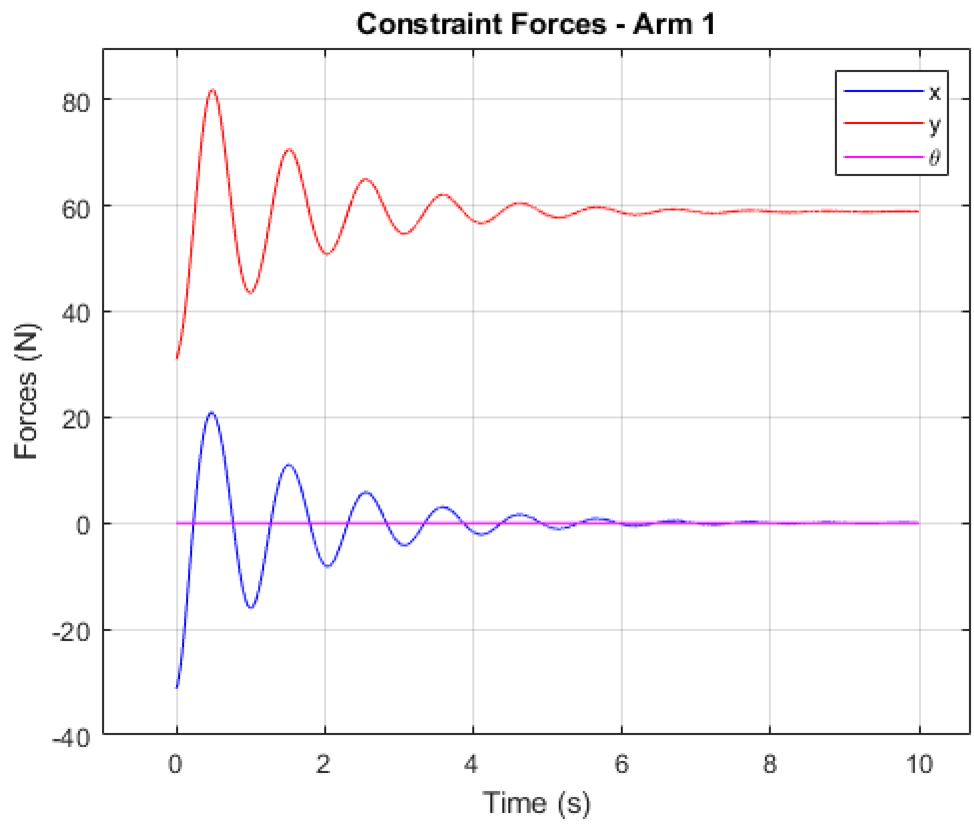

Figure 9: Constraint forces of arm 1

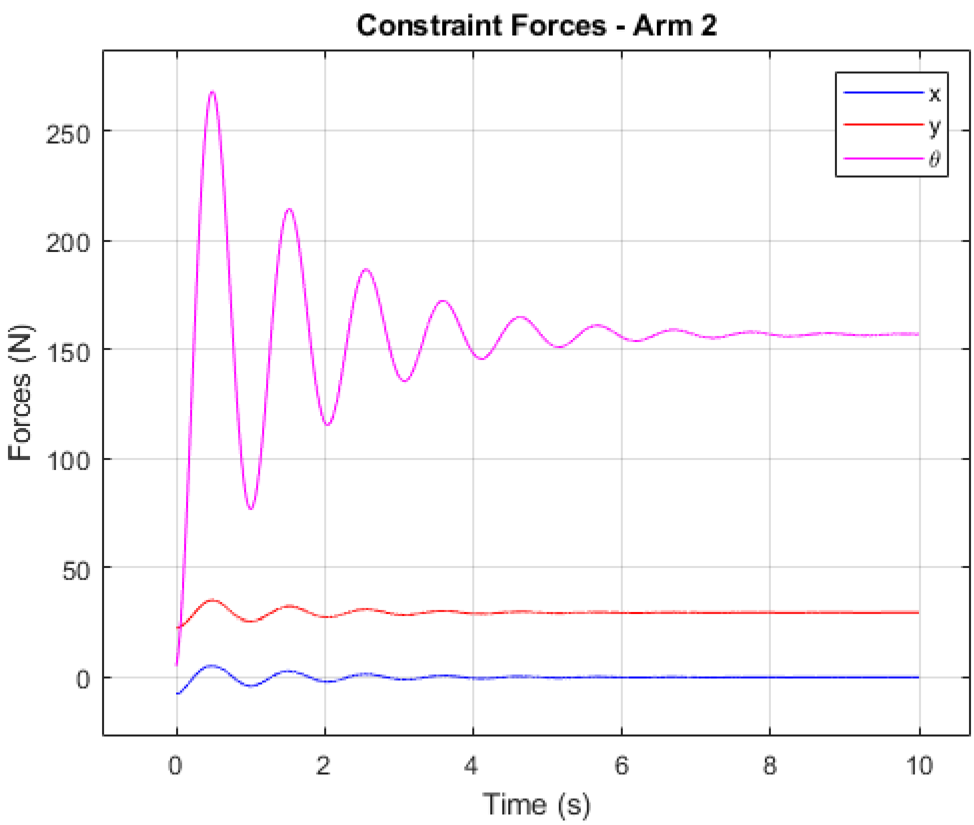

Figure 10: Constraint forces of arm 2

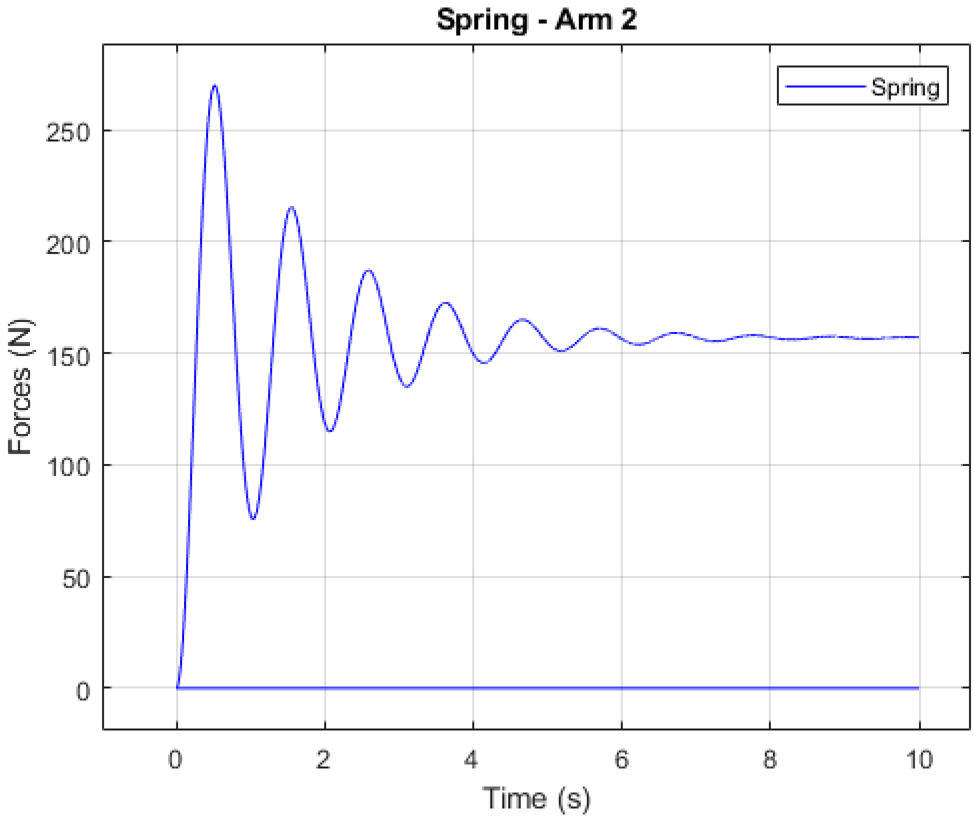

Figure 11: Spring Forces

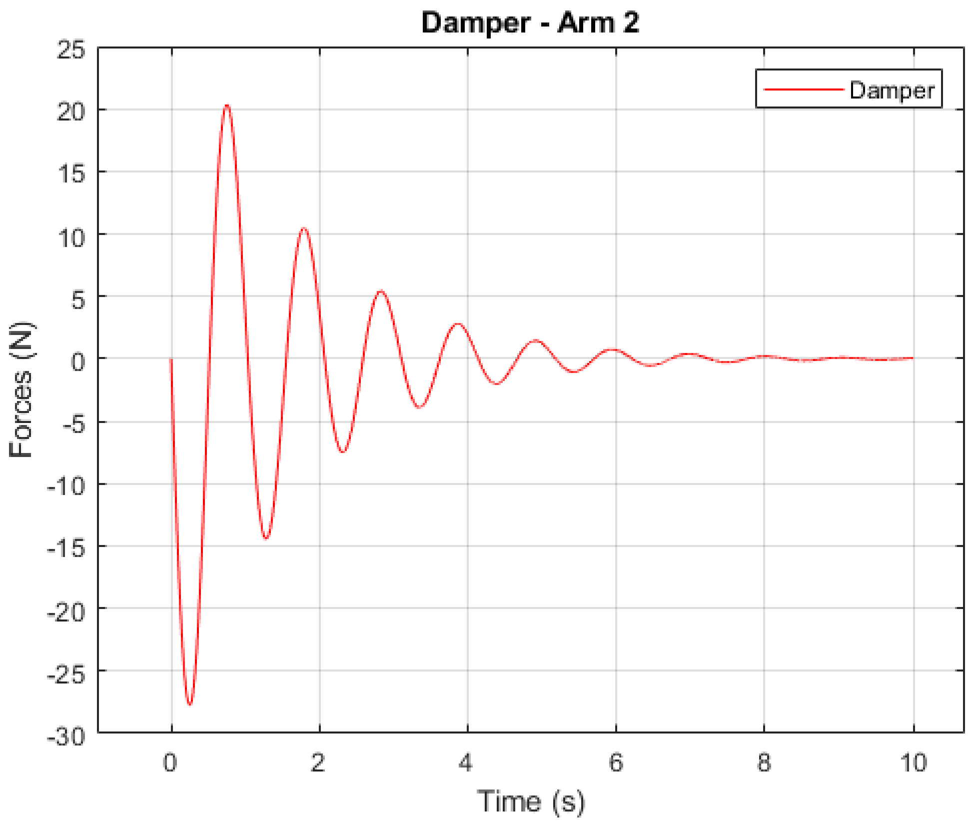

Figure 12: Damper Forces

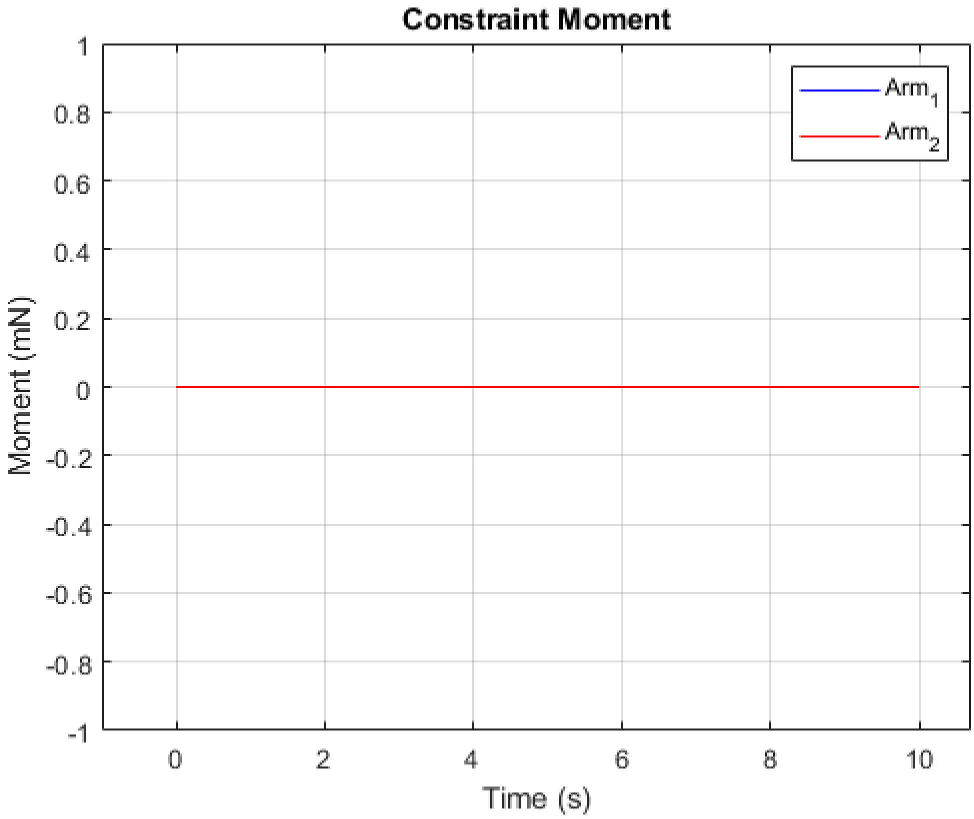

Figure 13: Moment

6 Appendix

6.1 Main Code

```
close all; clear all;
tspan = 0:0.001:10;
params = params;
L_2 = params.L2;
L_3 = params.L3;
y0 = zeros(1,17);
%q_guess = [-2; 0.5; pi/4; 1; 1; pi/10];

grashof = [params.L1,params.L2,params.L3,params.G];
if max(grashof)+min(grashof) < sum(grashof)-max(grashof)+min(grashof) % See if Grashof ...
    is satisfied
    %q_guess = NewtonRaphston(tspan(1),q_guess)
else
    error("Grashof is not satisfied")
end
q_guess =  [1.0036; 2.2393; 0; 5.0018; 1.1197; 5.4405];
y0 = [q_guess;zeros(11,1)];
% Runge Kutta with correction
opts = odeset; opts=odeset(opts,'RelTol',1e-13,'AbsTol',1e-14);
[t y] = ode45(@(t,y) EOM(t,y,params),tspan,y0',opts);
% Position, velocity, acceleration
q = y(:,1:6);
q_dot = y(:,6:12);
B = @(theta) [-sin(theta) -sin(theta); cos(theta) -sin(theta)];
% Acceeleration, Constraint Forces, Constraint Torso
dy = zeros(length(tspan),17);
lambda = zeros(5,length(tspan));
for i = 1:length(tspan)
    dy(i,:) = EOM(tspan(i),y(i,:)',params);
    lambda(:,i) = dy(i,13:17);
    CF(i,:) = -Phi_q(params,q(i,:)).'*lambda(:,i);
    phik = Phi_q(params,q(i,:));
    CT(i,1) = ([L_2/2;0].'*B(q(i,3)).'*phik(2:3,1:2).'-phik(2:3,3).')*lambda(2:3,i);
    CT(i,2) = ([-L_3/2;0].'*B(q(i,6)).'*phik(2:3,4:5).'-phik(2:3,6).')*lambda(2:3,i);
end
q_ddot = dy(:,7:12);

% Constraint forces - Arm 1
figure(14)
plot(t,CT(:,1),'b',t,CT(:,2),'r')
hold on
axis padded
grid on
legend('Arm_1','Arm_2')
title('Constraint Moment')
ylabel('Moment (mN)')
xlabel('Time (s)')
hold off


% Spring-Damper Forces
K0 = params.K0;
C0 = params.C0;
Spring_Force = zeros(length(tspan));
Damper_Force_Force = zeros(length(tspan));
for i = 1:length(tspan)
    Spring_Force(i) = K0*(q(i,6)-y0(6));
    Damper_Force(i) = C0*q_dot(i,6);
end

% Getting points
A = @(theta) [cos(theta) -sin(theta); sin(theta) cos(theta)];
for i = 1:length(tspan)
    P1(i,:) = [q(i,1);q(i,2)] + A(q(i,3))*[-L_2/2;0]; % Second arm left point
    P2(i,:) = [q(i,1);q(i,2)] + A(q(i,3))*[L_2/2;0];  % Second arm right point
    P3(i,:) = [q(i,4);q(i,5)] + A(q(i,6))*[-L_3/2;0]; % Third arm left point
    P4(i,:) = [q(i,4);q(i,5)] + A(q(i,6))*[L_3/2;0];  % Third arm right point
```

```
end

% figure(1)% Live Trajectory
% hold on
% grid on
% title('Live Trajectory')
% xlabel('X-axis')
% ylabel('Y-axis')
% xlim([-4 8])
% ylim([-4 5])
% tic
% for p = 1:length(tspan)
%     Body1 = line([P1(p,1) P2(p,1)], [P1(p,2) P2(p,2)]);
%     Body1.Color = 'r';
%     Body2 = line([P3(p,1) P4(p,1)], [P3(p,2) P4(p,2)]);
%     Body2.Color = 'b';
%     Body3 = line([0 P1(p,1)], [0 P1(p,2)]);
%     Body3.Color = 'm';
%     pause(0.000001)
%     delete(Body1);
%     delete(Body2);
%     delete(Body3);
% end
% toc
% plot(q(p,1),q(p,2),'ro-');
% plot(q(p,4),q(p,5),'bo-');
% Body1 = line([P1(end,1) P2(end,1)], [P1(end,2) P2(end,2)]);
% Body1.Color = 'r';
% Body2 = line([P3(end,1) P4(end,1)], [P3(end,2) P4(end,2)]);
% Body2.Color = 'b';
% Body3 = line([0 P1(end,1)], [0 P1(end,2)]);
% Body3.Color = 'm';
% hold off
%
% Trajectory
% figure(2)
% plot(P1(:,1),P1(:,2),'.','Color','b');
% hold on
% plot(P2(:,1),P2(:,2),'b');
% plot(P3(:,1),P3(:,2),'.','Color','r');
% plot(P4(:,1),P4(:,2),'r');
% plot(q(end,1),q(end,2),'bo-');
% plot(q(end,4),q(end,5),'ro-');
% axis equal
% grid on
% axis padded
% title('Trajectory')
% xlabel('X-axis')
% ylabel('Y-axis')
% Body3 = line([0 P1(end,1)], [0 P1(end,2)]);
% Body3.Color = 'm';
% Body1 = line([P1(end,1) P2(end,1)], [P1(end,2) P2(end,2)]);
% Body1.Color = 'b';
% Body2 = line([P3(end,1) P4(end,1)], [P3(end,2) P4(end,2)]);
% Body2.Color = 'r';
% hold off

% Constraint forces - Arm 1
% figure(9)
% plot(t,CF(:,1),'b',t,CF(:,2),'r',t,CF(:,3),'m')
% hold on
% axis padded
% grid on
% legend('x','y','\theta')
% title('Constraint Forces - Arm 1')
% ylabel('Forces (N)')
% xlabel('Time (s)')
% hold off
%
% % Constraint forces - Arm 2
% figure(10)
% plot(t,CF(:,4),'b',t,CF(:,5),'r',t,CF(:,6),'m')
% hold on
```

```
% axis padded
% grid on
% legend('x','y','\theta')
% title('Constraint Forces - Arm 2')
% ylabel('Forces (N)')
% xlabel('Time (s)')
% hold off

% Damper Forces
figure(15)
plot(t,Damper_Force,'r')
hold on
axis padded
grid on
legend('Damper')
title('Damper - Arm 2')
ylabel('Forces (N)')
xlabel('Time (s)')
hold off

% Spring Force
% figure(11)
% plot(t,Spring_Force,'b')
% hold on
% axis padded
% grid on
% legend('Spring')
% title('Spring - Arm 2')
% ylabel('Forces (N)')
% xlabel('Time (s)')
% hold off

% % Position - Arm 1
% figure(3)
% sgtitle('Arm 1 - Position')
% subplot(2,2,1,'Color','r')
% plot(t,q(:,1))
% grid on
% axis padded
% title('Position (m)')
% xlabel('Time (s)')
% ylabel('X-axis')
% subplot(2,2,2)
% plot(t,q(:,2))
% grid on
% axis padded
% title('Position (m)')
% xlabel('Time (s)')
% ylabel('Y-axis')
% subplot(2,2,[3 4],'Color','b')
% plot(t,q(:,3))
% grid on
% axis padded
% title('\theta_1')
% xlabel('Time (s)')
% ylabel('Orientation (Rad)')
%
% % Velocity - Arm 1
% figure(4)
% sgtitle('Arm 1 - Velocity')
% grid on
% axis padded
% subplot(2,2,1)
% plot(t,q_dot(:,1))
% grid on
% axis padded
% title('dX')
% xlabel('Time (s)')
% ylabel('Velocity (m/s)')
% subplot(2,2,2)
% plot(t,q_dot(:,2))
% grid on
% axis padded
```

```
% title('dY')
% xlabel('Time (s)')
% ylabel('Velocity (m/s)')
% subplot(2,2,[3 4])
% plot(t,q_dot(:,3))
% grid on
% axis padded
% title('\theta_1')
% xlabel('Time (s)')
% ylabel('Angular Speed (Rad/s)')
%
% % Acceleration - Arm 1
% figure(5)
% sgtitle('Arm 1 - Acceleration')
% grid on
% axis padded
% subplot(2,2,1)
% plot(t,q_ddot(:,1))
% grid on
% axis padded
% title('ddX')
% xlabel('Time (s)')
% ylabel('Acceleration (m/s^2)')
% subplot(2,2,2)
% plot(t,q_ddot(:,2))
% grid on
% axis padded
% title('ddY')
% xlabel('Time (s)')
% ylabel('Acceleration (m/s^2)')
% subplot(2,2,[3 4])
% plot(t,q_ddot(:,3))
% grid on
% axis padded
% title('dd\theta_1')
% xlabel('Time (s)')
% ylabel('Angular Acceleration (Rad/s^2)')
%
% % Position - Arm 2
% figure(6)
% sgtitle('Arm 2 - Position')
% subplot(2,2,1,'Color','r')
% plot(t,q(:,4))
% grid on
% axis padded
% title('Position (m)')
% xlabel('Time (s)')
% ylabel('X-axis')
% subplot(2,2,2)
% plot(t,q(:,5))
% grid on
% axis padded
% title('Position (m)')
% xlabel('Time (s)')
% ylabel('Y-axis')
% subplot(2,2,[3 4],'Color','b')
% plot(t,q(:,6))
% grid on
% axis padded
% title('\theta_1')
% xlabel('Time (s)')
% ylabel('Orientation (Rad)')
%
% % Velocity - Arm 2
% figure(7)
% sgtitle('Arm 2 - Velocity')
% grid on
% axis padded
% subplot(2,2,1)
% plot(t,q_dot(:,4))
% grid on
% axis padded
% title('dX')
```

```
% xlabel('Time (s)')
% ylabel('Velocity (m/s)')
% subplot(2,2,2)
% plot(t,q_dot(:,5))
% grid on
% axis padded
% title('dY')
% xlabel('Time (s)')
% ylabel('Velocity (m/s)')
% subplot(2,2,[3 4])
% plot(t,q_dot(:,6))
% grid on
% axis padded
% title('\theta_1')
% xlabel('Time (s)')
% ylabel('Angular Speed (Rad/s)')
%
% % Acceleration - Arm 2
% figure(8)
% sgtitle('Arm 2 - Acceleration')
% grid on
% axis padded
% subplot(2,2,1)
% plot(t,q_ddot(:,4))
% grid on
% axis padded
% title('ddX')
% xlabel('Time (s)')
% ylabel('Acceleration (m/s^2)')
% subplot(2,2,2)
% plot(t,q_ddot(:,5))
% grid on
% axis padded
% title('ddY')
% xlabel('Time (s)')
% ylabel('Acceleration (m/s^2)')
% subplot(2,2,[3 4])
% plot(t,q_ddot(:,6))
% grid on
% axis padded
% title('dd\theta_1')
% xlabel('Time (s)')
% ylabel('Angular Acceleration (Rad/s^2)')
```

6.2 Equation of Motion

```
function dy = EOM(t,y,params)
m2 = params.m2;
m3 = params.m3;
J2 = params.J2;
J3 = params.J3;
g  = params.g;
K0 = params.K0;
C0 = params.C0;
persistent y0;
q = y(1:6);
q_dot = y(7:12);
lambda = y(13:17);


if t == 0
    y0 = q(6);
end
n_1 = K0*(q(6)-y0) + C0*q_dot(6);

M = diag([m2 m2 J2 m3 m3 J3]);                                  % Mass Matrix
Q_A = [0 -m2*g 0 0 -m3*g -n_1].';                                  % Applied Force
A = [M Phi_q(params,q).'; Phi_q(params,q) zeros(5)];            % Combined Matrix
B = [Q_A;Gamma(params,q,q_dot)];                                % Combined Matrix
C = A\B;                                                        % Acc,Lambda
```

```
q_ddot = C(1:11);                                    % Retrieving acc
dy = [q_dot;q_ddot];                                 % Return value
end
```

6.3 Matlab - Symbolic

```
sympref('AbbreviateOutput',false);
syms x_2 y_2 theta_2 x_3 y_3 theta_3 L_1 L_2 L_3 G omega t;
syms x_dot_2 y_dot_2 theta_dot_2 x_dot_3 y_dot_3 theta_dot_3;
syms A(theta)

A(theta) = [cos(theta) -sin(theta); sin(theta) cos(theta)];
r_2 = [x_2;y_2];
r_3 = [x_3;y_3];

sip_2 = [L_2/2;0];
sip_3 = [L_3/2;0];

% Defining Arm 1
C = [0;0];
C_3 = L_1;
rp2 = r_2+A(theta_2)*-sip_2;

phi_k3 = r_3 + A(theta_3)*sip_3-[G;0]                   % Absolute ...
    constraint - L_3
phi_k2 = r_2 + A(theta_2)*sip_2-r_3-A(theta_3)*-sip_3;          % Revolute ...
    constraint - L_3-L_2
phi_k1 = (rp2-C).'*(rp2-C)-C_3^2                        % Absolute ...
    constraint - L_2

% Constraints
phi_K = [phi_k1;phi_k2;phi_k3];                          % Kinematic ...
    constraint
q = [x_2 y_2 theta_2 x_3 y_3 theta_3].'                  % Generalized ...
    cartesian coordinates
phi = [phi_K]                                            % Constraint matrix
phiq = jacobian(phi,q)                                   % Jacobian Matrix
phit = -diff(phi,t)                                      % Nu Matrix
q_dot = [x_dot_2 y_dot_2 theta_dot_2 x_dot_3 y_dot_3 theta_dot_3].';  % Generealized ...
    cartesian velocity coordinates
gamma = -jacobian(phiq*q_dot,q)*q_dot-diff(phit,t)       % Gamma matrix
%making functions of the symbolic
matlabFunction(phi,'file','Phi');
matlabFunction(phiq,'file','Phi_q');
matlabFunction(phit,'file','Nu');
matlabFunction(gamma,'file','Gamma');
```

6.4 Parameters

```
function params = params
params.m2 = 6;
params.m3 = 3;
params.L1 = 3;
params.L2 = 6;
params.L3 = 3;
params.J2 = params.m2/12*params.L2^2;
params.J3 = params.m3/12*params.L3^2;
params.g  = 9.81;
params.G = 6;
params.K0 = 2500;
params.C0 = 80;

% params = params;
% L_2 = params.L2;
% L_3 = params.L3;
% x_2 = q(1);
```

```
% y_2 = q(2);
% theta_2 = q(3);
% x_3 = q(4);
% y_3 = q(5);
% theta_3 = q(6);
% x_dot_2 = q_dot(1);
% y_dot_2 = q_dot(2);
% theta_dot_2 = q_dot(3);
% x_dot_3 = q_dot(4);
% y_dot_3 = q_dot(5);
% theta_dot_3 = q_dot(6);
```

6.5 Φ_q

```
function phiq = Phi_q(params,q)
%PHI_Q
%    PHIQ = PHI_Q(L_2,L_3,THETA_2,THETA_3,X_2,Y_2)

%    This function was generated by the Symbolic Math Toolbox version 8.7.
%    08-May-2022 16:02:29

L_2 = params.L2;
L_3 = params.L3;
theta_2 = q(3);
theta_3 = q(6);
x_2 = q(1);
y_2 = q(2);


t2 = cos(theta_2);
t3 = cos(theta_3);
t4 = sin(theta_2);
t5 = sin(theta_3);
t6 = (L_2.*t2)./2.0;
t7 = (L_3.*t3)./2.0;
t8 = (L_2.*t4)./2.0;
t9 = (L_3.*t5)./2.0;
t10 = -t8;
t11 = -t9;
phiq = ...
    reshape([x_2.*2.0-L_2.*t2,1.0,0.0,0.0,0.0,0.0,y_2.*2.0-L_2.*t4,0.0,1.0,0.0,0.0,0.0,-L_2.*t4.*(t6-x_2)+L_2.*t2.*
```

6.6 γ

```
function gamma = Gamma(params,q,q_dot)
%GAMMA
%    GAMMA = GAMMA(L_2,L_3,THETA_2,THETA_3,THETA_DOT_2,THETA_DOT_3,X_2,X_DOT_2,Y_2,Y_DOT_2)

%    This function was generated by the Symbolic Math Toolbox version 8.7.
%    08-May-2022 16:02:29

L_2 = params.L2;
L_3 = params.L3;
theta_2 = q(3);
theta_3 = q(6);
theta_dot_2 = q_dot(3);
theta_dot_3 = q_dot(6);
x_2 = q(1);
x_dot_2 = q_dot(1);
y_2 = q(2);
y_dot_2 = q_dot(2);

t2 = cos(theta_2);
t3 = cos(theta_3);
t4 = sin(theta_2);
t5 = sin(theta_3);
```

```
t6 = L_2.^2;
t7 = theta_dot_2.^2;
t8 = theta_dot_3.^2;
t9 = (L_3.*t3.*t8)./2.0;
t10 = (L_3.*t5.*t8)./2.0;
gamma = ...
    [-theta_dot_2.*(theta_dot_2.*((t2.^2.*t6)./2.0+(t4.^2.*t6)./2.0+L_2.*t2.*(x_2-(L_2.*t2)./2.0)+L_2.*t4.*
```

6.7 Φ

```
function phi = Phi(G,L_1,L_2,L_3,theta_2,theta_3,x_2,x_3,y_2,y_3)
%PHI
%    PHI = PHI(G,L_1,L_2,L_3,THETA_2,THETA_3,X_2,X_3,Y_2,Y_3)

%    This function was generated by the Symbolic Math Toolbox version 8.7.
%    08-May-2022 16:02:28

t2 = cos(theta_2);
t3 = cos(theta_3);
t4 = sin(theta_2);
t5 = sin(theta_3);
t6 = (L_2.*t2)./2.0;
t7 = (L_3.*t3)./2.0;
t8 = (L_2.*t4)./2.0;
t9 = (L_3.*t5)./2.0;
phi = [(t6-x_2).^2+(t8-y_2).^2-L_1.^2;t6+t7+x_2-x_3;t8+t9+y_2-y_3;-G+t7+x_3;t9+y_3];
```

6.8 Newton Raphton

```
function w = NewtonRaphston(t,q)
es = 0.000000000001;     % Tolerance
maxit = 100000;        % Maximum iteration
iter = 0; % Defining iteration
driving = zeros(1,6);
driving(3) = 1;
while(1)
    Phi1 = [Phi(params,t,q);q(3)*1*t];
    Jac = [Phi_q(q,params); driving];
    dq = Jac\Phi1; % Defining the change
    q = q-dq; % Removing change from previous guess
    iter = iter+1; % Counting Iteration
    ea = 100*max(abs(dq./q)); % Looking at error-value
    if(iter≥maxit || ea≤es)  % See if conditions are satisfied
        w = q; % Saving data if satisfied
        break
    end
end
```