

M6 - Dynamic 4-Bar Mechanism

Jakob Alsaker - 201808760

May 8, 2022

Contents

1	Introduction	2
2	Exercises	2
3	Dynamics	3
3.1	Theory	3
3.2	Constraints	3
4	Numerical Integration	4
5	Plots	6
6	Appendix	13
6.1	Main Code	13
6.2	Equation of Motion	17
6.3	Matlab - Symbolic	18
6.4	Parameters	18
6.5	Φ_q	19
6.6	γ	19
6.7	Φ	20
6.8	Newton Raphson	20

1 Introduction

A 4 bar mechanism is shown on figure 1 below.

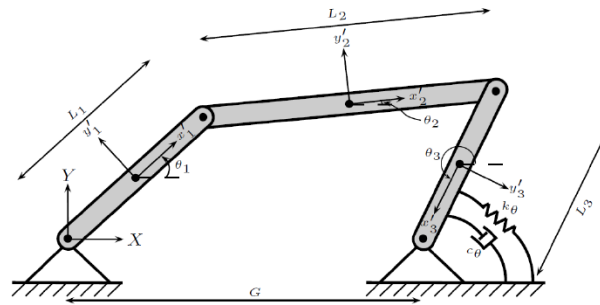


Figure 1: A 4 bar mechanism with a rotational spring damper element

The mechanism consists of an interconnected chain of 3 bars that are attached to ground at the 2 'free' ends. The model is tied together by 4 revolute joint constraints. The mass of the left most link is negligible, 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 parametric model of the mechanism in Matlab, which means that the model should be able to analyze mechanism with arbitrary bar lengths, and spring-damper properties, k_θ and c_{θ} respectively.

2 Exercises

- Calculate the constraint jacobian Φ_q
- Set up the acceleration equation γ
- 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.
- plot position, velocities and acceleration vs time for all generalized coordinates
- plot the spring and damper forces vs time
- Plot the reaction forces in the revolute joint 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

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

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

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

where

$$M = \text{diag}(m, m, J') = \begin{bmatrix} m_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & m_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & J'_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & m_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & m_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & J'_2 \end{bmatrix} \quad (3.3)$$

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

$$\Phi_q \delta q = 0 \quad (3.4)$$

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

$$M\ddot{q} + \Phi_q^T \lambda = Q_A \quad (3.5)$$

By combining it with the one of the kinematic constraints

$$\Phi(q, t) = 0 \quad (3.6)$$

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

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

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

$$\begin{bmatrix} M & \Phi_q^T \\ \Phi_q & 0 \end{bmatrix} \begin{bmatrix} \ddot{q} \\ \lambda \end{bmatrix} = \begin{bmatrix} Q_A \\ \gamma \end{bmatrix} \quad (3.7)$$

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.

$$DOF = nb - nh = 6 - 5 = 1 \quad (3.8)$$

The constraint is defined as

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

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

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

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

$$\gamma = \Phi_q \cdot \ddot{q} = -(\Phi_q \cdot \dot{q})_q \cdot \dot{q} + \Phi_{qt} \cdot \dot{q} + \Phi_{tq} \cdot \dot{q} - \Phi_{tt} = -(\Phi_q \cdot \dot{q})_q \cdot \dot{q} - \Phi_{tt} = \begin{bmatrix} -\ddot{y}_2 (2\dot{y}_2 - L_2 \dot{\theta}_2 \cos(\theta_2)) - \ddot{x}_2 (2\dot{x}_2 + L_2 \dot{\theta}_2 \sin(\theta_2)) - \dot{\theta}_2 \left(\frac{L_2^2 \cos^2(\theta_2)}{2} + \frac{L_2^2 \sin^2(\theta_2)}{2} + L_2 \sin(\theta_2) \left(y_2 - \frac{L_2 \sin(\theta_2)}{2}\right) + L_2 \cos(\theta_2) \left(x_2 - \frac{L_2 \cos(\theta_2)}{2}\right) \right) + L_2 \dot{x}_2 \sin(\theta_2) - L_2 \dot{y}_2 \cos(\theta_2) \\ \frac{L_2 \cos(\theta_2) \dot{\theta}_2^2}{2} + \frac{L_3 \cos(\theta_3) \dot{\theta}_3^2}{2} \\ \frac{L_2 \sin(\theta_2) \dot{\theta}_2^2}{2} + \frac{L_3 \sin(\theta_3) \dot{\theta}_3^2}{2} \\ \frac{L_3 \dot{\theta}_3^2 \cos(\theta_3)}{2} \\ \frac{L_3 \dot{\theta}_3^2 \sin(\theta_3)}{2} \end{bmatrix}$$

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.

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

where $n = K_0 \cdot (q(6) - q(6)_{start}) + C_0 \cdot \dot{q}(6)$

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

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

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

$$F_{constraint(6x1)} = -\Phi_{q(6x5)}^T \cdot \lambda_{(5x1)} \quad (3.14)$$

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

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

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

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

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:

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

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.

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

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

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

5 Plots

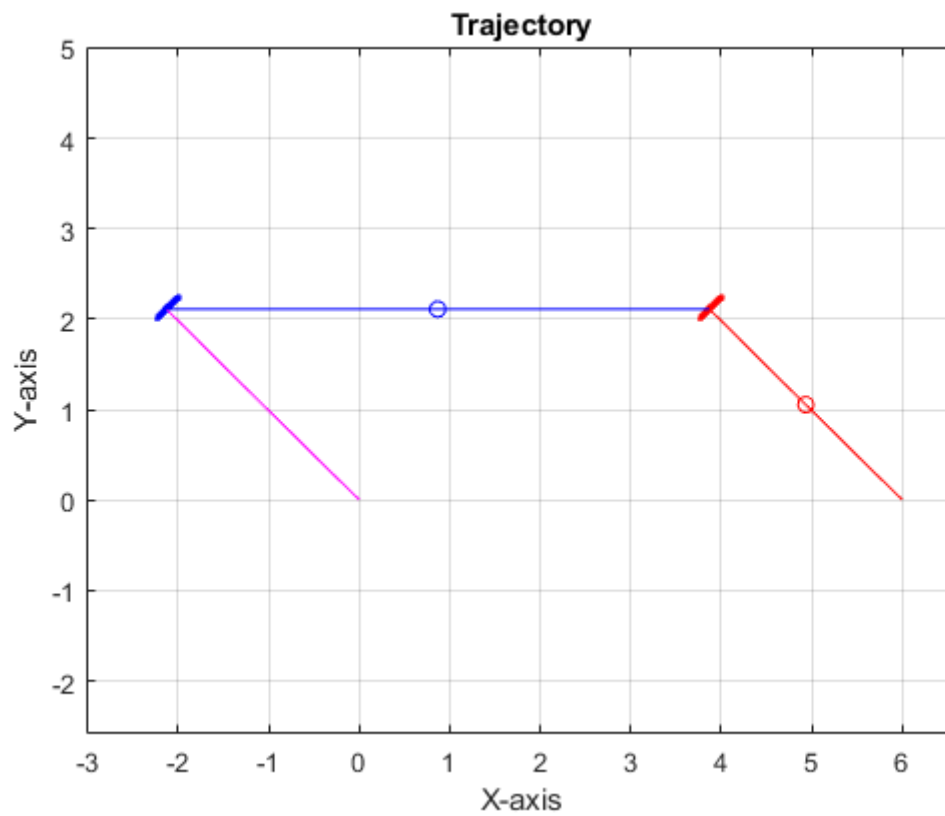


Figure 2: Trajectory

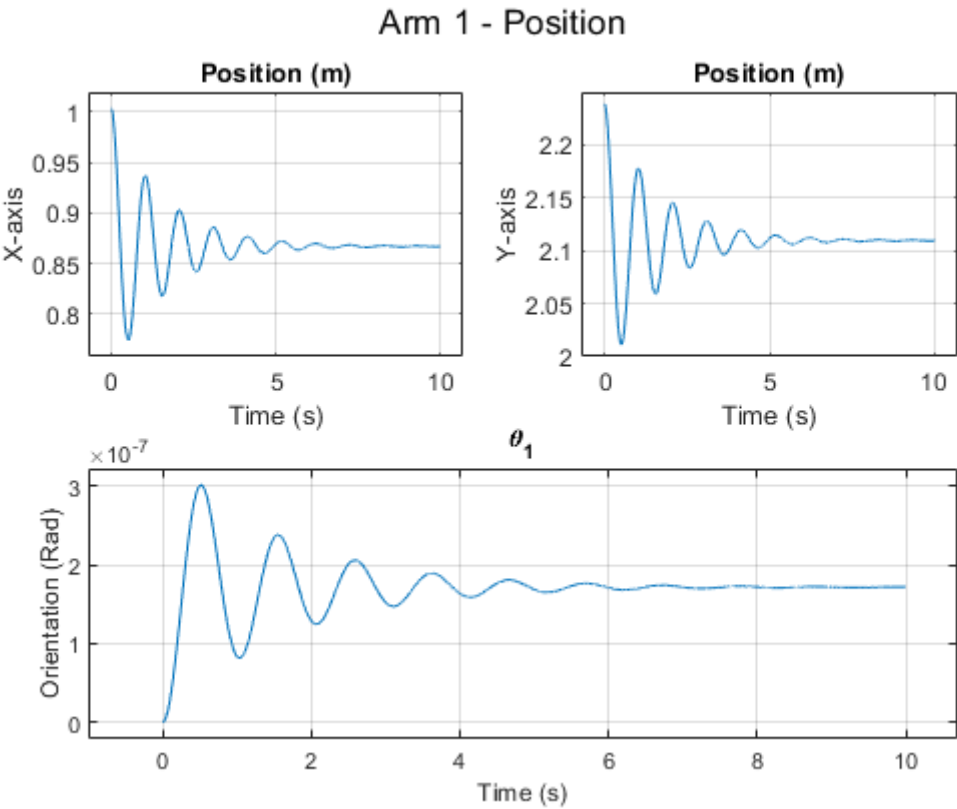


Figure 3: Position of arm 1

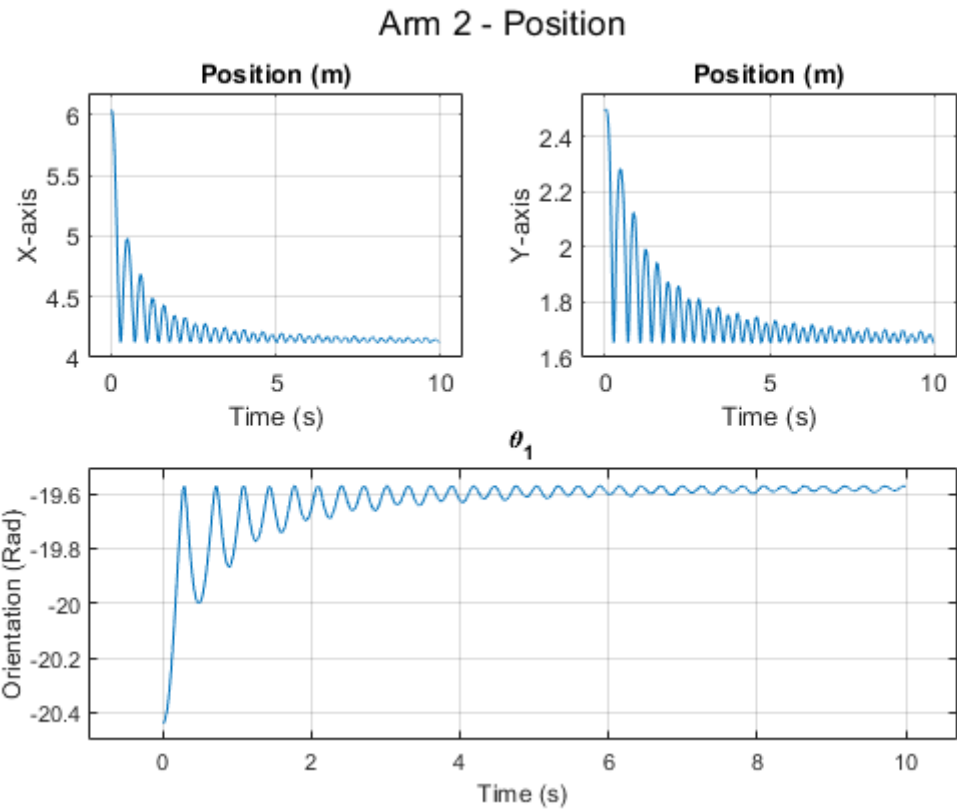


Figure 4: Position of arm 2

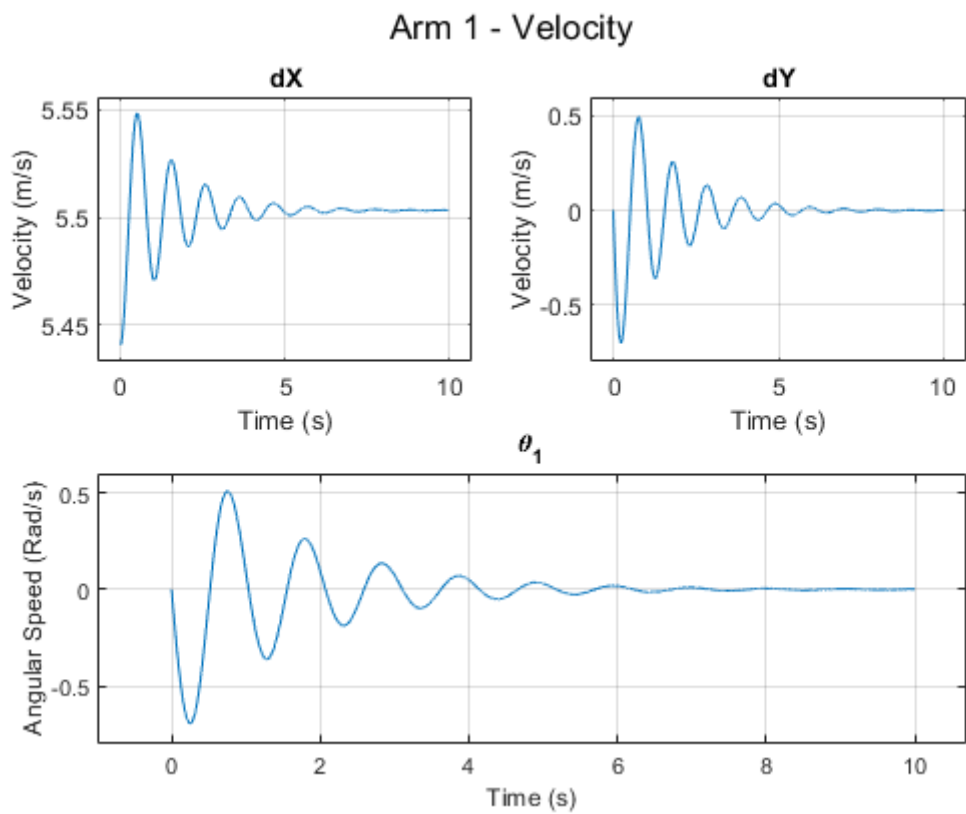


Figure 5: Velocity of arm 1

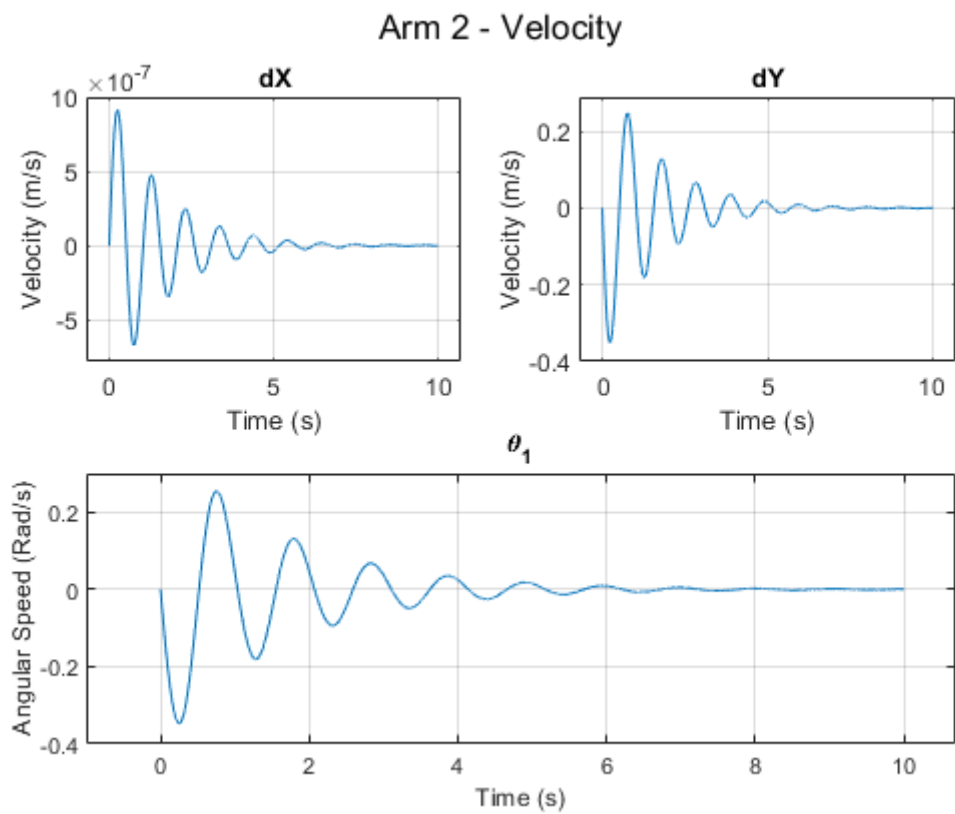


Figure 6: Velocity of arm 2

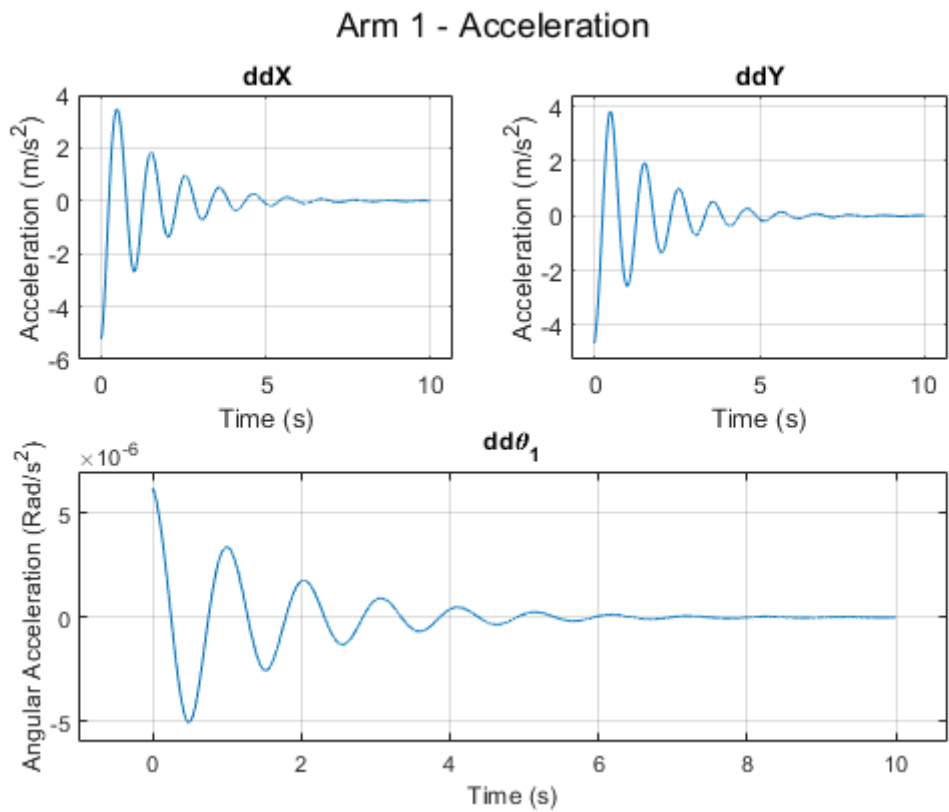


Figure 7: Acceleration of arm 1

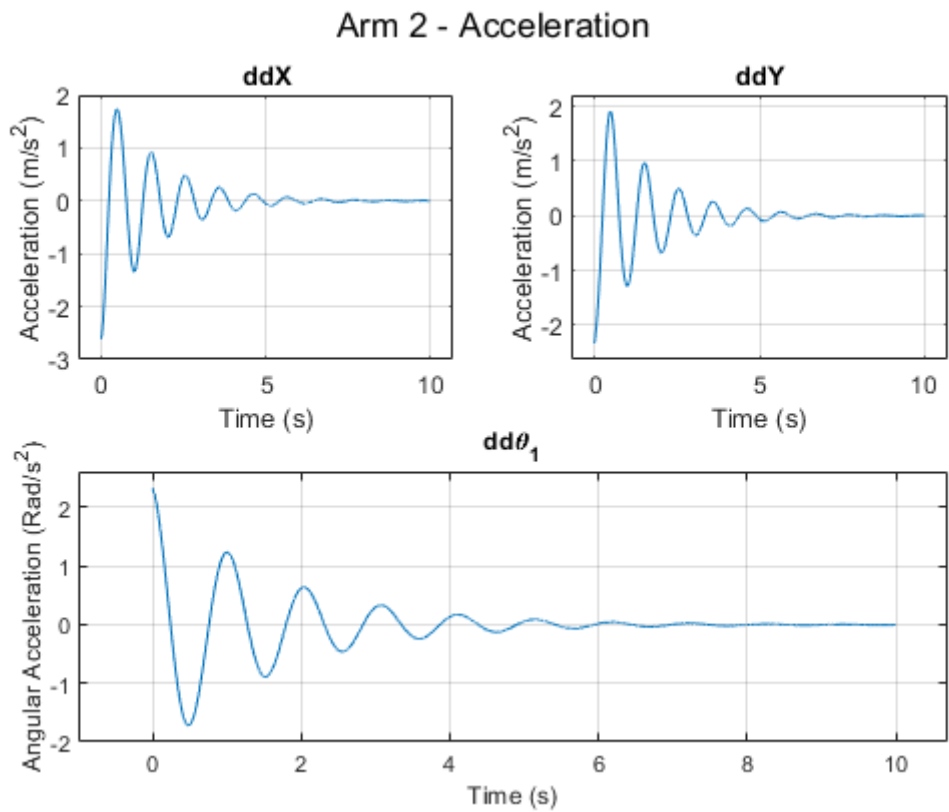


Figure 8: Acceleration of arm 2

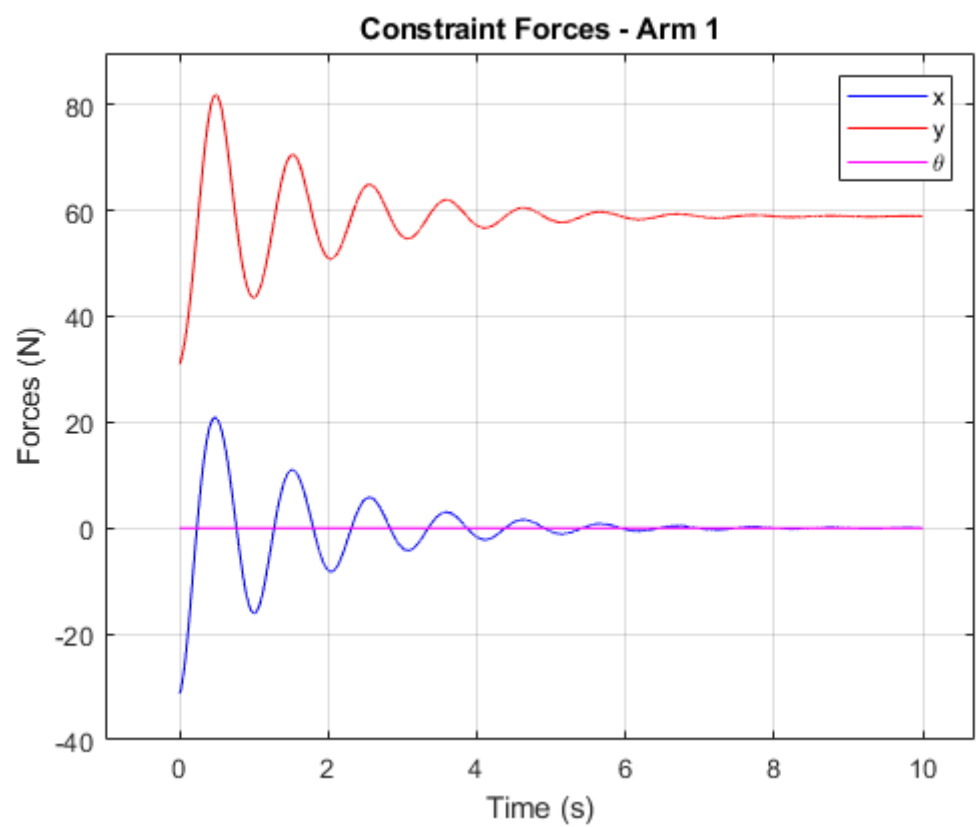


Figure 9: Constraint forces of arm 1

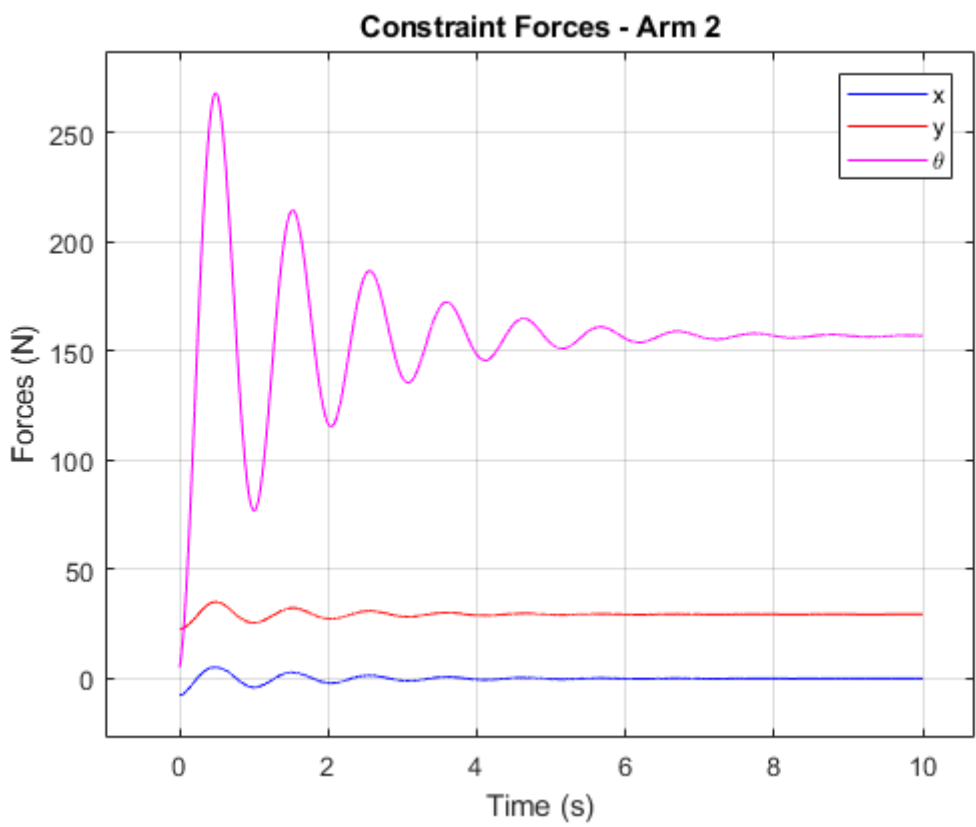


Figure 10: Constraint forces of arm 2

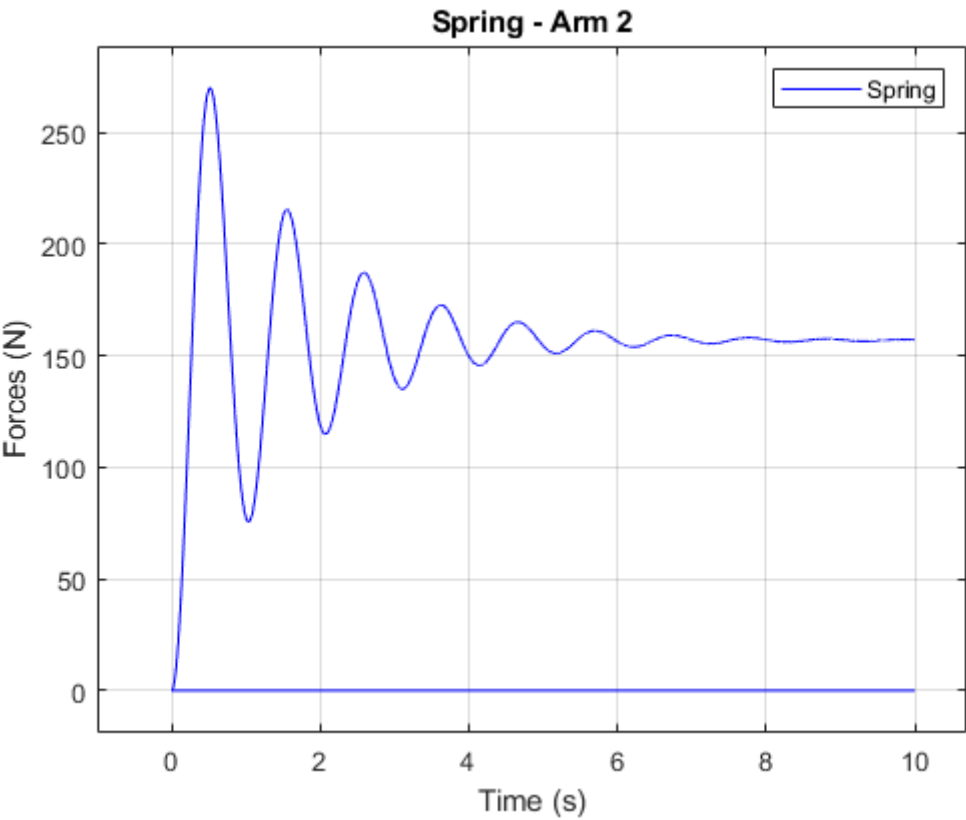


Figure 11: Spring Forces

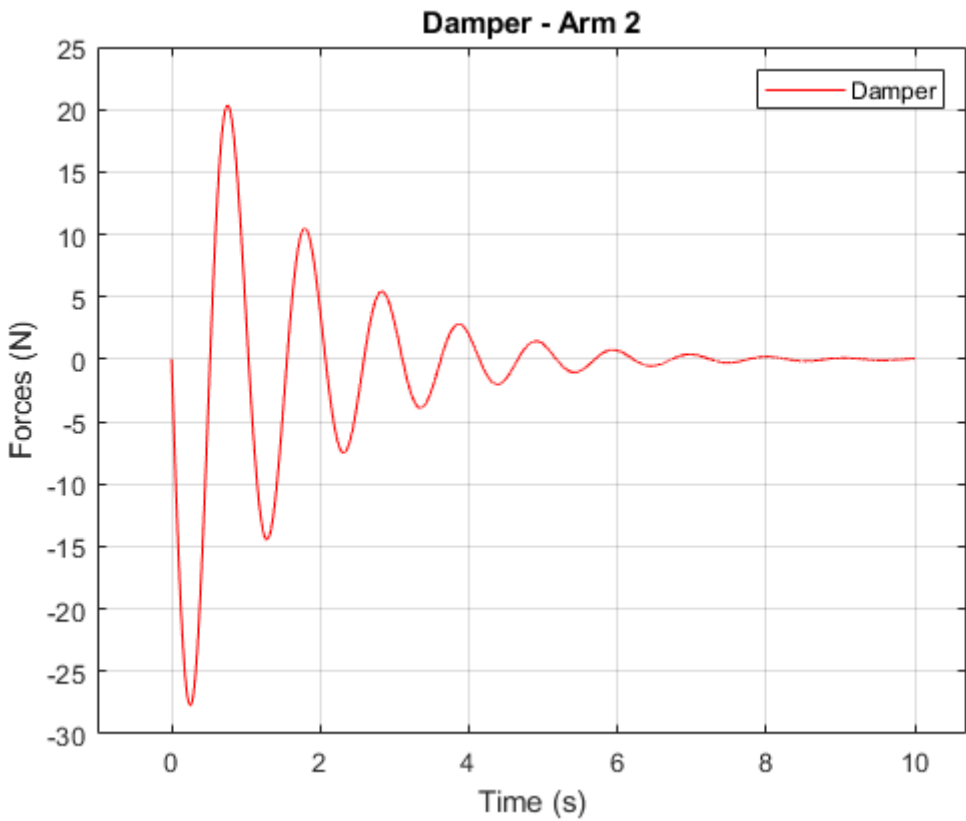


Figure 12: Damper Forces

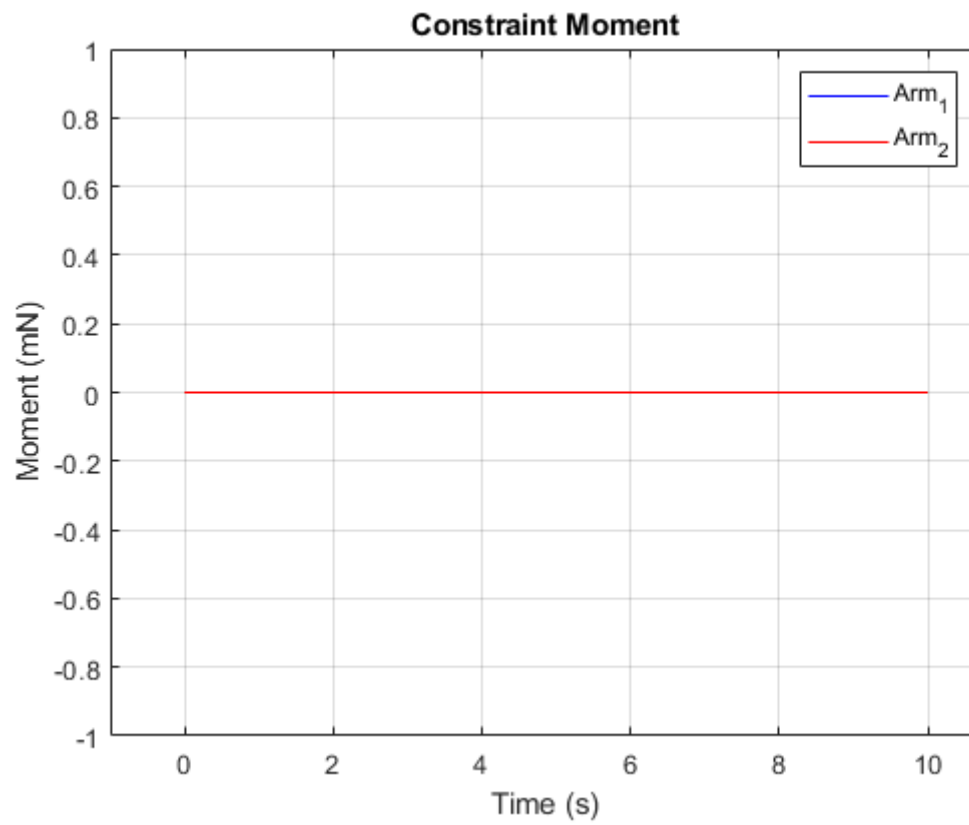


Figure 13: Moment

6 Appendix

6.1 Main Code

```

1 close all; clear all;
2 tspan = 0:0.001:10;
3 params = params;
4 L_2 = params.L2;
5 L_3 = params.L3;
6 y0 = zeros(1,17);
7 %q_guess = [-2; 0.5; pi/4; 1; 1; pi/10];
8
9 grashof = [params.L1,params.L2,params.L3,params.G];
10 if max(grashof)+min(grashof) < sum(grashof)-max(grashof)+min(grashof) % See if Grashof ...
    is satisfied
11     %q_guess = NewtonRaphston(tspan(1),q_guess)
12 else
13     error("Grashof is not satisfied")
14 end
15 q_guess = [1.0036; 2.2393; 0; 5.0018; 1.1197; 5.4405];
16 y0 = [q_guess;zeros(11,1)];
17 % Runge Kutta with correction
18 opts = odeset; opts=odeset(opts,'RelTol',1e-13,'AbsTol',1e-14);
19 [t y] = ode45(@(t,y) EOM(t,y,params),tspan,y0',opts);
20 % Position, velocity, acceleration
21 q = y(:,1:6);
22 q_dot = y(:,6:12);
23 B = @(theta) [-sin(theta) -sin(theta); cos(theta) -sin(theta)];
24 % Acceeleration, Constraint Forces, Constraint Torso
25 dy = zeros(length(tspan),17);
26 lambda = zeros(5,length(tspan));
27 for i = 1:length(tspan)
28     dy(i,:) = EOM(tspan(i),y(i,:)',params);
29     lambda(:,i) = dy(i,13:17);
30     CF(i,:) = -Phi.q(params,q(i,:)).'*lambda(:,i);
31     phik = Phi.q(params,q(i,:));
32     CT(i,1) = ([L_2/2;0].'*B(q(i,3)).'*phik(2:3,1:2).'-phik(2:3,3).')*lambda(2:3,i);
33     CT(i,2) = ([-L_3/2;0].'*B(q(i,6)).'*phik(2:3,4:5).'-phik(2:3,6).')*lambda(2:3,i);
34 end
35 q_ddot = dy(:,7:12);
36
37 % Constraint forces - Arm 1
38 figure(14)
39 plot(t,CT(:,1),'b',t,CT(:,2),'r')
40 hold on
41 axis padded
42 grid on
43 legend('Arm_1','Arm_2')
44 title('Constraint Moment')
45 ylabel('Moment (mN)')
46 xlabel('Time (s)')
47 hold off
48
49
50 % Spring-Damper Forces
51 K0 = params.K0;
52 C0 = params.C0;
53 Spring_Force = zeros(length(tspan));
54 Damper_Force_Force = zeros(length(tspan));
55 for i = 1:length(tspan)
56     Spring_Force(i) = K0*(q(i,6)-y0(6));
57     Damper_Force(i) = C0*q_dot(i,6);
58 end
59
60 % Getting points
61 A = @(theta) [cos(theta) -sin(theta); sin(theta) cos(theta)];
62 for i = 1:length(tspan)
63     P1(i,:) = [q(i,1);q(i,2)] + A(q(i,3))*[-L_2/2;0]; % Second arm left point
64     P2(i,:) = [q(i,1);q(i,2)] + A(q(i,3))*[L_2/2;0]; % Second arm right point
65     P3(i,:) = [q(i,4);q(i,5)] + A(q(i,6))*[-L_3/2;0]; % Third arm left point
66     P4(i,:) = [q(i,4);q(i,5)] + A(q(i,6))*[L_3/2;0]; % Third arm right point

```

```

67 end
68
69 % figure(1)% Live Trajectory
70 % hold on
71 % grid on
72 % title('Live Trajectory')
73 % xlabel('X-axis')
74 % ylabel('Y-axis')
75 % xlim([-4 8])
76 % ylim([-4 5])
77 % tic
78 % for p = 1:length(tspan)
79 %     Body1 = line([P1(p,1) P2(p,1)], [P1(p,2) P2(p,2)]);
80 %     Body1.Color = 'r';
81 %     Body2 = line([P3(p,1) P4(p,1)], [P3(p,2) P4(p,2)]);
82 %     Body2.Color = 'b';
83 %     Body3 = line([0 P1(p,1)], [0 P1(p,2)]);
84 %     Body3.Color = 'm';
85 %     pause(0.000001)
86 %     delete(Body1);
87 %     delete(Body2);
88 %     delete(Body3);
89 % end
90 % toc
91 % plot(q(p,1),q(p,2),'ro-');
92 % plot(q(p,4),q(p,5),'bo-');
93 % Body1 = line([P1(end,1) P2(end,1)], [P1(end,2) P2(end,2)]);
94 % Body1.Color = 'r';
95 % Body2 = line([P3(end,1) P4(end,1)], [P3(end,2) P4(end,2)]);
96 % Body2.Color = 'b';
97 % Body3 = line([0 P1(end,1)], [0 P1(end,2)]);
98 % Body3.Color = 'm';
99 % hold off
100 %
101 % Trajectory
102 % figure(2)
103 % plot(P1(:,1),P1(:,2),'.','Color','b');
104 % hold on
105 % plot(P2(:,1),P2(:,2),'b');
106 % plot(P3(:,1),P3(:,2),'.','Color','r');
107 % plot(P4(:,1),P4(:,2),'r');
108 % plot(q(end,1),q(end,2),'bo-');
109 % plot(q(end,4),q(end,5),'ro-');
110 % axis equal
111 % grid on
112 % axis padded
113 % title('Trajectory')
114 % xlabel('X-axis')
115 % ylabel('Y-axis')
116 % Body3 = line([0 P1(end,1)], [0 P1(end,2)]);
117 % Body3.Color = 'm';
118 % Body1 = line([P1(end,1) P2(end,1)], [P1(end,2) P2(end,2)]);
119 % Body1.Color = 'b';
120 % Body2 = line([P3(end,1) P4(end,1)], [P3(end,2) P4(end,2)]);
121 % Body2.Color = 'r';
122 % hold off
123
124 % Constraint forces - Arm 1
125 % figure(9)
126 % plot(t,CF(:,1),'b',t,CF(:,2),'r',t,CF(:,3),'m')
127 % hold on
128 % axis padded
129 % grid on
130 % legend('x','y','\theta')
131 % title('Constraint Forces - Arm 1')
132 % ylabel('Forces (N)')
133 % xlabel('Time (s)')
134 % hold off
135 %
136 % % Constraint forces - Arm 2
137 % figure(10)
138 % plot(t,CF(:,4),'b',t,CF(:,5),'r',t,CF(:,6),'m')
139 % hold on

```

```

140 % axis padded
141 % grid on
142 % legend('x','y','\theta')
143 % title('Constraint Forces - Arm 2')
144 % ylabel('Forces (N)')
145 % xlabel('Time (s)')
146 % hold off
147
148 % Damper Forces
149 figure(15)
150 plot(t,Damper.Force,'r')
151 hold on
152 axis padded
153 grid on
154 legend('Damper')
155 title('Damper - Arm 2')
156 ylabel('Forces (N)')
157 xlabel('Time (s)')
158 hold off
159
160 % Spring Force
161 % figure(11)
162 % plot(t, Spring_Force, 'b')
163 % hold on
164 % axis padded
165 % grid on
166 % legend('Spring')
167 % title('Spring - Arm 2')
168 % ylabel('Forces (N)')
169 % xlabel('Time (s)')
170 % hold off
171
172 % % Position - Arm 1
173 % figure(3)
174 % sgtitle('Arm 1 - Position')
175 % subplot(2,2,1,'Color','r')
176 % plot(t,q(:,1))
177 % grid on
178 % axis padded
179 % title('Position (m)')
180 % xlabel('Time (s)')
181 % ylabel('X-axis')
182 % subplot(2,2,2)
183 % plot(t,q(:,2))
184 % grid on
185 % axis padded
186 % title('Position (m)')
187 % xlabel('Time (s)')
188 % ylabel('Y-axis')
189 % subplot(2,2,[3 4],'Color','b')
190 % plot(t,q(:,3))
191 % grid on
192 % axis padded
193 % title('\theta_1')
194 % xlabel('Time (s)')
195 % ylabel('Orientation (Rad)')
196 %
197 % % Velocity - Arm 1
198 % figure(4)
199 % sgtitle('Arm 1 - Velocity')
200 % grid on
201 % axis padded
202 % subplot(2,2,1)
203 % plot(t,q-dot(:,1))
204 % grid on
205 % axis padded
206 % title('dX')
207 % xlabel('Time (s)')
208 % ylabel('Velocity (m/s)')
209 % subplot(2,2,2)
210 % plot(t,q-dot(:,2))
211 % grid on
212 % axis padded

```



```

213 % title('dY')
214 % xlabel('Time (s)')
215 % ylabel('Velocity (m/s)')
216 % subplot(2,2,[3 4])
217 % plot(t,q_dot(:,3))
218 % grid on
219 % axis padded
220 % title('\theta.1')
221 % xlabel('Time (s)')
222 % ylabel('Angular Speed (Rad/s)')
223 %
224 % % Acceleration - Arm 1
225 % figure(5)
226 % sgtitle('Arm 1 - Acceleration')
227 % grid on
228 % axis padded
229 % subplot(2,2,1)
230 % plot(t,q_ddot(:,1))
231 % grid on
232 % axis padded
233 % title('ddX')
234 % xlabel('Time (s)')
235 % ylabel('Acceleration (m/s^2)')
236 % subplot(2,2,2)
237 % plot(t,q_ddot(:,2))
238 % grid on
239 % axis padded
240 % title('ddY')
241 % xlabel('Time (s)')
242 % ylabel('Acceleration (m/s^2)')
243 % subplot(2,2,[3 4])
244 % plot(t,q_ddot(:,3))
245 % grid on
246 % axis padded
247 % title('dd\theta.1')
248 % xlabel('Time (s)')
249 % ylabel('Angular Acceleration (Rad/s^2)')
250 %
251 % % Position - Arm 2
252 % figure(6)
253 % sgtitle('Arm 2 - Position')
254 % subplot(2,2,1,'Color','r')
255 % plot(t,q(:,4))
256 % grid on
257 % axis padded
258 % title('Position (m)')
259 % xlabel('Time (s)')
260 % ylabel('X-axis')
261 % subplot(2,2,2)
262 % plot(t,q(:,5))
263 % grid on
264 % axis padded
265 % title('Position (m)')
266 % xlabel('Time (s)')
267 % ylabel('Y-axis')
268 % subplot(2,2,[3 4],'Color','b')
269 % plot(t,q(:,6))
270 % grid on
271 % axis padded
272 % title('\theta.1')
273 % xlabel('Time (s)')
274 % ylabel('Orientation (Rad)')
275 %
276 % % Velocity - Arm 2
277 % figure(7)
278 % sgtitle('Arm 2 - Velocity')
279 % grid on
280 % axis padded
281 % subplot(2,2,1)
282 % plot(t,q_dot(:,4))
283 % grid on
284 % axis padded
285 % title('dX')

```

```

286 % xlabel('Time (s)')
287 % ylabel('Velocity (m/s)')
288 % subplot(2,2,2)
289 % plot(t,q_dot(:,5))
290 % grid on
291 % axis padded
292 % title('dY')
293 % xlabel('Time (s)')
294 % ylabel('Velocity (m/s)')
295 % subplot(2,2,[3 4])
296 % plot(t,q_dot(:,6))
297 % grid on
298 % axis padded
299 % title('\theta_1')
300 % xlabel('Time (s)')
301 % ylabel('Angular Speed (Rad/s)')
302 %
303 % % Acceleration - Arm 2
304 % figure(8)
305 % sgtitle('Arm 2 - Acceleration')
306 % grid on
307 % axis padded
308 % subplot(2,2,1)
309 % plot(t,q_ddot(:,4))
310 % grid on
311 % axis padded
312 % title('ddX')
313 % xlabel('Time (s)')
314 % ylabel('Acceleration (m/s^2)')
315 % subplot(2,2,2)
316 % plot(t,q_ddot(:,5))
317 % grid on
318 % axis padded
319 % title('ddY')
320 % xlabel('Time (s)')
321 % ylabel('Acceleration (m/s^2)')
322 % subplot(2,2,[3 4])
323 % plot(t,q_ddot(:,6))
324 % grid on
325 % axis padded
326 % title('dd\theta_1')
327 % xlabel('Time (s)')
328 % ylabel('Angular Acceleration (Rad/s^2)')

```

6.2 Equation of Motion

```

1 function dy = EOM(t,y,params)
2 m2 = params.m2;
3 m3 = params.m3;
4 J2 = params.J2;
5 J3 = params.J3;
6 g = params.g;
7 K0 = params.K0;
8 C0 = params.C0;
9 persistent y0;
10 q = y(1:6);
11 q_dot = y(7:12);
12 lambda = y(13:17);
13
14
15 if t == 0
16     y0 = q(6);
17 end
18 n_1 = K0*(q(6)-y0) + C0*q_dot(6);
19
20 M = diag([m2 m2 J2 m3 m3 J3]);
21 Q_A = [0 -m2*g 0 0 -m3*g -n_1].';
22 A = [M Phi_q(params,q).'; Phi_q(params,q) zeros(5)];
23 B = [Q_A; Gamma(params,q,q_dot)];
24 C = A\B;

```

% Mass Matrix
 % Applied Force
 % Combined Matrix
 % Combined Matrix
 % Acc, Lambda

```

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

```

6.3 Matlab - Symbolic

```

1 sympref('AbbreviateOutput',false);
2 syms x_2 y_2 theta_2 x_3 y_3 theta_3 L_1 L_2 L_3 G omega t;
3 syms x_dot_2 y_dot_2 theta_dot_2 x_dot_3 y_dot_3 theta_dot_3;
4 syms A(theta)
5
6 A(theta) = [cos(theta) -sin(theta); sin(theta) cos(theta)];
7 r_2 = [x_2;y_2];
8 r_3 = [x_3;y_3];
9
10 sip_2 = [L_2/2;0];
11 sip_3 = [L_3/2;0];
12
13 % Defining Arm 1
14 C = [0;0];
15 C_3 = L_1;
16 rp2 = r_2+A(theta_2)*-sip_2;
17
18 phi_k3 = r_3 + A(theta_3)*sip_3-[G;0] % Absolute ...
19 % constraint - L_3
20 phi_k2 = r_2 + A(theta_2)*sip_2-r_3-A(theta_3)*-sip_3; % Revolute ...
21 % constraint - L_3-L_2
22 phi_k1 = (rp2-C).*(rp2-C)-C_3^2 % Absolute ...
23 % constraint - L_2
24
25 % Constraints
26 phi_K = [phi_k1;phi_k2;phi_k3]; % Kinematic ...
27 % constraint
28 q = [x_2 y_2 theta_2 x_3 y_3 theta_3].' % Generalized ...
29 % cartesian coordinates
30 phi = [phi_K] % Constraint matrix
31 phiq = jacobian(phi,q) % Jacobian Matrix
32 phit = -diff(phi,t) % Nu Matrix
33 q_dot = [x_dot_2 y_dot_2 theta_dot_2 x_dot_3 y_dot_3 theta_dot_3].'; % Generealized ...
34 % cartesian velocity coordinates
35 gamma = -jacobian(phiq*q_dot,q)*q_dot-diff(phit,t) % Gamma matrix
36
37 %making functions of the symbolic
38 matlabFunction(phi,'file','Phi');
39 matlabFunction(phiq,'file','Phi_q');
40 matlabFunction(phit,'file','Nu');
41 matlabFunction(gamma,'file','Gamma');

```

6.4 Parameters

```

1 function params = params
2 params.m2 = 6;
3 params.m3 = 3;
4 params.L1 = 3;
5 params.L2 = 6;
6 params.L3 = 3;
7 params.J2 = params.m2/12*params.L2^2;
8 params.J3 = params.m3/12*params.L3^2;
9 params.g = 9.81;
10 params.G = 6;
11 params.K0 = 2500;
12 params.C0 = 80;
13
14 % params = params;
15 % L_2 = params.L2;
16 % L_3 = params.L3;
17 % x_2 = q(1);

```

```

18 % y_2 = q(2);
19 % theta_2 = q(3);
20 % x_3 = q(4);
21 % y_3 = q(5);
22 % theta_3 = q(6);
23 % x_dot_2 = q_dot(1);
24 % y_dot_2 = q_dot(2);
25 % theta_dot_2 = q_dot(3);
26 % x_dot_3 = q_dot(4);
27 % y_dot_3 = q_dot(5);
28 % theta_dot_3 = q_dot(6);

```

6.5 Φ_q

```

1 function phiq = Phi_q(params,q)
2 %PHI_Q
3 %     PHI_Q = PHI_Q(L_2,L_3,THETA_2,THETA_3,X_2,Y_2)
4
5 %     This function was generated by the Symbolic Math Toolbox version 8.7.
6 %     08-May-2022 16:02:29
7
8 L_2 = params.L2;
9 L_3 = params.L3;
10 theta_2 = q(3);
11 theta_3 = q(6);
12 x_2 = q(1);
13 y_2 = q(2);
14
15
16 t2 = cos(theta_2);
17 t3 = cos(theta_3);
18 t4 = sin(theta_2);
19 t5 = sin(theta_3);
20 t6 = (L_2.*t2)./2.0;
21 t7 = (L_3.*t3)./2.0;
22 t8 = (L_2.*t4)./2.0;
23 t9 = (L_3.*t5)./2.0;
24 t10 = -t8;
25 t11 = -t9;
26 phiq = ...
    reshape([x_2.*2.0-L_2.*t2,1.0,0.0,0.0,0.0,y_2.*2.0-L_2.*t4,0.0,1.0,0.0,0.0,-L_2.*t4.*(t6-x_2)+L_2.*t2.

```

6.6 γ

```

1 function gamma = Gamma(params,q,q_dot)
2 %GAMMA
3 %     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)
4
5 %     This function was generated by the Symbolic Math Toolbox version 8.7.
6 %     08-May-2022 16:02:29
7
8 L_2 = params.L2;
9 L_3 = params.L3;
10 theta_2 = q(3);
11 theta_3 = q(6);
12 theta_dot_2 = q_dot(3);
13 theta_dot_3 = q_dot(6);
14 x_2 = q(1);
15 x_dot_2 = q_dot(1);
16 y_2 = q(2);
17 y_dot_2 = q_dot(2);
18
19 t2 = cos(theta_2);
20 t3 = cos(theta_3);
21 t4 = sin(theta_2);
22 t5 = sin(theta_3);

```

```

23 t6 = L_2.^2;
24 t7 = theta_dot_2.^2;
25 t8 = theta_dot_3.^2;
26 t9 = (L_3.*t3.*t8)./2.0;
27 t10 = (L_3.*t5.*t8)./2.0;
28 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

```

1 function phi = Phi(G,L_1,L_2,L_3,theta_2,theta_3,x_2,x_3,y_2,y_3)
2 %PHI
3 %    PHI = PHI(G,L_1,L_2,L_3,THETA_2,THETA_3,X_2,X_3,Y_2,Y_3)
4
5 %    This function was generated by the Symbolic Math Toolbox version 8.7.
6 %    08-May-2022 16:02:28
7
8 t2 = cos(theta_2);
9 t3 = cos(theta_3);
10 t4 = sin(theta_2);
11 t5 = sin(theta_3);
12 t6 = (L_2.*t2)./2.0;
13 t7 = (L_3.*t3)./2.0;
14 t8 = (L_2.*t4)./2.0;
15 t9 = (L_3.*t5)./2.0;
16 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 Raphston

```

1 function w = NewtonRaphston(t,q)
2 es = 0.000000000001; % Tolerance
3 maxit = 100000; % Maximum iteration
4 iter = 0; % Defining iteration
5 driving = zeros(1,6);
6 driving(3) = 1;
7 while(1)
8     Phil = [Phi(params,t,q);q(3)*1*t];
9     Jac = [Phi.q(q,params); driving];
10    dq = Jac\Phil; % Defining the change
11    q = q-dq; % Removing change from previous guess
12    iter = iter+1; % Counting Iteration
13    ea = 100*max(abs(dq./q)); % Looking at error-value
14    if(iter>maxit || ea<es) % See if conditions are satisfied
15        w = q; % Saving data if satisfied
16        break
17    end
18 end

```