M4 - Computational Dynamics

Jakob Alsaker - 201808760

April 6, 2022

Contents

1 Introduction **2**

2 Exercises **4**

3 Defining constraints **5**
3.1 Deriving Jacobian 5
3.2 Deriving ν 6
3.3 Deriving γ 6

4 Simulation **6**

5 Appendix **10**
5.1 Symbolics 10
5.2 Main 10
5.3 Φ 12
5.4 Φ_q 13
5.5 ν 13
5.6 γ 13

1 Introduction

Perform a full kinematic analysis of the inverted slider-crank mechanism shown in figure 1 and determine the maximum acceleration $|\ddot{r_E}|$ experienced at point E.

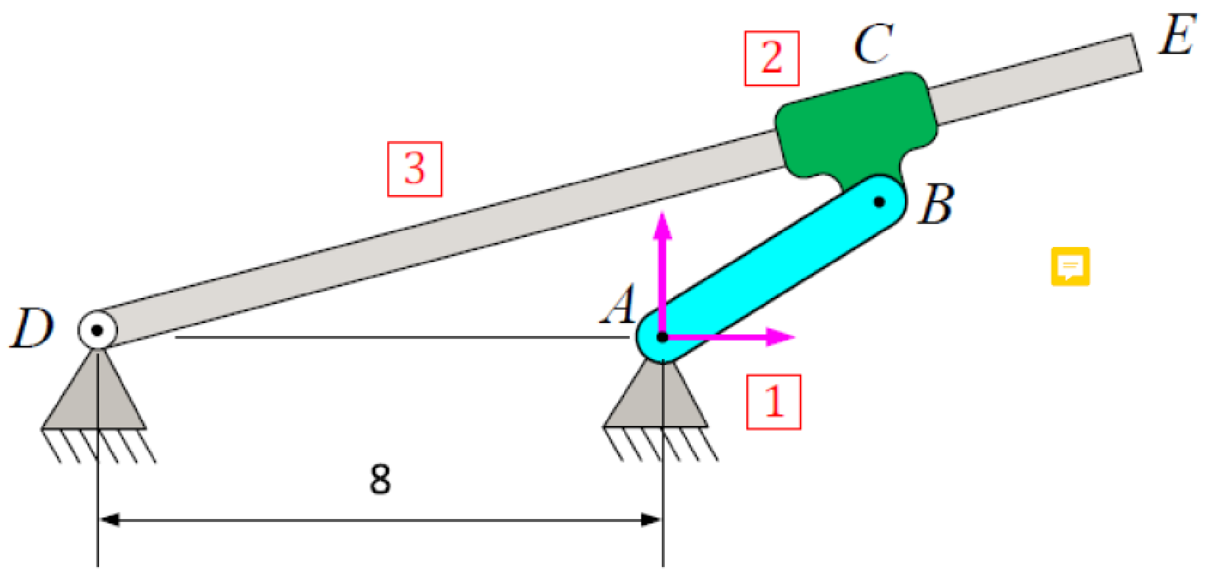


Figure 1: Slider-Crank mechanism

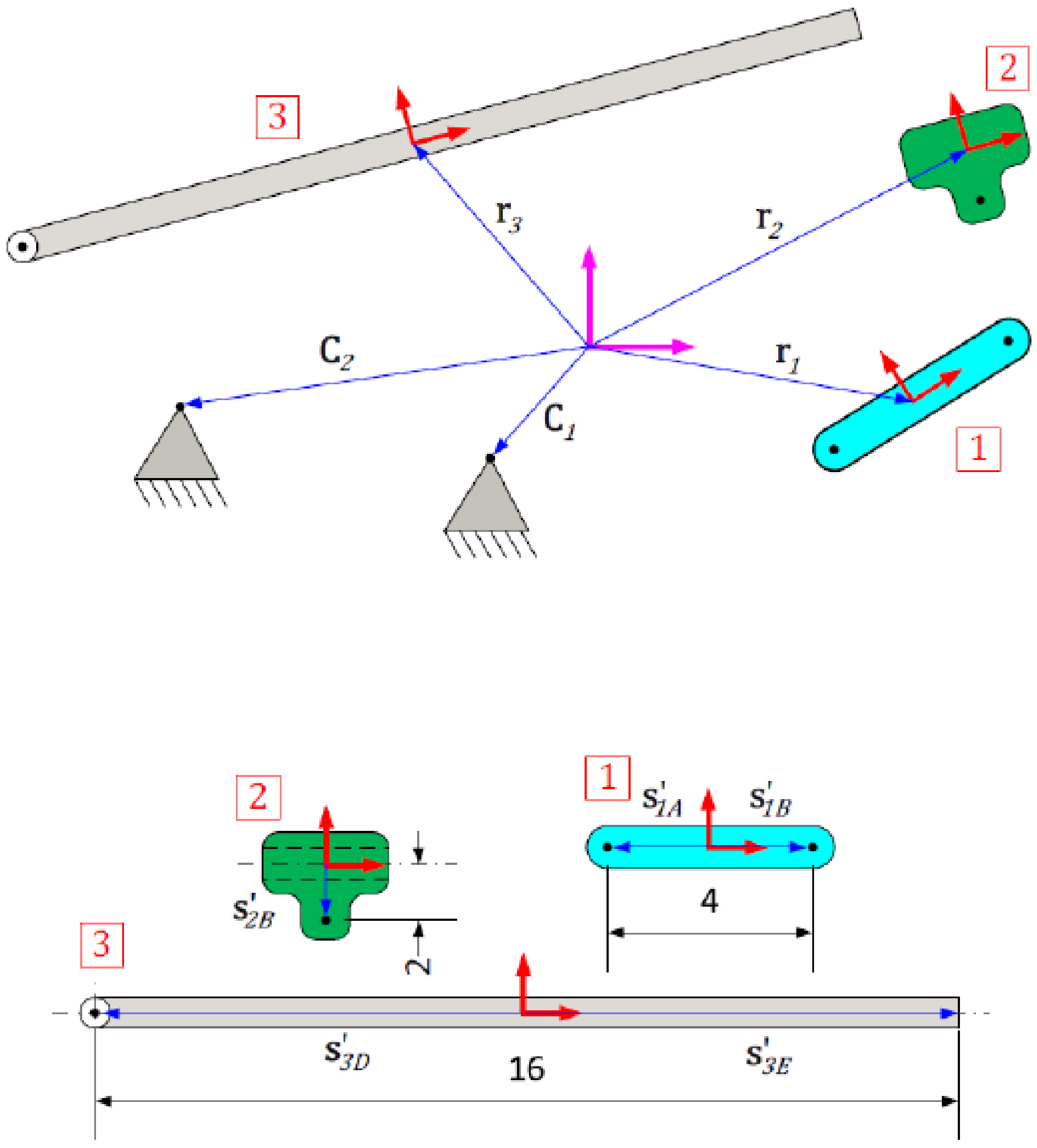


Figure 2: Expanded view of Slider-Crank mechanism

2 Exercises

a Define joints (Constraints needed (use figure 2)

b Setup the kinematic constraint equation Φ^K

c Setup a driver constraint equations Φ^D so body 1 rotates at $\phi = 1.5\frac{rad}{s}$

d Do position analysis for $t = 0...10 seconds$

e Plot graphs of results and animate

f Determine the constraint Jacobian Φ_q

g Setup the velocity- and acceleration equations

h Calculate the velocity $\dot{q}$ and acceleration $\ddot{q}$ over time

i Plot all results to verify their consistency $(q,\dot{q},\ddot{q})$

j Determine the maximum value of the acceleration in the point E, $|\ddot{r_E}|$

3 Defining constraints

Looking at figure 1 it can be seen that 3 bodies are described in the Slider-Crank mechanism. The First arm is described by being locked into a position while rotation and controlling the slider in the end. Body 2 is a slider which is locked with the first body while rotation and thereafter translating on the third body. The third body can be seen being locked into a position while rotation based on the slider. Having 3 bodies is to have 9 generalized Cartesian coordinates but as the second body can be describes as a relative revolution-translational composite joint, 2 bodies can describe the mechanism. Following constraints are used to define the mechanism using figure 2 to reduce the number of DOF.

1. Absolute x-y constraint on body 3 using C_2.
2. Absolute x-y constraint on body a using C_1.
3. Composite revolution-translational composite joint between body 1 and 3

The number of DOF can therefore now be calculated

$$DOF = 3 \cdot NB - NH = 3 \cdot 2 - 2 - 2 - 1 = 1 \tag{3.1}$$

Having 1 DOF can now result in the mechanism can be solve by having one absolute driven constraint.

The kinematic constraint can be made by using the constraint described above and using the dimensions from figure 2.

$$\Phi^K = \begin{bmatrix} x_1 - 2\cos(\theta_1) \\ y_1 - 2\sin(\theta_1) \\ \cos(\theta_2)\,(y_1 - y_2 + 2\sin(\theta_1) - 8\sin(\theta_2)) - \sin(\theta_2)\,(x_1 - x_2 + 2\cos(\theta_1) - 8\cos(\theta_2)) + 2 \\ x_2 - 8\cos(\theta_2) + 8 \\ y_2 - 8\sin(\theta_2) \end{bmatrix} \tag{3.2}$$

By concatenating the kinematic constraint with the driven constraint, the constraint matrix can be defined as

$$\Phi = \begin{bmatrix} x_1 - 2\cos(\theta_1) \\ y_1 - 2\sin(\theta_1) \\ \cos(\theta_2)\,(y_1 - y_2 + 2\sin(\theta_1) - 8\sin(\theta_2)) - \sin(\theta_2)\,(x_1 - x_2 + 2\cos(\theta_1) - 8\cos(\theta_2)) + 2 \\ x_2 - 8\cos(\theta_2) + 8 \\ y_2 - 8\sin(\theta_2) \\ \theta_1 - \frac{3t}{2} \end{bmatrix} \tag{3.3}$$

3.1 Deriving Jacobian

The Jacobian constraint can be found by deriving the constraint matrix by the generalised coordinates which is defined as

$$q = \begin{bmatrix} x_1 \\ y_1 \\ \theta_1 \\ x_2 \\ y_2 \\ \theta_2 \\ x_3 \\ y_3 \\ \theta_3 \end{bmatrix} \tag{3.4}$$

$$\Phi_q = \frac{\partial \Phi_i(\mathbf{q})}{\partial q_j} =$$
$$\begin{bmatrix} 1 & 0 & 2\sin(\theta_1) & 0 & 0 & 0 \\ 0 & 1 & -2\cos(\theta_1) & 0 & 0 & 0 \\ -\sin(\theta_2) & \cos(\theta_2) & 2\cos(\theta_1)\cos(\theta_2) + 2\sin(\theta_1)\sin(\theta_2) & \sin(\theta_2) & -\cos(\theta_2) & -8\cos(\theta_2)^2 - 8\sin(\theta_2)^2 - \cos(\theta_2)\,(x_1 - x_2 + 2\cos(\theta_1) - 8\cos(\theta_2)) - \sin(\theta_2)\,(y_1 - y_2 + 2\sin(\theta_1) - 8\sin(\theta_2)) \\ 0 & 0 & 0 & 1 & 0 & 8\sin(\theta_2) \\ 0 & 0 & 0 & 0 & 1 & -8\cos(\theta_2) \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{3.5}$$

3.2 Deriving ν

Nu can be found using the chain rule on Φ to derive the velocity.

$$\nu = \Phi_q \cdot \dot{q} = -\Phi_t = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{3}{2} \end{bmatrix} \tag{3.6}$$

Using equation (3.3) and taking the partial derivative, Nu can be found

3.3 Deriving γ

Gamma can be found by using the chain rule for nu.

$$\gamma = \Phi_q \cdot \ddot{q} = -(\Phi_q \cdot \dot{q})_q \cdot \dot{q} + \Phi_{qt} \cdot \dot{q} + \Phi_{tq} \cdot \dot{q} - \Phi_{tt} = -(\Phi_q \cdot \dot{q})_q \cdot \dot{q} - \Phi_{tt} = \tag{3.7}$$

$$\begin{bmatrix} -2\,\dot{\theta}_1^2\,\cos(\theta_1) \\ -2\,\dot{\theta}_1^2\,\sin(\theta_1) \\ \dot{\theta}_2\,\dot{x}_1\,\cos(\theta_2) - \dot{\theta}_2\left(\dot{\theta}_2\,(\sin(\theta_2)\,(x_1 - x_2 + 2\,\cos(\theta_1) - 8\,\cos(\theta_2)) - \cos(\theta_2)\,(y_1 - y_2 + 2\,\sin(\theta_1) - 8\,\sin(\theta_2))) - \dot{\theta}_1\,(2\,\cos(\theta_1)\,\sin(\theta_2) - 2\,\cos(\theta_2)\,\sin(\theta_1)) - \dot{x}_1\,\cos(\theta_2) + \dot{x}_2\,\cos(\theta_2) - \dot{y}_1\,\sin(\theta_2) + \dot{y}_2\,\sin(\theta_2)\right) - \dot{\theta}_1\left(\dot{\theta}_1\,(2\,\cos(\theta_1)\,\sin(\theta_2) - 2\,\cos(\theta_2)\,\sin(\theta_1)) - \dot{\theta}_2\,(2\,\cos(\theta_1)\,\sin(\theta_2) - 2\,\cos(\theta_2)\,\sin(\theta_1))\right) - \dot{\theta}_2\,\dot{x}_2\,\cos(\theta_2) + \dot{\theta}_2\,\dot{y}_1\,\sin(\theta_2) - \dot{\theta}_2\,\dot{y}_2\,\sin(\theta_2) \\ -8\,\dot{\theta}_2^2\,\cos(\theta_2) \\ -8\,\dot{\theta}_2^2\,\sin(\theta_2) \\ 0 \end{bmatrix}$$

The double partial derivative with respect to q and t is zero, it is removed from the equation.

4 Simulation

To simulate the position and orientation of the pendulum, values of q have to be found. As the system is nonlinear, Newton-Raphson method can be applied to approximate the values within a tolerance.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{4.1}$$

To determine the maximum value of the acceleration at point $|\ddot{r_E}|$, the resultant acceleration have to found. By applying Pythagoras for the absolute value of the x- and y-acceleration at each point and thereafter finding the highest value, will give the resultant acceleration.

$$|\ddot{r}_E|_{Result} = max(\sqrt{x_i^2 + y_i^2}) = 17.60[m/s^2] \tag{4.2}$$

To determine the maximum acceleration, the angular acceleration times the radius can be used.

$$|\ddot{r}_E|_\theta = \dot{\omega} \cdot R = 19.00[Rad/s^2] \tag{4.3}$$

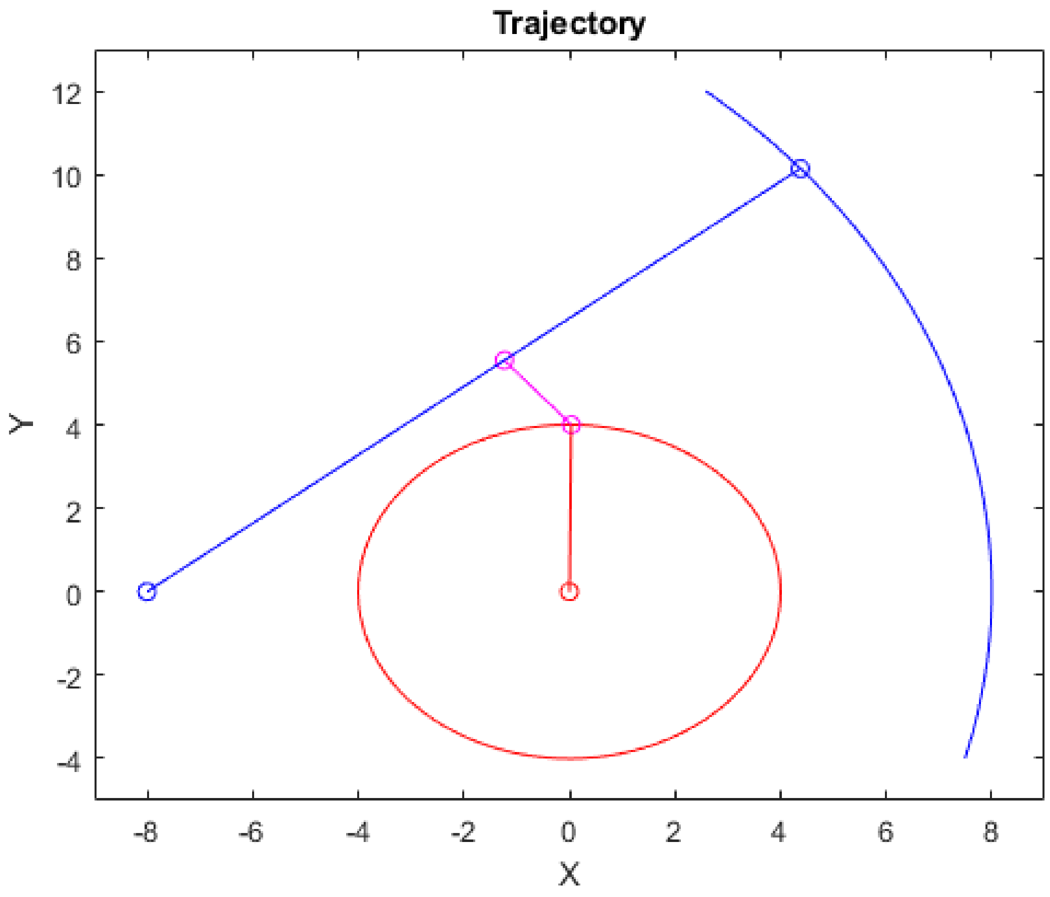
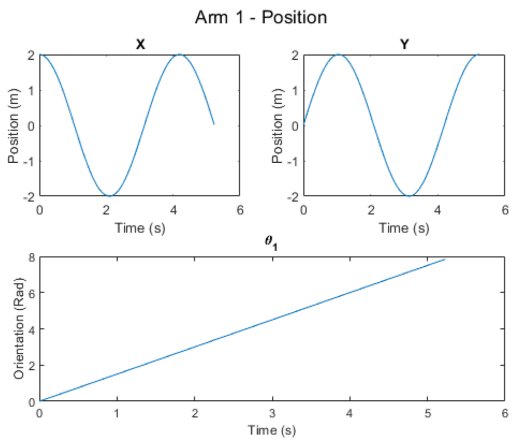
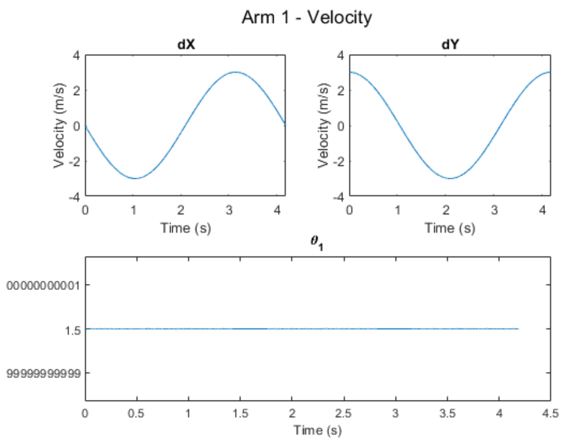


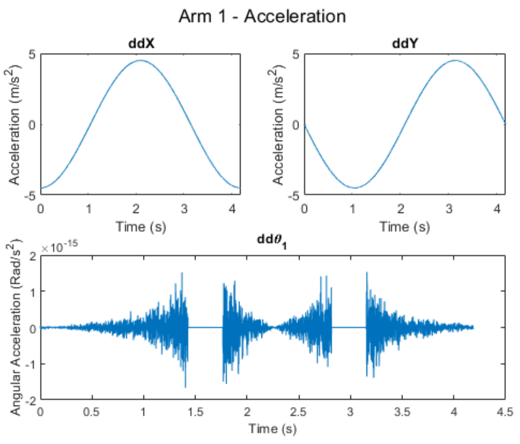
Figure 3: Trajectory



(a) Position

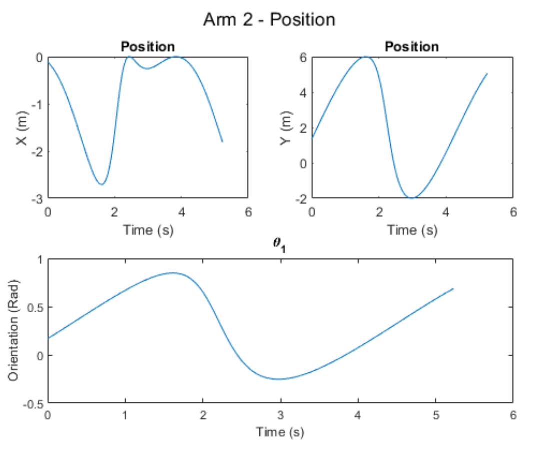


(b) Velocity

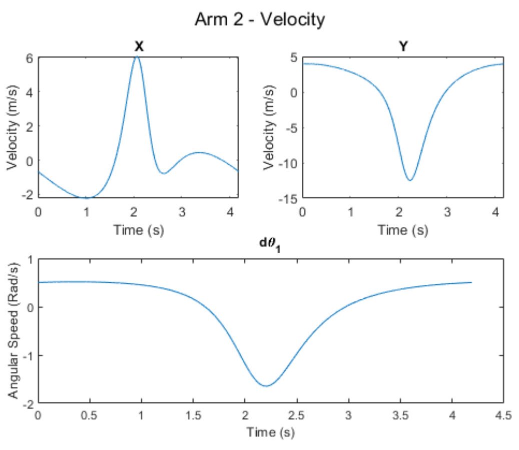


(c) Acceleration

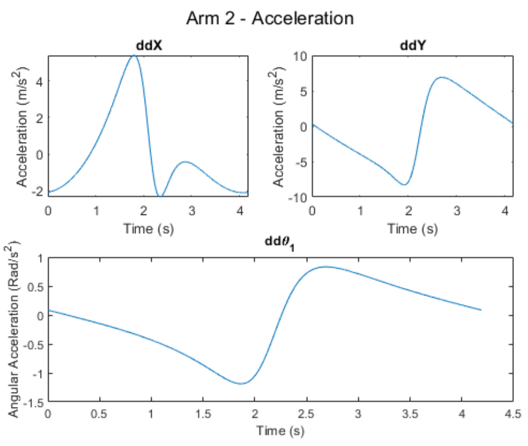
Figure 4: Kinematics of Arm 1



(a) Position



(b) Velocity



(c) Acceleration

Figure 5: Kinematics of Arm 2

5 Appendix

5.1 Symbolics

```
sympref('AbbreviateOutput',false);
syms x_1 y_1 theta_1 x_2 y_2 theta_2 L_1 L_2 omega t;
syms x_dot_1 y_dot_1 theta_dot_1 x_dot_2 y_dot_2 theta_dot_2;



R = [0 -1; 1 0]

r_1 = [x_1;y_1];
A1 = [cos(theta_1) -sin(theta_1); sin(theta_1) cos(theta_1)];
r_2 = [x_2;y_2];
A2 = [cos(theta_2) -sin(theta_2); sin(theta_2) cos(theta_2)];


C = -2
Q_i = [-8;0];
P_i = [8;0];
si_p = [8;0];
vi_p = transpose([P_i(1)-Q_i(1) P_i(2)-Q_i(2)]);
vi = A2*vi_p
vvi = norm(Q_i - P_i);
si_j = [2;0];


phi_k2 = (1/vvi)*vi_p.'*R.'*A2.'*(r_1+A1*si_j-r_2-A2*si_p)-C
phi_k3 = r_2 + A2*[-8; 0]-[-8;0]         % Rod x-y constraint
phi_k1 = r_1 + A1*[-2; 0]                % Rod x-y constraint
phi_K = [phi_k1; phi_k2; phi_k3]
phi_D = [theta_1 - 1.5*t]
q = [x_1;y_1;theta_1;x_2;y_2;theta_2]
phi = [phi_K;phi_D]
phiq = jacobian(phi,q) % Jacobian
phit = -diff(phi,t)
q_dot = [x_dot_1 y_dot_1 theta_dot_1 x_dot_2 y_dot_2 theta_dot_2].';
gamma = -jacobian(phiq*q_dot,q)*q_dot-diff(phit,t)



% matlabFunction(phi,'file','Phi');
% matlabFunction(phiq,'file','Phi_q');
% matlabFunction(phit,'file','Nu');
% matlabFunction(gamma,'file','Gamma');
```

5.2 Main

```
close all; clear all
% Defining start values
n_rev = 1;                 % How many revolutions
rev = (2*pi)/1.5;          % One Revolution
t = 0:0.001:rev*n_rev;      % Time-span
N = length(t);             % Length of time

q = [1; 1; pi/6; 0.5; 0.5; pi/3;];  % Guess Values

% Finding Position, velocity and acceleration
[q,q_dot,q_ddot] = NewtonRaphston(t,q);

pp = 15; % Percentage into the simulation
p = round(N*(pp/100)); % Point in simulation

% Geometry of arms
Arm1_x = [0 4*cos(q(3,p))];
Arm1_y = [0 4*sin(q(3,p))];
```

```
Arm2_x = [-8 8*cos(q(6,p))];
Arm2_y = [0 16*sin(q(6,p))];

% Plotting Trajectory
figure(1)
plot(Arm1_x,Arm1_y,'o-');
hold on
plot(Arm2_x,Arm2_y,'o-');
plot(q(1,:),q(2,:),q(4,:),q(5,:))
xlim([-10 10])
ylim([-6 12])
title('Trajectory')
xlabel('X')
ylabel('Y')
hold off

figure(2)
sgtitle('Arm 1 - Position')
subplot(2,2,1)
plot(t,q(1,:))
title('X')
xlabel('Time (s)')
ylabel('Position (m)')
subplot(2,2,2)
plot(t,q(2,:))
title('Y')
xlabel('Time (s)')
ylabel('Position (m)')
subplot(2,2,[3 4])
plot(t,q(3,:))
title('\theta_1')
xlabel('Time (s)')
ylabel('Orientation (Rad)')

figure(3)
sgtitle('Arm 2 - Position')
subplot(2,2,1)
plot(t,q(4,:))
title('X')
xlabel('Time (s)')
ylabel('Position (m)')
subplot(2,2,2)
plot(t,q(5,:))
title('Y')
xlabel('Time (s)')
ylabel('Position (m)')
subplot(2,2,[3 4])
plot(t,q(6,:))
title('\theta_1')
xlabel('Time (s)')
ylabel('Orientation (Rad)')

figure(4)
sgtitle('Arm 1 - Velocity')
subplot(2,2,1)
plot(t,q_dot(1,:))
title('dX')
xlabel('Time (s)')
ylabel('Velocity (m/s)')
subplot(2,2,2)
plot(t,q_dot(2,:))
title('dY')
xlabel('Time (s)')
ylabel('Velocity (m/s)')
subplot(2,2,[3 4])
plot(t,q_dot(3,:))
title('\theta_1')
xlabel('Time (s)')
ylabel('Angular Speed (Rad/s)')

figure(5)
sgtitle('Arm 2 - Velocity')
subplot(2,2,1)
```

```
plot(t,q_dot(4,:))
title('X')
xlabel('Time (s)')
ylabel('Velocity (m/s)')
subplot(2,2,2)
plot(t,q_dot(5,:))
title('Y')
xlabel('Time (s)')
ylabel('Velocity (m/s)')
subplot(2,2,[3 4])
plot(t,q_dot(6,:))
title('d\theta_1')
xlabel('Time (s)')
ylabel('Angular Speed (Rad/s)')

figure(6)
sgtitle('Arm 1 - Acceleration')
subplot(2,2,1)
plot(t,q_ddot(1,:))
title('ddX')
xlabel('Time (s)')
ylabel('Acceleration (m/s^2)')
subplot(2,2,2)
plot(t,q_ddot(2,:))
title('ddY')
xlabel('Time (s)')
ylabel('Acceleration (m/s^2)')
subplot(2,2,[3 4])
plot(t,q_ddot(3,:))
title('dd\theta_1')
xlabel('Time (s)')
ylabel('Angular Acceleration (Rad/s^2)')

figure(7)
sgtitle('Arm 2 - Acceleration')
subplot(2,2,1)
plot(t,q_ddot(4,:))
title('ddX')
xlabel('Time (s)')
ylabel('Acceleration (m/s^2)')
subplot(2,2,2)
plot(t,q_ddot(5,:))
title('ddY')
xlabel('Time (s)')
ylabel('Acceleration (m/s^2)')
subplot(2,2,[3 4])
plot(t,q_ddot(6,:))
title('dd\theta_1')
xlabel('Time (s)')
ylabel('Angular Acceleration (Rad/s^2)')
```

5.3 Φ

```
function phi = Phi(t,q)
%Phi
%    PHI = Phi(T,THETA_1,THETA_2,X_1,X_2,Y_1,Y_2)

%    This function was generated by the Symbolic Math Toolbox version 9.0.
%    27-Mar-2022 13:05:23
x_1 = q(1);
y_1 = q(2);
theta_1 = q(3);
x_2 = q(4);
y_2 = q(5);
theta_2 = q(6);


t2 = cos(theta_1);
t3 = cos(theta_2);
t4 = sin(theta_1);
```

```
t5 = sin(theta_2);
t6 = t2.*2.0;
t7 = t4.*2.0;
t8 = t3.*8.0;
t9 = t5.*8.0;
t10 = -t8;
t11 = -t9;
phi = ...
    [-t6+x_1;-t7+y_1;-t5.*(t6+t10+x_1-x_2)+t3.*(t7+t11+y_1-y_2)+2.0;t10+x_2+8.0;t11+y_2;t.*(-3.0./2.0)+thet
```

5.4 Φ_q

```
function phiq = Phi_q(q)
%Phi_q
%    PHIQ = Phi_q(THETA_1,THETA_2,X_1,X_2,Y_1,Y_2)

%    This function was generated by the Symbolic Math Toolbox version 9.0.
%    27-Mar-2022 13:05:24
x_1 = q(1);
y_1 = q(2);
theta_1 = q(3);
x_2 = q(4);
y_2 = q(5);
theta_2 = q(6);



t2 = cos(theta_1);
t3 = cos(theta_2);
t4 = sin(theta_1);
t5 = sin(theta_2);
t6 = t2.*2.0;
t7 = t4.*2.0;
t8 = t3.*8.0;
t9 = t5.*8.0;
t10 = -t8;
phiq = ...
    reshape([1.0,0.0,-t5,0.0,0.0,0.0,0.0,1.0,t3,0.0,0.0,0.0,t7,-t6,t3.*t6+t5.*t7,0.0,0.0,1.0,0.0,0.0,t5,1.0
```

5.5 ν

```
function phit = Nu
%Nu
%    PHIT = Nu

%    This function was generated by the Symbolic Math Toolbox version 9.0.
%    27-Mar-2022 13:05:24

phit = [0.0;0.0;0.0;0.0;0.0;3.0./2.0];
```

5.6 γ

```
function gamma = Gamma(q,q_dot,params)
%Gamma
%    GAMMA = ...
    Gamma(THETA_1,THETA_2,THETA_DOT_1,THETA_DOT_2,X_1,X_2,X_DOT_1,X_DOT_2,Y_1,Y_2,Y_DOT_1,Y_DOT_2)

%    This function was generated by the Symbolic Math Toolbox version 9.0.
%    27-Mar-2022 13:05:24

x_1 = q(1);
y_1 = q(2);
theta_1 = q(3);
```

```
x_2 = q(4);
y_2 = q(5);
theta_2 = q(6);

x_dot_1 = q_dot(1);
y_dot_1 = q_dot(2);
theta_dot_1 = q_dot(3);
x_dot_2 = q_dot(4);
y_dot_2 = q_dot(5);
theta_dot_2 = q_dot(6);


t2 = cos(theta_1);
t3 = cos(theta_2);
t4 = sin(theta_1);
t5 = sin(theta_2);
t6 = theta_dot_1.^2;
t7 = theta_dot_2.^2;
t8 = t2.*t5.*2.0;
t9 = t3.*t4.*2.0;
t10 = -t9;
t11 = t8+t10;
t12 = t11.*theta_dot_1;
gamma = ...
    [t2.*t6.*-2.0;t4.*t6.*-2.0;theta_dot_2.*(t12+t3.*x_dot_1-t3.*x_dot_2+t5.*y_dot_1-t5.*y_dot_2-theta_dot_
```