Human Neuromechanical Control and Learning
Tutorial 2: Redundancy

Completed by: Jenna Kelly Luchak
CID: 01429938

February 8, 2018

Question 1

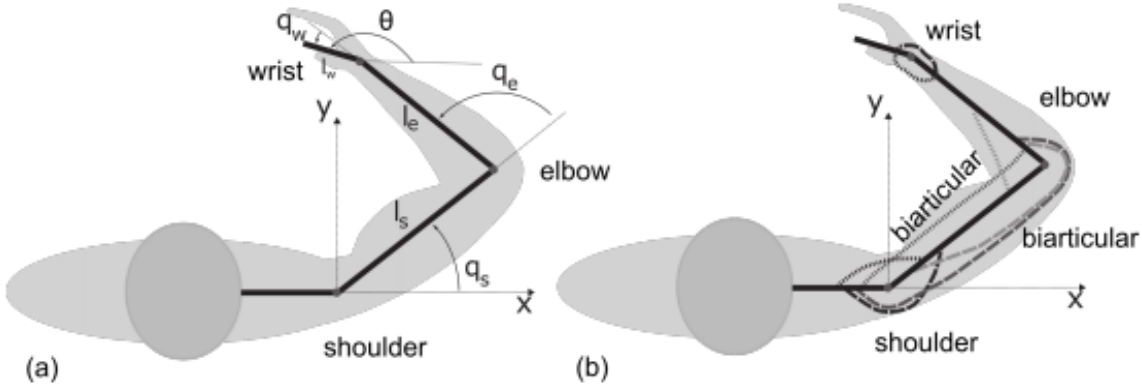

Figure 1: A 3 joint system of planar arm movements. A) The angle convention where q_w q_e q_s are the wrist, elbow and shoulder angles respectively, and $l_w l_e$ l_s are the distances of the hand, forearm and the upper arm respectively. B) 8 representative muscles acting on the limb, where the wrist, elbow, biarticular and shoulder each possess a flexor and extensor muscle.

a) Deriving the direct kinematics of the 3-joint arm system.
Let the end-effector of the arm be the tip of the hand which has position (x,y).
Let the origin of the arm be located at the shoulder joint at position (0,0).
The position of the end-effector can be derived as follows through kinematics.

Equation1

$$x = \boldsymbol{x_{upperarm} + x_{forearm} + x_{hand}}$$
$$x = \boldsymbol{l_s cos(q_s) + l_e cos(q_s + q_e) + l_w cos(q_s + q_e + q_w)}$$

Equation 2

$$y = \boldsymbol{y_{upperarm} + y_{forearm} + y_{hand}}$$
$$y = \boldsymbol{l_s sin(q_s) + l_e sin(q_s + q_e) + l_w sin(q_s + q_e + q_w)}$$

In order to derive the inverse kinematics of the 3-joint arm system, assume that the wrist orientation with respect to the body is constant.
Therefore,

$$\boldsymbol{\theta = constant = q_s + q_e + q_w}$$
$$\boldsymbol{\frac{d\theta}{dt} = 0 \rightarrow \frac{dq_s}{dt} + \frac{dq_e}{dt} + \frac{dq_w}{dt} = 0}$$

Based on this fact, the previous 3-joint system can be simplified to a 2 joint system, composed of the upper arm and forearm. The terms x_{hand} and y_{hand} were removed from Equations 1 and 2. Therefore, the wrist is now recognized as the end-effector and two new angles, α , β are introduced. A schematic follows in Figure 2.

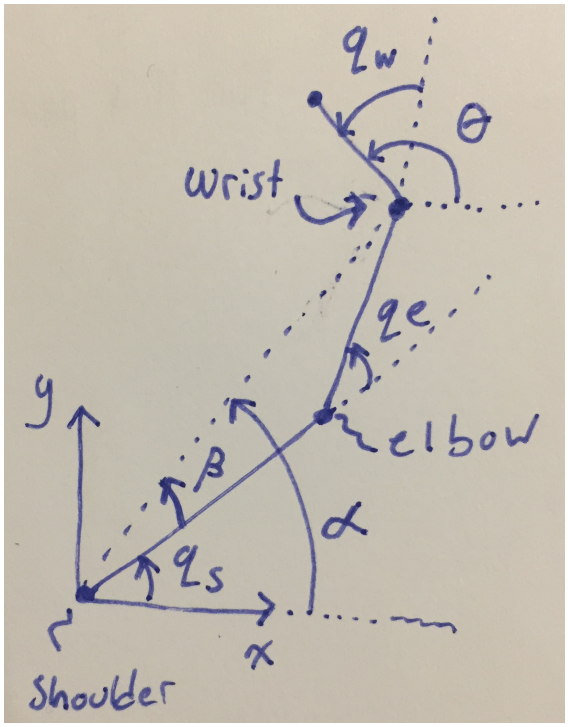

Figure 2: A 2 joint system of a planar arm with a relative fixed wrist orientation.

$$x = x_{upperarm} + x_{forearm}$$
$$y = y_{upperarm} + y_{forearm}$$

$$x^2 + y^2 = l_s^2 + l_e^2 - 2l_s l_e \cos(\pi - q_e)$$
$$q_e = \cos^{-1}\left(\frac{x^2 + y^2 - l_s^2 - l_e^2}{2l_s l_e}\right)$$
$$\alpha = arctan2\left(\frac{y}{x}\right)$$
$$l_e^2 = x^2 + y^2 + l_s^2 - 2l_s\sqrt{x^2 + y^2} * cos\beta$$

$$\beta = \cos^{-1}\left(\frac{x^2 + y^2 + l_s^2 - l_e^2}{2l_s\sqrt{x^2} + y^2}\right)$$
$$q_s = \alpha - \beta$$
$$q_w = \theta - q_s - q_e = \theta - (\alpha - \beta) - \cos^{-1}\left(\frac{x^2 + y^2 - l_s^2 - l_e^2}{2l_s l_e}\right)$$

b) In order to derive the differential kinematics of the 3-joint arm system, I took the derivatives of the direct kinematics equations found in Equations 1 and 2, as follows.

Equation 3

$$\frac{dx}{dt} = \dot{x} = -l_s sin(q_s)\dot{q_s} - l_e sin(q_s + q_e)(\dot{q_s} + \dot{q_e}) - l_w sin(q_s + q_e + q_w)(\dot{q_s} + \dot{q_e} + \dot{q_w})$$

Equation 4

$$\frac{dy}{dt} = \dot{y} = l_s cos(q_s)\dot{q_s} + l_e cos(q_s + q_e)(\dot{q_s} + \dot{q_e}) + l_w cos(q_s + q_e + q_w)(\dot{q_s} + \dot{q_e} + \dot{q_w})$$

Equations 3 and 4 can be re-written into a simpler formula, as follows.

$$\dot{x} = \dot{q_s}\left(-l_s sin(q_s) - l_e sin(q_s + q_e) - l_w sin(q_s + q_e + q_w)\right) + \dot{q_e}\left(-l_e sin(q_s + q_e) - l_w sin(q_s + q_e + q_w)\right) + \dot{q_w}\left(-l_w sin(q_s + q_e + q_w)\right)$$

$$\dot{y} = \dot{q_s}\left(l_s cos(q_s) + l_e cos(q_s + q_e) + l_w cos(q_s + q_e + q_w)\right) + \dot{q_e}\left(l_e cos(q_s + q_e) + l_w cos(q_s + q_e + q_w)\right) + \dot{q_w}\left(l_w cos(q_s + q_e + q_w)\right)$$

Where, the Jacobian transforms angular velocities in joint space to velocities in task space.

Equation 5

$$\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix} = [Jacobian] * \begin{bmatrix}\dot{q_s}\\ \dot{q_e}\\ \dot{q_w}\end{bmatrix}$$

$$[Jacobian] =$$

$$\begin{bmatrix} -l_s sin(q_s) - l_e sin(q_s + q_e) - l_w sin(q_s + q_e + q_w) & \left(-l_e sin(q_s + q_e) - l_w sin(q_s + q_e + q_w)\right) & -l_w sin(q_s + q_e + q_w) \\ l_s cos(q_s) + l_e cos(q_s + q_e) + l_w cos(q_s + q_e + q_w) & \left(l_e cos(q_s + q_e) + l_w cos(q_s + q_e + q_w)\right) & l_w cos(q_s + q_e + q_w) \end{bmatrix}$$

There also exists a Jacobian in muscle space that transforms task joint velocity to muscle space velocity. The equation is as follows.

$$\dot{\lambda} = J_\mu \dot{q}$$

The Jacobian is only dependent on the moment arms of the 8 representative muscles in the limb. The first column represents the relationship of all of the muscles with respect to the shoulder joint, the second column with respect to the elbow joint and the third column the wrist joint. The equation is as follows.

Equation 6

$$J_\mu(\rho) = \left(\frac{\partial \lambda_i}{\partial q_j}\right)$$

$$J_\mu = \begin{bmatrix} \rho_{Shoulder+} & 0 & 0 \\ -\rho_{shoulder-} & 0 & 0 \\ \rho_{\text{biarticular shoulder}+} & \rho_{\text{biarticular elbow}+} & 0 \\ -\rho_{\text{biarticular shoulder}-} & -\rho_{\text{biarticular elbow}-} & 0 \\ 0 & \rho_{elbow+} & 0 \\ 0 & -\rho_{elbow-} & 0 \\ 0 & 0 & \rho_{wrist+} \\ 0 & 0 & -\rho_{wrist-} \end{bmatrix}$$

The moment arms in Equation 6 follows the following convention. Moment arm ρ_{m+} and ρ_{m-} represents the agonist and antagonist muscles of the arbitrary muscle pair "m".

In general, the stiffness matrix in joint space is characterized by the following equation.

$$K = \frac{dJ_\mu^T}{dq}\mu + J_\mu^T K_\mu J_\mu$$

However, I assumed that there are no external forces or muscle tensions acting on the limb so $\mu = 0$ and the equation for the stiffness matrix will be defined as follows.

Equation 7

$$K = J_\mu^T K_\mu J_\mu$$

In order to compute the stiffness matrix in joint space, I assumed that the muscle stiffness matrix is completely diagonal in muscle space and all moment arms from Equation 6 are equal to ρ.

Therefore, the Jacobian in muscle space becomes, as follows.

$$J_\mu = \begin{bmatrix} \rho & 0 & 0 \\ -\rho & 0 & 0 \\ \rho & \rho & 0 \\ -\rho & -\rho & 0 \\ 0 & \rho & 0 \\ 0 & -\rho & 0 \\ 0 & 0 & \rho \\ 0 & 0 & -\rho \end{bmatrix}$$

The stiffness matrix is diagonal and each element represents the stiffness of a muscle in the arm. For example, K1 is the agonist of the shoulder muscle, K2 the antagonist of the shoulder muscle, K3 the agonist of the biarticular, K4 the antagonist of the biarticular, K5 the agonist of the elbow, K6 the antagonist of the elbow, K7 the agonist of the wrist and K8 the antagonist of the wrist. The equation of the stiffness matrix in muscle space is as follows.

Equation 8

$$K_\mu = \begin{bmatrix} K_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & K_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & K_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & K_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & K_5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & K_6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & K_7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & K_8 \end{bmatrix}$$

Therefore, by subbing into Equation 8, stiffness was calculated as follows.

$$K = \begin{bmatrix} \rho & -\rho & \rho & -\rho & 0 & 0 & 0 & 0 \\ 0 & 0 & \rho & -\rho & \rho & -\rho & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \rho & -\rho \end{bmatrix} * \begin{bmatrix} K_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & K_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & K_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & K_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & K_5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & K_6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & K_7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & K_8 \end{bmatrix} * \begin{bmatrix} \rho & 0 & 0 \\ -\rho & 0 & 0 \\ \rho & \rho & 0 \\ -\rho & -\rho & 0 \\ 0 & \rho & 0 \\ 0 & -\rho & 0 \\ 0 & 0 & \rho \\ 0 & 0 & -\rho \end{bmatrix}$$

$$K = \begin{bmatrix} \rho K_1 & -\rho K_2 & \rho K_3 & -\rho K_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & \rho K_3 & -\rho K_4 & \rho K_5 & -\rho K_6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \rho K_7 & -\rho K_8 \end{bmatrix} * \begin{bmatrix} \rho & 0 & 0 \\ -\rho & 0 & 0 \\ \rho & \rho & 0 \\ -\rho & -\rho & 0 \\ 0 & \rho & 0 \\ 0 & -\rho & 0 \\ 0 & 0 & \rho \\ 0 & 0 & -\rho \end{bmatrix}$$

$$K = \begin{bmatrix} \rho^2 K_1 + \rho^2 K_2 + \rho^2 K_3 + \rho^2 K_4 & \rho^2 K_3 + \rho^2 K_4 & 0 \\ \rho^2 K_3 + \rho^2 K_4 & \rho^2 K_3 + \rho^2 K_4 + \rho^2 K_5 + \rho^2 K_6 & 0 \\ 0 & 0 & \rho^2 K_7 + \rho^2 K_8 \end{bmatrix}$$

c) Figures 3-7 below depict the plotted results of my Matlab Script.

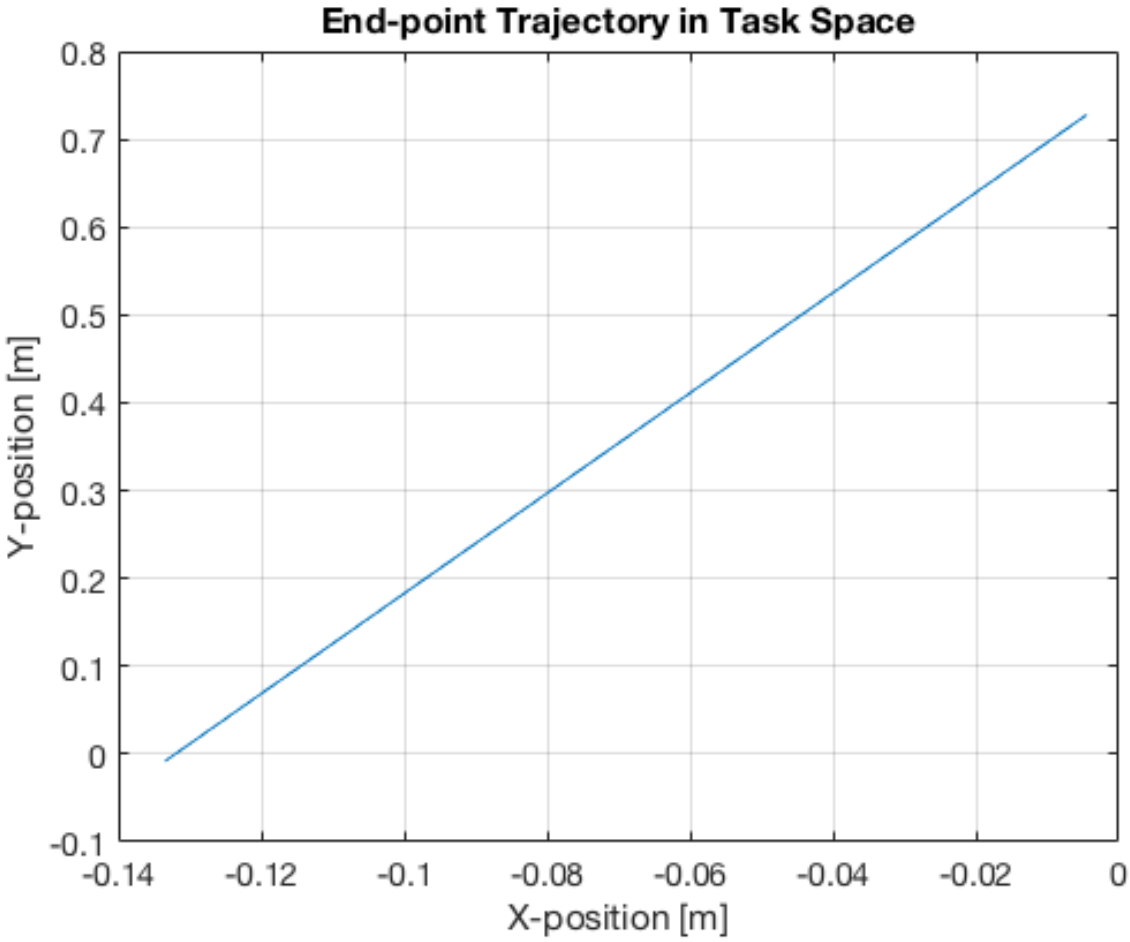

Figure 3: Endpoint Trajectory in Task Space

As can be seen from Figure 3, the trajectory follows a linear path with a positive slope.

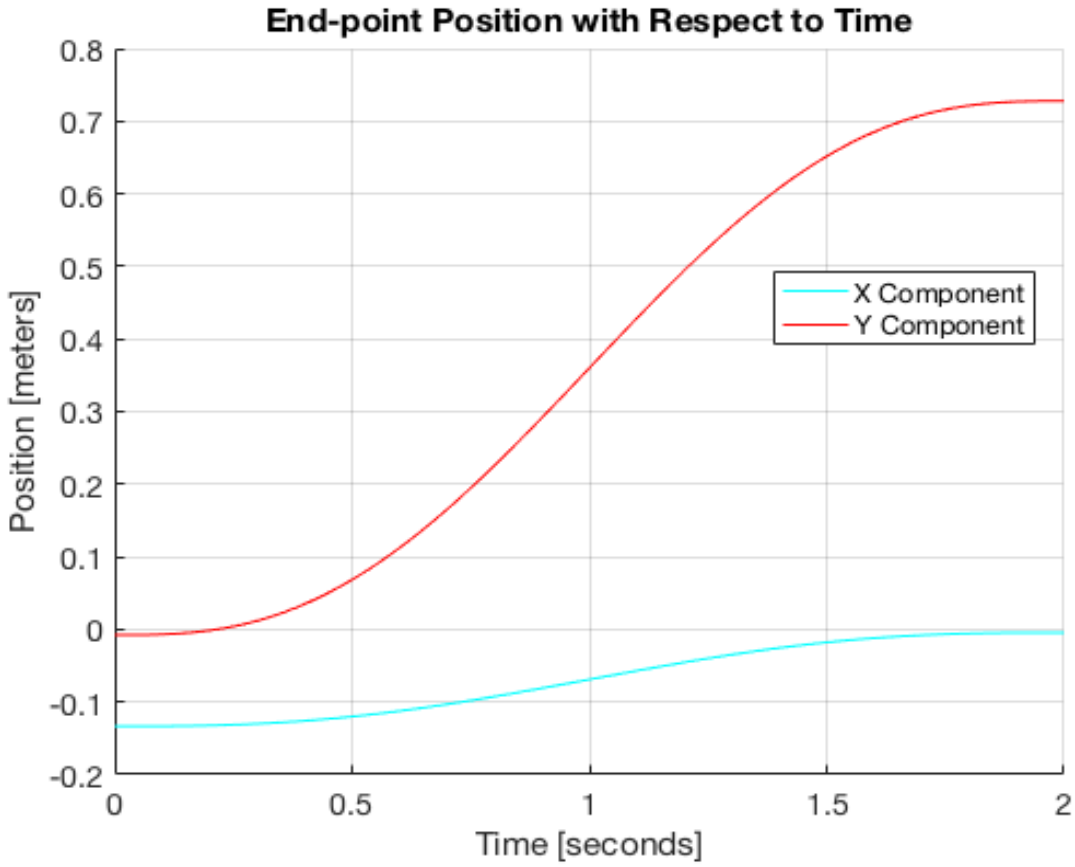

Figure 4: Endpoint Position in Task Space with Respect to Time

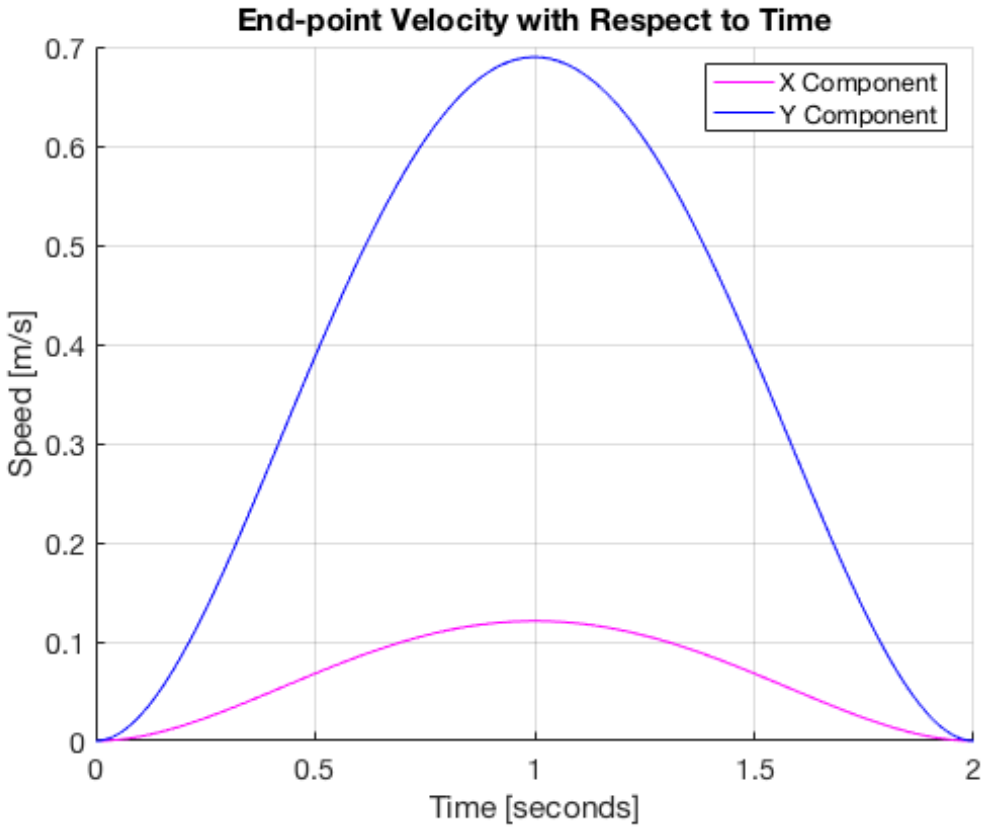

Figure 5: Endpoint Velocity in Task Space with Respect to Time

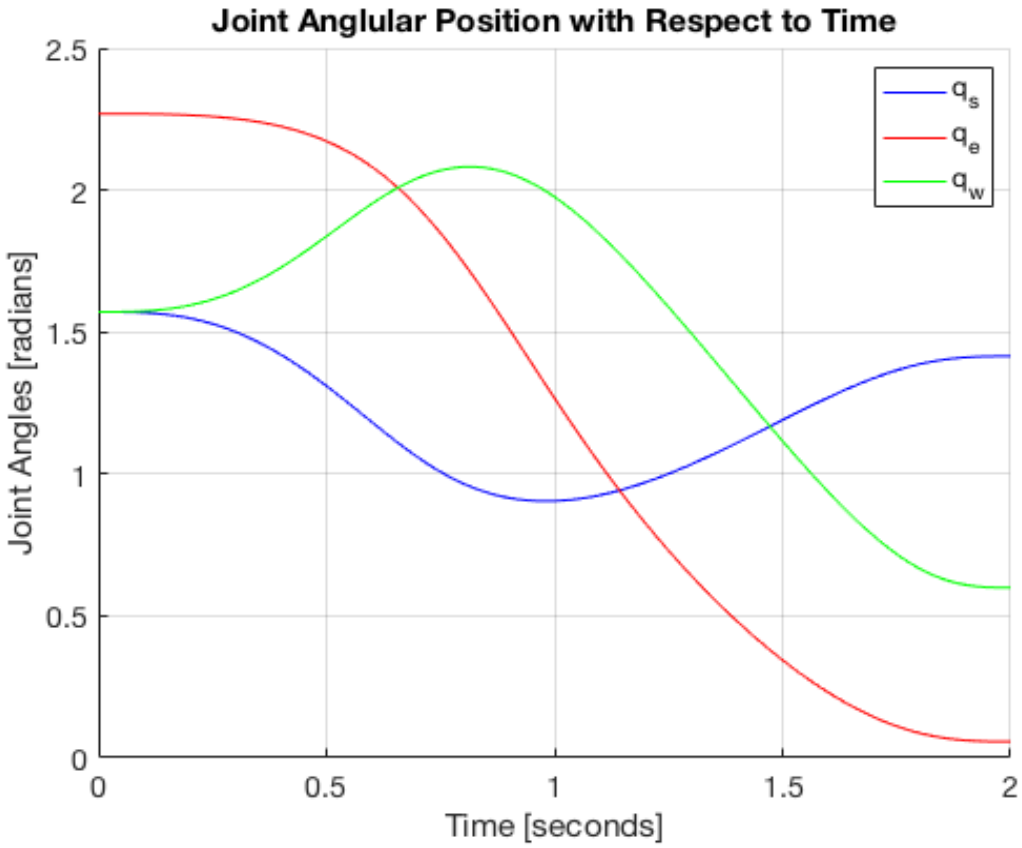

Figure 6: Angular Position in Joint Space with Respect to Time

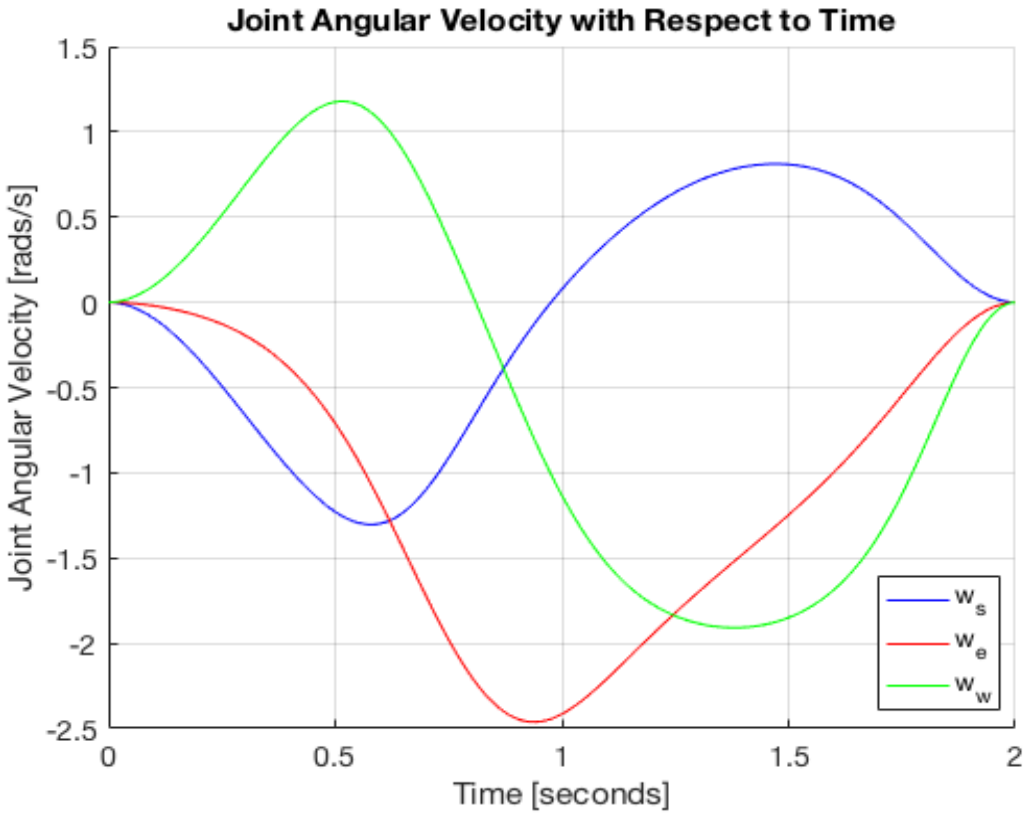

Figure 7: Angular Velocity in Joint Space with Respect to Time

The results, plotted above, were found using Matlab. Below you will see the script used to produce the above results. The entire script is commented, however, I have provided extra explanation before each step to discuss my process.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This script was prepared by:
% Jenna Luchak
% CID: 01429938
% For Human Neuromechanical Control: Tutorial #2
% February 8, 2018
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Clear Workspace and Command Window

clc;
clear;
```

The first step in my code was to define a vector for time such that it would start at 0, end at 2 seconds and increase in intervals of an indicated time step. For my script the time step chosen was 0.01.

```
%% Define the time constraints of the project

ts = 0.01; % Time step in seconds
T = 2; % Total time duration in seconds
t = 0:ts:T; % Time Vector
```

The second step in my code, I defined the joint angular position and the joint angular velocity as empty matrices with 3 rows (one for each joint of the limb – shoulder, elbow, wrist) and T/ts+1 number of columns because that is equivalent to the number of elements in the time vector. The purpose of these empty matrices is that later on in the script, when the joint angle and joint velocity is computed at every time step the value will be printed in these matrices. The first row of the joint angular matrix was defined as the initial angles of the limb in radians. The initial values were given in the assignment question in degrees; therefore I had to convert the value. The initial angles of the shoulder, elbow and wrist were 90˚, 130˚ and 90˚ respectively.

```
%% Define the joint angles and joint velocity matrices
% i.e. Initialize two empty matrices with 3 rows.
% First row represents the shoulder joint, second row the elbow joint and
% third row is the wrist joint values.

q = zeros(3,T/ts+1);
q(:,1) = [deg2rad(90) deg2rad(130) deg2rad(90)]; % Initial angles
q_dot = zeros(3,T/ts+1);
```

The third step in this code defined the length values for each component of the arm. The upper arm was 0.3m, the lower arm was 0.3m and the hand was 0.15m.

```
%% Define the initial lengths of the arm system

l_s = 0.3; % Upper arm length in meters
l_e = 0.3; % Forearm length in meters
```

```
l_w = 0.15; % Hand length in meters
```

The fourth step in this code defined the equations for the end-effector of the arms position and velocity in task space. The position in x and y direction were provided in the assignment. The velocity equations in the x and y direction was determined by taking the directive of the position equations with respect to t. The equations are written below in the code.

```
%% Given Equations for desired position and velocity of the arm endpoint

g = (6*t.^5/(T.^5))-(15*t.^4/(T.^4))+(10*t.^3/(T.^3)); % Intermediary
Equation
g_dot = (30*t.^4/(T.^5))-(60*t.^3/(T.^4))+(30*t.^2/(T.^3)); % Derivative of
intermediary equation

x = -0.1334+0.1289*g; % Desired position in x direction [m]
x_dot = 0.1289*g_dot; % Desired velocity in x direction [m/s]

y = -0.0077+0.7355*g; % Desired position in y direction [m]
y_dot = 0.7355*g_dot; % Desired velocity in y direction [m/s]

% Combine the x and y component velocities into one matrix
velocity = [x_dot;y_dot];
```

The fifth step in this script was to create a for loop so that the joint angular position and joint angular velocity could be computed for every iteration of the time vector previously defined. The joint angular position is updated using the joint angular position and velocity information from the previous time step. The Jacobian equation is computed for every iteration because it is dependent on the current time steps angular position. The joint angular velocity is updated using the pseudo-inverse of the Jacobian and the velocity in task space of the arm. The pseudo inverse of the Jacobian was used to determine joint angular velocity because the Jacobian is not a square matrix.

```
%% Calculate the joint angle profile, speed profile and position
% i.e. use a "for" loop to determine the actual joint angles and
% angular velocities at every time step up to the specified time.

for i=2:1:(T/ts+1)

% Update joint angle matrix
    q(:,i)=q(:,i-1)+q_dot(:,i-1)*ts;

% Calculate the Jacobian
    J=[(-l_s*sin(q(1,i))-l_e*sin(sum(q(1:2,i)))-l_w*sin(sum(q(:,i)))) (-
l_e*sin(sum(q(1:2,i)))-l_w*sin(sum(q(:,i)))) (-l_w*sin(sum(q(:,i))));
(l_s*cos(q(1,i))+l_e*cos(sum(q(1:2,i)))+l_w*cos(sum(q(:,i))))
(l_e*cos(sum(q(1:2,i)))+l_w*cos(sum(q(:,i)))) (l_w*cos(sum(q(:,i))))];

% Calculate the Psuedo Inverse of the Jacobian
    pseudo = pinv(J);

% Compute the angular velocity of the arm
    q_dot(:,i)=pseudo*velocity(:,i);

end
```

The final step of this script was to print out the needed results into plots. In addition to their corresponding x and y data points, each plot was printed on a different figure, with labeled

axes, a title and grid lines.

```
%% Print a plot of the end-effector trajectory
figure(1)
plot(x,y);
grid on
xlabel('X-position [m]'); % x axis title
ylabel('Y-position [m]'); % y axis title
title('End-point Trajectory in Task Space');
set(gca,'FontSize',14);

%% Print a plot of the end-point position with respect to time
figure(2)
grid on
hold on
plot(t,x,'c'); % x component of end effector position
plot(t,y,'r'); % y component of end effector position
legend('X Component','Y Component','location','best');
xlabel('Time [seconds]'); % x axis title
ylabel('Position [meters]'); % y axis title
title('End-point Position with Respect to Time');
set(gca,'FontSize',14)
hold off

%% Print a plot of the end-point velocity with respect to time
figure(3)
grid on
hold on
plot(t,x_dot,'m'); % x component of end effector speed
plot(t,y_dot,'b'); % y component of end effector speed
legend('X Component','Y Component','location','best');
xlabel('Time [seconds]'); % x axis title
ylabel('Speed [m/s]'); % y axis title
title('End-point Velocity with Respect to Time');
set(gca,'FontSize',14)
hold off

%% Print a plot of the joint angles with respect to time
figure(4)
grid on
hold on
plot(t,q(1,:),'b'); % Shoulder joint
plot(t,q(2,:),'r'); % Elbow joint
plot(t,q(3,:),'g'); % Wrist joint
legend('q_s','q_e','q_w','location','best');
xlabel('Time [seconds]'); % x axis title
ylabel('Joint Angles [radians]'); % y axis title
title('Joint Anglular Position with Respect to Time');
set(gca,'FontSize',14)
hold off

%% Print a plot of the angular joint velocity with respect to time
figure(5)
hold on
grid on
plot(t,q_dot(1,:),'b'); % Shoulder joint velocity
plot(t,q_dot(2,:),'r'); % Elbow joint velocity
plot(t,q_dot(3,:),'g'); % Wrist joint velocity
legend('w_s','w_e','w_w','location','best');
xlabel('Time [seconds]'); % x axis title
ylabel('Joint Angular Velocity [rads/s]'); % y axis title
```

```
title('Joint Angular Velocity with Respect to Time');
set(gca,'FontSize',14)
hold off

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% End of Script
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```