Arm Motion Modeling

System Description

A double-pendulum system hanging in gravity is shown in the figure above. $q = [\theta_1, \theta_2]$ are the system configuration variables. We assume the z-axis is pointing out from the screen/paper, thus the positive direction of rotation is counter-clockwise. The solution steps are:

1. Computing the Lagrangian of the system.
2. Computing the Euler-Lagrange equations, and solve them for $\ddot{\theta}_1$ and $\ddot{\theta}_2$.
3. Numerically evaluating the solutions for τ_1 and τ_2, and simulating the system for θ_1, θ_2, $\dot{\theta}_1$, $\dot{\theta}_2$, $\ddot{\theta}_1$ and $\ddot{\theta}_2$.
4. Animating the simulation.

In [14]:

```
from IPython.core.display import HTML
display(HTML("<table><tr><td><img src='images/double-pendulum.jpg' width=500' height='350'></table>"))
```

Import Libraries and Define System Constants

Import libraries:

In [2]:

```
# Imports required for data processing
import os
import csv
import pandas as pd

# Imports required for dynamics calculations
import sympy
from sympy.abc import t
from sympy import symbols, Eq, Function, solve, sin, cos, Matrix, Subs, substitution, Derivative, simplify, symbols, lambdify
import math
from math import pi
import numpy as np
import matplotlib.pyplot as plt

# Imports required for animation
from plotly.offline import init_notebook_mode, iplot
from IPython.display import display, HTML
import plotly.graph_objects as go
```

Define the system's constants:

In [3]:

```
# Masses, length and center-of-mass positions (calculated using the lab measurements)
# Mass calculations (mass unit is kg)
# m_body = 90.6                          # Average weights for American adult male
#                                        # from "Anthropometric Reference Data for Children and Adults:
#                                        # United States, 2015–2018"
m_body_dict = {'ID': 51, 'JD': 79.5, 'JR': 76, 'KS': 59.3, 'KW': 63.8, 'LC': 61.2,
               'LD': 97.3, 'LS': 82.2, 'MK': 93.5, 'MV': 98.5, 'SM': 68.5, 'TD': 70,
               'TM': 66.2}

# m_upper_arm = 0.028 * m_body           # Average upper arm weights relative to body weight, from “Biomechanics
#                                        # and Motor Control of Human Movement” by David Winter (2009), 4th edition
m_upper_arm_dict = {'ID': 0.028 * m_body_dict['ID'], 'JD': 0.028 * m_body_dict['JD'],
                    'JR': 0.028 * m_body_dict['JR'], 'KS': 0.028 * m_body_dict['KS'],
                    'KW': 0.028 * m_body_dict['KW'], 'LC': 0.028 * m_body_dict['LC'],
                    'LD': 0.028 * m_body_dict['LD'], 'LS': 0.028 * m_body_dict['LS'],
                    'MK': 0.028 * m_body_dict['MK'], 'MV': 0.028 * m_body_dict['MV'],
                    'SM': 0.028 * m_body_dict['SM'], 'TD': 0.028 * m_body_dict['TD'],
                    'TM': 0.028 * m_body_dict['TM']}

m_lower_arm = 0.7395                   # Average lower prosthetics weights, calculated using lab measurements



# Arm length calculations (length unit is m)
# H_body = 1.769                         # Average height for American adult male, from “Height and body-mass
#                                        # index trajectories of school-aged children and adolescents from
#                                        # 1985 to 2019 in 200 countries and territories: a pooled analysis
#                                        # of 2181 population-based studies with 65 million participants”
H_body_dict = {'ID': 1.62, 'JD': 1.76, 'JR': 1.77, 'KS': 1.64, 'KW': 1.62, 'LC': 1.58,
               'LD': 1.875, 'LS': 1.635, 'MK': 1.78, 'MV': 1.805, 'SM': 1.79, 'TD': 1.69,
               'TM': 1.735}

# L_upper_arm = 0.186 * H_body           # Average upper arm length relative to body height
#                                        # from “Biomechanics and Motor Control of Human Movement” by David
#                                        # Winter (2009), 4th edition
L_upper_arm_dict = {'ID': 0.186 * H_body_dict['ID'], 'JD': 0.186 * H_body_dict['JD'],
                    'JR': 0.186 * H_body_dict['JR'], 'KS': 0.186 * H_body_dict['KS'],
                    'KW': 0.186 * H_body_dict['KW'], 'LC': 0.186 * H_body_dict['LC'],
                    'LD': 0.186 * H_body_dict['LD'], 'LS': 0.186 * H_body_dict['LS'],
                    'MK': 0.186 * H_body_dict['MK'], 'MV': 0.186 * H_body_dict['MV'],
```

```
                     'SM': 0.186 * H_body_dict['SM'], 'TD': 0.186 * H_body_dict['TD'],
                     'TM': 0.186 * H_body_dict['TM']}

L_lower_arm = 0.42                     # Average lower prosthetics length, calculated using lab measurements



# Arm center of mass length calculations (length unit is m)
# L_upper_arm_COM = 0.436 * L_upper_arm  # Average upper arm length from shoulder to center of mass relative
#                                        # to upper arm length, from “Biomechanics and Motor Control of Human
#                                        # Movement” by David Winter (2009), 4th edition
L_upper_arm_COM_dict = {'ID': 0.436 * L_upper_arm_dict['ID'], 'JD': 0.436 * L_upper_arm_dict['JD'],
                        'JR': 0.436 * L_upper_arm_dict['JR'], 'KS': 0.436 * L_upper_arm_dict['KS'],
                        'KW': 0.436 * L_upper_arm_dict['KW'], 'LC': 0.436 * L_upper_arm_dict['LC'],
                        'LD': 0.436 * L_upper_arm_dict['LD'], 'LS': 0.436 * L_upper_arm_dict['LS'],
                        'MK': 0.436 * L_upper_arm_dict['MK'], 'MV': 0.436 * L_upper_arm_dict['MV'],
                        'SM': 0.436 * L_upper_arm_dict['SM'], 'TD': 0.436 * L_upper_arm_dict['TD'],
                        'TM': 0.436 * L_upper_arm_dict['TM']}

L_lower_arm_COM = 0.2388               # Average lower prosthetics length from elbow to center of mass,
                                       # calculated using lab measurements
```

Extracting Data

Extracting angles data and computing angular valocities and angular accelerations from the angles:

In [4]:
```
def calculate_Vel(Ang_list, time_list, index):
    return ((Ang_list[index+1] - Ang_list[index])
          / (time_list[index+1] - time_list[index]))


def calculate_Acc(Vel_list, time_list, index):
    return ((Vel_list[index+1] - Vel_list[index])
          / (time_list[index+1] - time_list[index]))


data_csv_dir = './data/Control Data/CSV Converted Files'
frame_frequency = 120
print("current directory: ", os.getcwd())

participants_list = []
time_list = []
Elbow_Ang_list = []
Shl_Flex_Ang_list = []
Elbow_Vel_list = []
Shl_Flex_Vel_list = []
Elbow_Acc_list = []
Shl_Flex_Acc_list = []

for file in os.listdir(data_csv_dir):
    file_name = file.split(".")[0]
    participant_name = file.split("_")[0]

    if file.endswith(".csv"):
        frame = 0
        file_time_list = []
        file_R_Elbow_Ang_list = []
        file_R_Shl_Flex_Ang_list = []
        file_L_Elbow_Ang_list = []
        file_L_Shl_Flex_Ang_list = []
        file_R_Elbow_Vel_list = []
        file_R_Shl_Flex_Vel_list = []
        file_L_Elbow_Vel_list = []
        file_L_Shl_Flex_Vel_list = []
        file_R_Elbow_Acc_list = []
        file_R_Shl_Flex_Acc_list = []
        file_L_Elbow_Acc_list = []
        file_L_Shl_Flex_Acc_list = []

        data_path = os.path.join(data_csv_dir, file)

        # Cutting out weird data behavior on data edges
        if file == 'TD_WN7.csv':
            data_rows = open(data_path).read().strip().split("\n")[40:]
        elif file == 'TD_WN4.csv':
            data_rows = open(data_path).read().strip().split("\n")[24:-12]
        elif file == 'TD_WN11.csv':
            data_rows = open(data_path).read().strip().split("\n")[24:-3]
        else:
            data_rows = open(data_path).read().strip().split("\n")[24:]

        # Extract time [sec], elbow angles [rad], and shoulder angles [rad] from data
        for row in data_rows:
            splitted_row = row.strip().split("\t")

            # Check if loop finished all data
            if len(splitted_row) < 80:
                break

            file_time_list.append(frame/frame_frequency)
            file_R_Elbow_Ang_list.append(float(splitted_row[9]) * 2*pi/360)
            file_R_Shl_Flex_Ang_list.append(float(splitted_row[11]) * 2*pi/360)
            file_L_Elbow_Ang_list.append(float(splitted_row[21]) * 2*pi/360)
            file_L_Shl_Flex_Ang_list.append(float(splitted_row[23]) * 2*pi/360)
            frame += 1

        # Extract elbow and shoulder velocities [rad/sec] from angles
        for i in range(len(file_time_list) - 1):
            R_Elbow_Vel = calculate_Vel(file_R_Elbow_Ang_list, file_time_list, i)
            R_Shl_Flex_Vel = calculate_Vel(file_R_Shl_Flex_Ang_list, file_time_list, i)
            L_Elbow_Vel = calculate_Vel(file_L_Elbow_Ang_list, file_time_list, i)
            L_Shl_Flex_Vel = calculate_Vel(file_L_Shl_Flex_Ang_list, file_time_list, i)

            file_R_Elbow_Vel_list.append(R_Elbow_Vel)
            file_R_Shl_Flex_Vel_list.append(R_Shl_Flex_Vel)
            file_L_Elbow_Vel_list.append(L_Elbow_Vel)
            file_L_Shl_Flex_Vel_list.append(L_Shl_Flex_Vel)

        # Extract elbow and shoulder Accelerations [rad/sec^2] from velocities
        for i in range(len(file_time_list) - 2):
```

```
            R_Elbow_Acc = calculate_Acc(file_R_Elbow_Vel_list, file_time_list, i)
            R_Shl_Flex_Acc = calculate_Acc(file_R_Shl_Flex_Vel_list, file_time_list, i)
            L_Elbow_Acc = calculate_Acc(file_L_Elbow_Vel_list, file_time_list, i)
            L_Shl_Flex_Acc = calculate_Acc(file_L_Shl_Flex_Vel_list, file_time_list, i)

            file_R_Elbow_Acc_list.append(R_Elbow_Acc)
            file_R_Shl_Flex_Acc_list.append(R_Shl_Flex_Acc)
            file_L_Elbow_Acc_list.append(L_Elbow_Acc)
            file_L_Shl_Flex_Acc_list.append(L_Shl_Flex_Acc)

        # Adjust lists length
        file_time_list = file_time_list[:-2]
        file_R_Elbow_Ang_list = file_R_Elbow_Ang_list[:-2]
        file_R_Shl_Flex_Ang_list = file_R_Shl_Flex_Ang_list[:-2]
        file_L_Elbow_Ang_list = file_L_Elbow_Ang_list[:-2]
        file_L_Shl_Flex_Ang_list = file_L_Shl_Flex_Ang_list[:-2]

        file_R_Elbow_Vel_list = file_R_Elbow_Vel_list[:-1]
        file_R_Shl_Flex_Vel_list = file_R_Shl_Flex_Vel_list[:-1]
        file_L_Elbow_Vel_list = file_L_Elbow_Vel_list[:-1]
        file_L_Shl_Flex_Vel_list = file_L_Shl_Flex_Vel_list[:-1]

        participants_list.append(participant_name)
        participants_list.append(participant_name)

        time_list.append(file_time_list)
        time_list.append(file_time_list)

        Elbow_Ang_list.append(file_R_Elbow_Ang_list)
        Shl_Flex_Ang_list.append(file_R_Shl_Flex_Ang_list)
        Elbow_Ang_list.append(file_L_Elbow_Ang_list)
        Shl_Flex_Ang_list.append(file_L_Shl_Flex_Ang_list)
        Elbow_Vel_list.append(file_R_Elbow_Vel_list)
        Shl_Flex_Vel_list.append(file_R_Shl_Flex_Vel_list)
        Elbow_Vel_list.append(file_L_Elbow_Vel_list)
        Shl_Flex_Vel_list.append(file_L_Shl_Flex_Vel_list)
        Elbow_Acc_list.append(file_R_Elbow_Acc_list)
        Shl_Flex_Acc_list.append(file_R_Shl_Flex_Acc_list)
        Elbow_Acc_list.append(file_L_Elbow_Acc_list)
        Shl_Flex_Acc_list.append(file_L_Shl_Flex_Acc_list)
```

```
current directory:  /home/yael/Documents/MSR_Courses/Spring_2021/ME499-Final_Project/Motorized-Prosthetic-Arm
```

System Modeling

Computing the Lagrangian of the system:

In [5]:
```
m1, m2, g, R1, R1_COM, R2, R2_COM = symbols(r'm1, m2, g, R1, R1_COM, R2, R2_COM')

# The system torque variables as function of t
tau1 = Function(r'tau1')(t)
tau2 = Function(r'tau2')(t)

# The system configuration variables as function of t
theta1 = Function(r'theta1')(t)
theta2 = Function(r'theta2')(t)

# The velocity as derivative of position wrt t
theta1_dot = theta1.diff(t)
theta2_dot = theta2.diff(t)

# The acceleration as derivative of velocity wrt t
theta1_ddot = theta1_dot.diff(t)
theta2_ddot = theta2_dot.diff(t)

# Converting the polar coordinates to cartesian coordinates
x1 = R1_COM*sin(theta1)
x2 = R1*sin(theta1) + R2_COM*sin(theta1 + theta2)

y1 = -R1_COM*cos(theta1)
y2 = -R1*cos(theta1) - R2_COM*cos(theta1 + theta2)

# Calculating the kinetic and potential energy of the system
KE = 1/2*m1*((x1.diff(t))**2 + (y1.diff(t))**2) + 1/2*m2*((x2.diff(t))**2 + (y2.diff(t))**2)
PE = m1*g*y1 + m2*g*y2

# Computing the Lagrangian
L = simplify(KE - PE)
print('L: ')
display(L)
```

```
L:
```

$$0.5R_{1COM}^2 m_1 \left(\frac{d}{dt}\theta_1(t)\right)^2 + R_{1COM} g m_1 \cos(\theta_1(t)) + g m_2 \left(R_1 \cos(\theta_1(t)) + R_{2COM} \cos(\theta_1(t) + \theta_2(t))\right)$$
$$+ 0.5 m_2 \left(R_1^2 \left(\frac{d}{dt}\theta_1(t)\right)^2 + 2R_1 R_{2COM} \cos(\theta_2(t)) \left(\frac{d}{dt}\theta_1(t)\right)^2 + 2R_1 R_{2COM} \cos(\theta_2(t)) \frac{d}{dt}\theta_1(t) \frac{d}{dt}\theta_2(t) + R_{2COM}^2 \left(\frac{d}{dt}\theta_1(t)\right)^2 + 2R_{2COM}^2 \frac{d}{dt}\theta_1(t) \frac{d}{dt}\theta_2(t) + R_{2COM}^2 \left(\frac{d}{dt}\theta_2(t)\right)^2\right)$$

Computing the Euler-Lagrange equations:

In [6]:
```
# Define the derivative of L wrt the functions: x, xdot
L_dtheta1 = L.diff(theta1)
L_dtheta2 = L.diff(theta2)

L_dtheta1_dot = L.diff(theta1_dot)
L_dtheta2_dot = L.diff(theta2_dot)

# Define the derivative of L_dxdot wrt to time t
L_dtheta1_dot_dt = L_dtheta1_dot.diff(t)
L_dtheta2_dot_dt = L_dtheta2_dot.diff(t)

# Define the left hand side of the the Euler-Lagrange as a matrix
lhs = Matrix([simplify(L_dtheta1_dot_dt - L_dtheta1),
              simplify(L_dtheta2_dot_dt - L_dtheta2)])

# Define the right hand side of the the Euler-Lagrange as a Matrix
rhs = Matrix([tau1, tau2])

# Compute the Euler-Lagrange equations as a matrix
```

```
EL_eqns = Eq(lhs, rhs)

print('Euler-Lagrange matrix for this systems:')
display(EL_eqns)
```

Euler-Lagrange matrix for this systems:

$$\left[\begin{matrix}1.0R_{1COM}^{2}m_{1}\frac{d^{2}}{dt^{2}}\theta_{1}(t)+R_{1COM}gm_{1}\sin\left(\theta_{1}(t)\right)+gm_{2}\left(R_{1}\sin\left(\theta_{1}(t)\right)+R_{2COM}\sin\left(\theta_{1}(t)+\theta_{2}(t)\right)\right)\\+m_{2}\left(R_{1}^{2}\frac{d^{2}}{dt^{2}}\theta_{1}(t)-2R_{1}R_{2COM}\sin\left(\theta_{2}(t)\right)\frac{d}{dt}\theta_{1}(t)\frac{d}{dt}\theta_{2}(t)-R_{1}R_{2COM}\sin\left(\theta_{2}(t)\right)\left(\frac{d}{dt}\theta_{2}(t)\right)^{2}+2R_{1}R_{2COM}\cos\left(\theta_{2}(t)\right)\frac{d^{2}}{dt^{2}}\theta_{1}(t)+R_{1}R_{2COM}\cos\left(\theta_{2}(t)\right)\frac{d^{2}}{dt^{2}}\theta_{2}(t)\right.\\\left.+R_{2COM}^{2}\frac{d^{2}}{dt^{2}}\theta_{1}(t)+R_{2COM}^{2}\frac{d^{2}}{dt^{2}}\theta_{2}(t)\right)\\R_{2COM}m_{2}\left(R_{1}\sin\left(\theta_{2}(t)\right)\left(\frac{d}{dt}\theta_{1}(t)\right)^{2}+R_{1}\cos\left(\theta_{2}(t)\right)\frac{d^{2}}{dt^{2}}\theta_{1}(t)+R_{2COM}\frac{d^{2}}{dt^{2}}\theta_{1}(t)+R_{2COM}\frac{d^{2}}{dt^{2}}\theta_{2}(t)+g\sin\left(\theta_{1}(t)+\theta_{2}(t)\right)\right)\end{matrix}\right]$$

$$=\left[\begin{matrix}\tau_{1}(t)\\\tau_{2}(t)\end{matrix}\right]$$

Solve the equations for τ_1 and τ_2:

In [7]:

```
# Solve the Euler-Lagrange equations for the shoulder and elbow torques
T = Matrix([tau1, tau2])
soln = solve(EL_eqns, T, dict=True)

# Initialize the solutions
solution = [0, 0]
i = 0

for sol in soln:
    for v in T:
        solution[i] = simplify(sol[v])
        display(Eq(T[i], solution[i]))
        i =+ 1
```

$$\tau_{1}(t)=R_{1}^{2}m_{2}\frac{d^{2}}{dt^{2}}\theta_{1}(t)-2.0R_{1}R_{2COM}m_{2}\sin\left(\theta_{2}(t)\right)\frac{d}{dt}\theta_{1}(t)\frac{d}{dt}\theta_{2}(t)-R_{1}R_{2COM}m_{2}\sin\left(\theta_{2}(t)\right)\left(\frac{d}{dt}\theta_{2}(t)\right)^{2}+2.0R_{1}R_{2COM}m_{2}\cos\left(\theta_{2}(t)\right)\frac{d^{2}}{dt^{2}}\theta_{1}(t)+R_{1}R_{2COM}m_{2}\cos\left(\theta_{2}(t)\right)\frac{d^{2}}{dt^{2}}\theta_{2}(t)+R_{1}gm_{2}\sin\left(\theta_{1}(t)\right)+R_{1COM}^{2}m_{1}\frac{d^{2}}{dt^{2}}\theta_{1}(t)+R_{1COM}gm_{1}\sin\left(\theta_{1}(t)\right)+R_{2COM}^{2}m_{2}\frac{d^{2}}{dt^{2}}\theta_{1}(t)+R_{2COM}^{2}m_{2}\frac{d^{2}}{dt^{2}}\theta_{2}(t)+R_{2COM}gm_{2}\sin\left(\theta_{1}(t)+\theta_{2}(t)\right)$$

$$\tau_{2}(t)=R_{2COM}m_{2}\left(R_{1}\sin\left(\theta_{2}(t)\right)\left(\frac{d}{dt}\theta_{1}(t)\right)^{2}+R_{1}\cos\left(\theta_{2}(t)\right)\frac{d^{2}}{dt^{2}}\theta_{1}(t)+R_{2COM}\frac{d^{2}}{dt^{2}}\theta_{1}(t)+R_{2COM}\frac{d^{2}}{dt^{2}}\theta_{2}(t)+g\sin\left(\theta_{1}(t)+\theta_{2}(t)\right)\right)$$

Simulating the system:

In [8]:

```
# Substitute the derivative variables with a dummy variables and plug-in the constants
solution_0_subs = solution[0]
solution_1_subs = solution[1]

theta1_dot_dummy = symbols('dtheta1')
theta2_dot_dummy = symbols('dtheta2')
theta1_ddot_dummy = symbols('ddtheta1')
theta2_ddot_dummy = symbols('ddtheta2')

# solution_0_subs = solution_0_subs.subs([(m1, m_upper_arm), (m2, m_lower_arm), (R1, L_upper_arm), (R2, L_lower_arm), (R1_COM, L_upper_arm_COM), (R2_COM, L_lower_arm_COM)
# solution_1_subs = solution_1_subs.subs([(m1, m_upper_arm), (m2, m_lower_arm), (R1, L_upper_arm), (R2, L_lower_arm), (R1_COM, L_upper_arm_COM), (R2_COM, L_lower_arm_COM)

solution_0_subs = solution_0_subs.subs([(g, 9.81)])
solution_1_subs = solution_1_subs.subs([(g, 9.81)])

# display(Eq(T[0], solution_0_subs))
# display(Eq(T[1], solution_1_subs))

solution_0_subs = solution_0_subs.subs([((theta1.diff(t)).diff(t), theta1_ddot_dummy),
                                        ((theta2.diff(t)).diff(t), theta2_ddot_dummy)])
solution_1_subs = solution_1_subs.subs([((theta1.diff(t)).diff(t), theta1_ddot_dummy),
                                        ((theta2.diff(t)).diff(t), theta2_ddot_dummy)])

solution_0_subs = solution_0_subs.subs([(theta1.diff(t), theta1_dot_dummy),
                                        (theta2.diff(t), theta2_dot_dummy)])
solution_1_subs = solution_1_subs.subs([(theta1.diff(t), theta1_dot_dummy),
                                        (theta2.diff(t), theta2_dot_dummy)])

# Lambdify the thetas and its derivatives
func1 = lambdify([theta1, theta2, theta1_dot_dummy, theta2_dot_dummy, theta1_ddot_dummy,
                  theta2_ddot_dummy, m1, m2, R1, R2, R1_COM, R2_COM], solution_0_subs, modules = sympy)
func2 = lambdify([theta1, theta2, theta1_dot_dummy, theta2_dot_dummy, theta1_ddot_dummy,
                  theta2_ddot_dummy, m1, m2, R1, R2, R1_COM, R2_COM], solution_1_subs, modules = sympy)

# Initialize the torque and power lists
Shl_Flex_tau_list, Elbow_tau_list = [], []
Shl_Flex_power_list, Elbow_power_list = [], []

for i in range(len(time_list)):
    # Initialize the torque and power lists
    tau1_list, tau2_list = [], []
    power1_list, power2_list = [], []

    t_list = time_list[i]
    theta1_list = Shl_Flex_Ang_list[i]
    theta2_list = Elbow_Ang_list[i]
    dtheta1_list = Shl_Flex_Vel_list[i]
    dtheta2_list = Elbow_Vel_list[i]
    ddtheta1_list = Shl_Flex_Acc_list[i]
    ddtheta2_list = Elbow_Acc_list[i]

    # Plug-in the angles, angular velocities and angular accelerations for every time step to find the torques
    for j in range(len(t_list)):
        tau1_list.append(func1(theta1_list[j], theta2_list[j], dtheta1_list[j], dtheta2_list[j],
                               ddtheta1_list[j], ddtheta2_list[j], m_upper_arm_dict[participants_list[i]],
                               m_lower_arm, L_upper_arm_dict[participants_list[i]], L_lower_arm,
                               L_upper_arm_COM_dict[participants_list[i]], L_lower_arm_COM))

        tau2_list.append(func2(theta1_list[j], theta2_list[j], dtheta1_list[j], dtheta2_list[j],
                               ddtheta1_list[j], ddtheta2_list[j], m_upper_arm_dict[participants_list[i]],
                               m_lower_arm, L_upper_arm_dict[participants_list[i]], L_lower_arm,
                               L_upper_arm_COM_dict[participants_list[i]], L_lower_arm_COM))
```

```python
        # Calculate the power required to reach the required angular velocities and joints torques for every time step
        power1_list.append(dtheta1_list[j] * tau1_list[j])
        power2_list.append(dtheta2_list[j] * tau2_list[j])

    Shl_Flex_tau_list.append(tau1_list)
    Elbow_tau_list.append(tau2_list)

    Shl_Flex_power_list.append(power1_list)
    Elbow_power_list.append(power2_list)

    print(f"Trial {i}/{len(time_list)-1} finished \t maximum torque is {format(max(tau2_list), '.3f')} [Nm]\t maximum power is {format(max(power2_list), '.3f')} [W]")
```

```
Trial 0/203 finished     maximum torque is 1.929 [Nm]    maximum power is 0.968 [W]
Trial 1/203 finished     maximum torque is 2.216 [Nm]    maximum power is 2.258 [W]
Trial 2/203 finished     maximum torque is 3.126 [Nm]    maximum power is 3.431 [W]
Trial 3/203 finished     maximum torque is 3.753 [Nm]    maximum power is 2.448 [W]
Trial 4/203 finished     maximum torque is 2.113 [Nm]    maximum power is 2.634 [W]
Trial 5/203 finished     maximum torque is 2.409 [Nm]    maximum power is 4.074 [W]
Trial 6/203 finished     maximum torque is 1.745 [Nm]    maximum power is 2.063 [W]
Trial 7/203 finished     maximum torque is 2.379 [Nm]    maximum power is 1.769 [W]
Trial 8/203 finished     maximum torque is 2.085 [Nm]    maximum power is 1.952 [W]
Trial 9/203 finished     maximum torque is 2.202 [Nm]    maximum power is 1.177 [W]
Trial 10/203 finished    maximum torque is 2.498 [Nm]    maximum power is 2.797 [W]
Trial 11/203 finished    maximum torque is 3.138 [Nm]    maximum power is 3.980 [W]
Trial 12/203 finished    maximum torque is 1.827 [Nm]    maximum power is 2.661 [W]
Trial 13/203 finished    maximum torque is 1.831 [Nm]    maximum power is 1.548 [W]
Trial 14/203 finished    maximum torque is 2.643 [Nm]    maximum power is 2.506 [W]
Trial 15/203 finished    maximum torque is 1.949 [Nm]    maximum power is 1.771 [W]
Trial 16/203 finished    maximum torque is 2.293 [Nm]    maximum power is 2.658 [W]
Trial 17/203 finished    maximum torque is 2.320 [Nm]    maximum power is 4.121 [W]
Trial 18/203 finished    maximum torque is 1.781 [Nm]    maximum power is 2.715 [W]
Trial 19/203 finished    maximum torque is 2.058 [Nm]    maximum power is 3.501 [W]
Trial 20/203 finished    maximum torque is 2.971 [Nm]    maximum power is 1.741 [W]
Trial 21/203 finished    maximum torque is 2.432 [Nm]    maximum power is 3.929 [W]
Trial 22/203 finished    maximum torque is 2.289 [Nm]    maximum power is 2.062 [W]
Trial 23/203 finished    maximum torque is 1.933 [Nm]    maximum power is 1.531 [W]
Trial 24/203 finished    maximum torque is 2.088 [Nm]    maximum power is 1.166 [W]
Trial 25/203 finished    maximum torque is 2.189 [Nm]    maximum power is 2.242 [W]
Trial 26/203 finished    maximum torque is 1.683 [Nm]    maximum power is 2.473 [W]
Trial 27/203 finished    maximum torque is 1.957 [Nm]    maximum power is 1.527 [W]
Trial 28/203 finished    maximum torque is 1.939 [Nm]    maximum power is 1.911 [W]
Trial 29/203 finished    maximum torque is 1.862 [Nm]    maximum power is 1.182 [W]
Trial 30/203 finished    maximum torque is 1.635 [Nm]    maximum power is 2.278 [W]
Trial 31/203 finished    maximum torque is 1.577 [Nm]    maximum power is 1.567 [W]
Trial 32/203 finished    maximum torque is 2.348 [Nm]    maximum power is 2.592 [W]
Trial 33/203 finished    maximum torque is 2.109 [Nm]    maximum power is 4.635 [W]
Trial 34/203 finished    maximum torque is 1.919 [Nm]    maximum power is 1.520 [W]
Trial 35/203 finished    maximum torque is 2.203 [Nm]    maximum power is 2.940 [W]
Trial 36/203 finished    maximum torque is 1.815 [Nm]    maximum power is 1.817 [W]
Trial 37/203 finished    maximum torque is 1.562 [Nm]    maximum power is 1.380 [W]
Trial 38/203 finished    maximum torque is 1.874 [Nm]    maximum power is 1.619 [W]
Trial 39/203 finished    maximum torque is 2.076 [Nm]    maximum power is 2.065 [W]
Trial 40/203 finished    maximum torque is 2.057 [Nm]    maximum power is 1.557 [W]
Trial 41/203 finished    maximum torque is 2.135 [Nm]    maximum power is 1.611 [W]
Trial 42/203 finished    maximum torque is 1.656 [Nm]    maximum power is 2.253 [W]
Trial 43/203 finished    maximum torque is 1.767 [Nm]    maximum power is 1.735 [W]
Trial 44/203 finished    maximum torque is 3.006 [Nm]    maximum power is 1.578 [W]
Trial 45/203 finished    maximum torque is 3.526 [Nm]    maximum power is 2.605 [W]
Trial 46/203 finished    maximum torque is 2.130 [Nm]    maximum power is 1.060 [W]
Trial 47/203 finished    maximum torque is 2.309 [Nm]    maximum power is 2.922 [W]
Trial 48/203 finished    maximum torque is 2.114 [Nm]    maximum power is 2.193 [W]
Trial 49/203 finished    maximum torque is 1.601 [Nm]    maximum power is 1.672 [W]
Trial 50/203 finished    maximum torque is 3.968 [Nm]    maximum power is 3.813 [W]
Trial 51/203 finished    maximum torque is 3.439 [Nm]    maximum power is 4.145 [W]
Trial 52/203 finished    maximum torque is 2.384 [Nm]    maximum power is 1.641 [W]
Trial 53/203 finished    maximum torque is 1.807 [Nm]    maximum power is 1.430 [W]
Trial 54/203 finished    maximum torque is 1.781 [Nm]    maximum power is 2.892 [W]
Trial 55/203 finished    maximum torque is 1.731 [Nm]    maximum power is 2.061 [W]
Trial 56/203 finished    maximum torque is 2.325 [Nm]    maximum power is 2.868 [W]
Trial 57/203 finished    maximum torque is 2.276 [Nm]    maximum power is 4.228 [W]
Trial 58/203 finished    maximum torque is 2.123 [Nm]    maximum power is 2.077 [W]
Trial 59/203 finished    maximum torque is 2.041 [Nm]    maximum power is 1.953 [W]
Trial 60/203 finished    maximum torque is 2.244 [Nm]    maximum power is 1.126 [W]
Trial 61/203 finished    maximum torque is 2.346 [Nm]    maximum power is 2.143 [W]
Trial 62/203 finished    maximum torque is 3.548 [Nm]    maximum power is 2.410 [W]
Trial 63/203 finished    maximum torque is 3.619 [Nm]    maximum power is 2.826 [W]
Trial 64/203 finished    maximum torque is 2.534 [Nm]    maximum power is 2.340 [W]
Trial 65/203 finished    maximum torque is 2.335 [Nm]    maximum power is 4.337 [W]
Trial 66/203 finished    maximum torque is 1.971 [Nm]    maximum power is 1.505 [W]
Trial 67/203 finished    maximum torque is 2.061 [Nm]    maximum power is 2.517 [W]
Trial 68/203 finished    maximum torque is 2.376 [Nm]    maximum power is 3.009 [W]
Trial 69/203 finished    maximum torque is 2.875 [Nm]    maximum power is 4.121 [W]
Trial 70/203 finished    maximum torque is 1.904 [Nm]    maximum power is 1.072 [W]
Trial 71/203 finished    maximum torque is 1.942 [Nm]    maximum power is 1.714 [W]
Trial 72/203 finished    maximum torque is 2.201 [Nm]    maximum power is 1.443 [W]
Trial 73/203 finished    maximum torque is 2.129 [Nm]    maximum power is 1.718 [W]
Trial 74/203 finished    maximum torque is 1.974 [Nm]    maximum power is 1.706 [W]
Trial 75/203 finished    maximum torque is 1.978 [Nm]    maximum power is 2.523 [W]
Trial 76/203 finished    maximum torque is 2.469 [Nm]    maximum power is 1.152 [W]
Trial 77/203 finished    maximum torque is 2.401 [Nm]    maximum power is 1.364 [W]
Trial 78/203 finished    maximum torque is 1.861 [Nm]    maximum power is 1.752 [W]
Trial 79/203 finished    maximum torque is 1.875 [Nm]    maximum power is 1.706 [W]
Trial 80/203 finished    maximum torque is 2.243 [Nm]    maximum power is 2.217 [W]
Trial 81/203 finished    maximum torque is 2.522 [Nm]    maximum power is 4.007 [W]
Trial 82/203 finished    maximum torque is 2.391 [Nm]    maximum power is 2.629 [W]
Trial 83/203 finished    maximum torque is 2.542 [Nm]    maximum power is 4.609 [W]
Trial 84/203 finished    maximum torque is 2.741 [Nm]    maximum power is 1.506 [W]
Trial 85/203 finished    maximum torque is 2.333 [Nm]    maximum power is 1.294 [W]
Trial 86/203 finished    maximum torque is 2.170 [Nm]    maximum power is 2.756 [W]
Trial 87/203 finished    maximum torque is 2.589 [Nm]    maximum power is 4.681 [W]
Trial 88/203 finished    maximum torque is 2.358 [Nm]    maximum power is 2.869 [W]
Trial 89/203 finished    maximum torque is 1.987 [Nm]    maximum power is 1.439 [W]
Trial 90/203 finished    maximum torque is 2.278 [Nm]    maximum power is 2.368 [W]
Trial 91/203 finished    maximum torque is 2.525 [Nm]    maximum power is 2.188 [W]
Trial 92/203 finished    maximum torque is 2.173 [Nm]    maximum power is 1.750 [W]
Trial 93/203 finished    maximum torque is 1.912 [Nm]    maximum power is 0.842 [W]
Trial 94/203 finished    maximum torque is 2.015 [Nm]    maximum power is 2.232 [W]
Trial 95/203 finished    maximum torque is 2.040 [Nm]    maximum power is 3.430 [W]
Trial 96/203 finished    maximum torque is 1.774 [Nm]    maximum power is 1.637 [W]
Trial 97/203 finished    maximum torque is 1.556 [Nm]    maximum power is 1.528 [W]
Trial 98/203 finished    maximum torque is 2.215 [Nm]    maximum power is 3.812 [W]
Trial 99/203 finished    maximum torque is 2.002 [Nm]    maximum power is 3.090 [W]
Trial 100/203 finished   maximum torque is 2.126 [Nm]    maximum power is 2.131 [W]
Trial 101/203 finished   maximum torque is 2.360 [Nm]    maximum power is 3.469 [W]
Trial 102/203 finished   maximum torque is 1.947 [Nm]    maximum power is 1.375 [W]
Trial 103/203 finished   maximum torque is 1.990 [Nm]    maximum power is 2.304 [W]
Trial 104/203 finished   maximum torque is 2.212 [Nm]    maximum power is 3.225 [W]
Trial 105/203 finished   maximum torque is 3.599 [Nm]    maximum power is 4.341 [W]
```

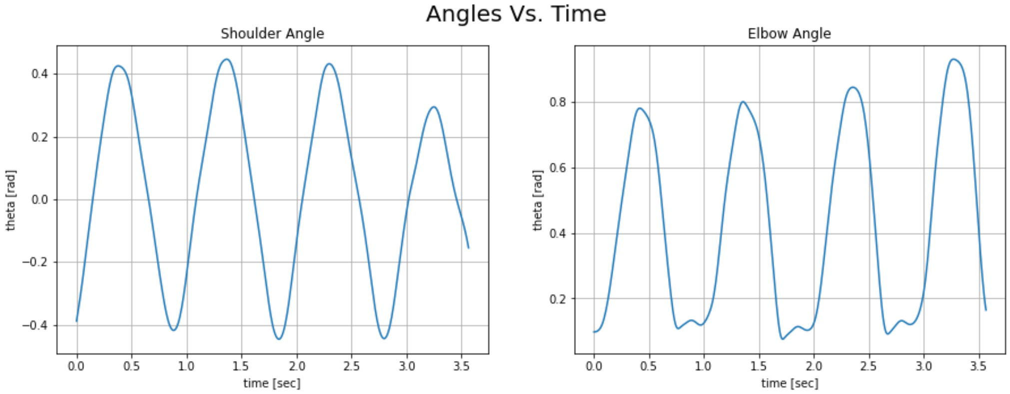
```
Trial 106/203 finished   maximum torque is 2.234 [Nm]    maximum power is 2.158 [W]
Trial 107/203 finished   maximum torque is 2.002 [Nm]    maximum power is 1.892 [W]
Trial 108/203 finished   maximum torque is 2.106 [Nm]    maximum power is 2.203 [W]
Trial 109/203 finished   maximum torque is 1.893 [Nm]    maximum power is 0.919 [W]
Trial 110/203 finished   maximum torque is 2.191 [Nm]    maximum power is 1.887 [W]
Trial 111/203 finished   maximum torque is 2.268 [Nm]    maximum power is 1.469 [W]
Trial 112/203 finished   maximum torque is 1.966 [Nm]    maximum power is 1.381 [W]
Trial 113/203 finished   maximum torque is 2.072 [Nm]    maximum power is 2.131 [W]
Trial 114/203 finished   maximum torque is 2.578 [Nm]    maximum power is 1.739 [W]
Trial 115/203 finished   maximum torque is 1.988 [Nm]    maximum power is 1.013 [W]
Trial 116/203 finished   maximum torque is 2.339 [Nm]    maximum power is 1.967 [W]
Trial 117/203 finished   maximum torque is 2.146 [Nm]    maximum power is 2.601 [W]
Trial 118/203 finished   maximum torque is 2.098 [Nm]    maximum power is 1.906 [W]
Trial 119/203 finished   maximum torque is 1.820 [Nm]    maximum power is 1.798 [W]
Trial 120/203 finished   maximum torque is 3.030 [Nm]    maximum power is 2.246 [W]
Trial 121/203 finished   maximum torque is 2.473 [Nm]    maximum power is 2.025 [W]
Trial 122/203 finished   maximum torque is 2.410 [Nm]    maximum power is 3.457 [W]
Trial 123/203 finished   maximum torque is 2.514 [Nm]    maximum power is 5.727 [W]
Trial 124/203 finished   maximum torque is 2.730 [Nm]    maximum power is 2.621 [W]
Trial 125/203 finished   maximum torque is 4.763 [Nm]    maximum power is 7.720 [W]
Trial 126/203 finished   maximum torque is 2.091 [Nm]    maximum power is 2.594 [W]
Trial 127/203 finished   maximum torque is 2.475 [Nm]    maximum power is 3.097 [W]
Trial 128/203 finished   maximum torque is 2.455 [Nm]    maximum power is 1.926 [W]
Trial 129/203 finished   maximum torque is 1.996 [Nm]    maximum power is 1.769 [W]
Trial 130/203 finished   maximum torque is 2.532 [Nm]    maximum power is 1.744 [W]
Trial 131/203 finished   maximum torque is 2.613 [Nm]    maximum power is 2.392 [W]
Trial 132/203 finished   maximum torque is 2.044 [Nm]    maximum power is 2.620 [W]
Trial 133/203 finished   maximum torque is 1.554 [Nm]    maximum power is 1.610 [W]
Trial 134/203 finished   maximum torque is 2.198 [Nm]    maximum power is 2.649 [W]
Trial 135/203 finished   maximum torque is 1.966 [Nm]    maximum power is 1.637 [W]
Trial 136/203 finished   maximum torque is 2.294 [Nm]    maximum power is 2.665 [W]
Trial 137/203 finished   maximum torque is 2.576 [Nm]    maximum power is 4.996 [W]
Trial 138/203 finished   maximum torque is 2.785 [Nm]    maximum power is 2.246 [W]
Trial 139/203 finished   maximum torque is 3.341 [Nm]    maximum power is 2.181 [W]
Trial 140/203 finished   maximum torque is 1.717 [Nm]    maximum power is 1.455 [W]
Trial 141/203 finished   maximum torque is 1.652 [Nm]    maximum power is 1.183 [W]
Trial 142/203 finished   maximum torque is 2.295 [Nm]    maximum power is 2.777 [W]
Trial 143/203 finished   maximum torque is 2.086 [Nm]    maximum power is 3.021 [W]
Trial 144/203 finished   maximum torque is 2.387 [Nm]    maximum power is 0.947 [W]
Trial 145/203 finished   maximum torque is 2.380 [Nm]    maximum power is 1.233 [W]
Trial 146/203 finished   maximum torque is 1.937 [Nm]    maximum power is 1.207 [W]
Trial 147/203 finished   maximum torque is 3.033 [Nm]    maximum power is 3.924 [W]
Trial 148/203 finished   maximum torque is 1.970 [Nm]    maximum power is 1.947 [W]
Trial 149/203 finished   maximum torque is 1.832 [Nm]    maximum power is 1.943 [W]
Trial 150/203 finished   maximum torque is 1.823 [Nm]    maximum power is 1.170 [W]
Trial 151/203 finished   maximum torque is 2.183 [Nm]    maximum power is 1.349 [W]
Trial 152/203 finished   maximum torque is 2.172 [Nm]    maximum power is 2.224 [W]
Trial 153/203 finished   maximum torque is 1.603 [Nm]    maximum power is 1.979 [W]
Trial 154/203 finished   maximum torque is 2.040 [Nm]    maximum power is 2.128 [W]
Trial 155/203 finished   maximum torque is 1.822 [Nm]    maximum power is 2.508 [W]
Trial 156/203 finished   maximum torque is 3.221 [Nm]    maximum power is 1.860 [W]
Trial 157/203 finished   maximum torque is 3.713 [Nm]    maximum power is 3.304 [W]
Trial 158/203 finished   maximum torque is 2.351 [Nm]    maximum power is 1.031 [W]
Trial 159/203 finished   maximum torque is 2.428 [Nm]    maximum power is 1.585 [W]
Trial 160/203 finished   maximum torque is 2.144 [Nm]    maximum power is 1.369 [W]
Trial 161/203 finished   maximum torque is 2.231 [Nm]    maximum power is 1.931 [W]
Trial 162/203 finished   maximum torque is 2.292 [Nm]    maximum power is 2.646 [W]
Trial 163/203 finished   maximum torque is 2.537 [Nm]    maximum power is 3.914 [W]
Trial 164/203 finished   maximum torque is 2.434 [Nm]    maximum power is 1.870 [W]
Trial 165/203 finished   maximum torque is 2.354 [Nm]    maximum power is 4.254 [W]
Trial 166/203 finished   maximum torque is 2.030 [Nm]    maximum power is 1.526 [W]
Trial 167/203 finished   maximum torque is 2.073 [Nm]    maximum power is 2.896 [W]
Trial 168/203 finished   maximum torque is 2.131 [Nm]    maximum power is 3.605 [W]
Trial 169/203 finished   maximum torque is 1.901 [Nm]    maximum power is 2.855 [W]
Trial 170/203 finished   maximum torque is 2.360 [Nm]    maximum power is 3.497 [W]
Trial 171/203 finished   maximum torque is 2.789 [Nm]    maximum power is 5.233 [W]
Trial 172/203 finished   maximum torque is 1.869 [Nm]    maximum power is 1.753 [W]
Trial 173/203 finished   maximum torque is 1.669 [Nm]    maximum power is 1.630 [W]
Trial 174/203 finished   maximum torque is 2.013 [Nm]    maximum power is 1.113 [W]
Trial 175/203 finished   maximum torque is 2.360 [Nm]    maximum power is 1.998 [W]
Trial 176/203 finished   maximum torque is 2.137 [Nm]    maximum power is 2.338 [W]
Trial 177/203 finished   maximum torque is 2.082 [Nm]    maximum power is 1.151 [W]
Trial 178/203 finished   maximum torque is 2.223 [Nm]    maximum power is 1.230 [W]
Trial 179/203 finished   maximum torque is 2.251 [Nm]    maximum power is 1.591 [W]
Trial 180/203 finished   maximum torque is 2.075 [Nm]    maximum power is 2.092 [W]
Trial 181/203 finished   maximum torque is 1.920 [Nm]    maximum power is 2.020 [W]
Trial 182/203 finished   maximum torque is 2.279 [Nm]    maximum power is 2.805 [W]
Trial 183/203 finished   maximum torque is 2.452 [Nm]    maximum power is 2.878 [W]
Trial 184/203 finished   maximum torque is 2.433 [Nm]    maximum power is 2.116 [W]
Trial 185/203 finished   maximum torque is 2.447 [Nm]    maximum power is 2.890 [W]
Trial 186/203 finished   maximum torque is 2.123 [Nm]    maximum power is 1.694 [W]
Trial 187/203 finished   maximum torque is 1.985 [Nm]    maximum power is 2.341 [W]
Trial 188/203 finished   maximum torque is 1.871 [Nm]    maximum power is 2.371 [W]
Trial 189/203 finished   maximum torque is 1.941 [Nm]    maximum power is 2.554 [W]
Trial 190/203 finished   maximum torque is 2.553 [Nm]    maximum power is 2.004 [W]
Trial 191/203 finished   maximum torque is 1.659 [Nm]    maximum power is 2.123 [W]
Trial 192/203 finished   maximum torque is 3.164 [Nm]    maximum power is 2.245 [W]
Trial 193/203 finished   maximum torque is 3.586 [Nm]    maximum power is 1.153 [W]
Trial 194/203 finished   maximum torque is 2.758 [Nm]    maximum power is 1.518 [W]
Trial 195/203 finished   maximum torque is 1.986 [Nm]    maximum power is 1.950 [W]
Trial 196/203 finished   maximum torque is 1.611 [Nm]    maximum power is 2.922 [W]
Trial 197/203 finished   maximum torque is 1.658 [Nm]    maximum power is 1.788 [W]
Trial 198/203 finished   maximum torque is 1.963 [Nm]    maximum power is 3.649 [W]
Trial 199/203 finished   maximum torque is 1.663 [Nm]    maximum power is 3.701 [W]
Trial 200/203 finished   maximum torque is 1.867 [Nm]    maximum power is 1.517 [W]
Trial 201/203 finished   maximum torque is 2.245 [Nm]    maximum power is 2.314 [W]
Trial 202/203 finished   maximum torque is 1.546 [Nm]    maximum power is 1.681 [W]
Trial 203/203 finished   maximum torque is 1.603 [Nm]    maximum power is 1.502 [W]
```

In [9]:

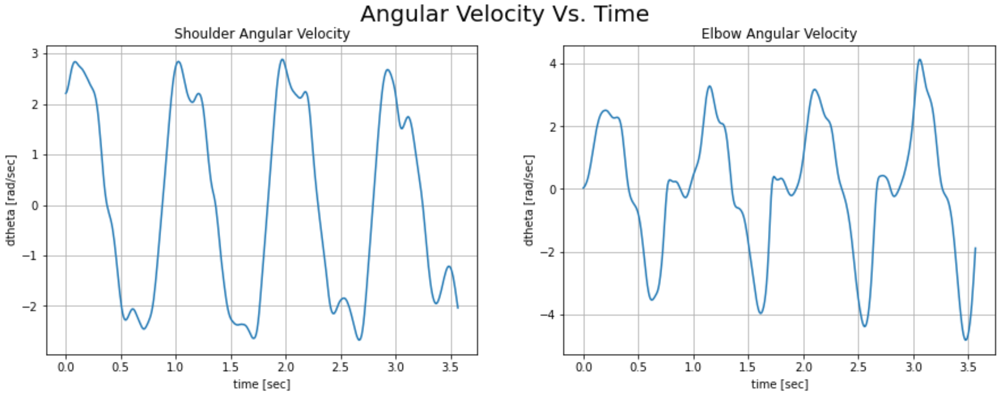
```
index = 69
t_list = time_list[index]
theta1_list = Shl_Flex_Ang_list[index]
theta2_list = Elbow_Ang_list[index]
dtheta1_list = Shl_Flex_Vel_list[index]
dtheta2_list = Elbow_Vel_list[index]
ddtheta1_list = Shl_Flex_Acc_list[index]
ddtheta2_list = Elbow_Acc_list[index]
tau1_list = Shl_Flex_tau_list[index]
tau2_list = Elbow_tau_list[index]
power1_list = Shl_Flex_power_list[index]
power2_list = Elbow_power_list[index]

# Compute the trajectory of the arm's motion
N = int((max(t_list) - min(t_list))/(1/frame_frequency))
tvec = np.linspace(min(t_list), max(t_list), N)
traj = np.zeros((6, N))
for i in range(N):
    traj[0, i] = theta1_list[i]
```

```python
    traj[1, i] = theta2_list[i]
    traj[2, i] = dtheta1_list[i]
    traj[3, i] = dtheta2_list[i]
    traj[4, i] = ddtheta1_list[i]
    traj[5, i] = ddtheta2_list[i]

# Calculate the length difference between the time list and the trajectory lists
diff = (len(t_list) - len(traj[0]))

# Plot the trajectory lists (angles, velocities, accelerations, torques, and power)
plt.figure(figsize=(15,5))
plt.suptitle('Angles Vs. Time', fontsize=20)
plt.subplot(121)
plt.plot(t_list[:-diff], traj[0])
plt.ylabel('theta [rad]')
plt.xlabel('time [sec]')
plt.grid()
plt.title('Shoulder Angle')

plt.subplot(122)
plt.plot(t_list[:-diff], traj[1])
plt.ylabel('theta [rad]')
plt.xlabel('time [sec]')
plt.grid()
plt.title('Elbow Angle')
plt.show()

plt.figure(figsize=(15,5))
plt.suptitle('Angular Velocity Vs. Time', fontsize=20)
plt.subplot(121)
plt.plot(t_list[:-diff], traj[2])
plt.ylabel('dtheta [rad/sec]')
plt.xlabel('time [sec]')
plt.grid()
plt.title('Shoulder Angular Velocity')

plt.subplot(122)
plt.plot(t_list[:-diff], traj[3])
plt.ylabel('dtheta [rad/sec]')
plt.xlabel('time [sec]')
plt.grid()
plt.title('Elbow Angular Velocity')
plt.show()

plt.figure(figsize=(15,5))
plt.suptitle('Angular Acceleration Vs. Time', fontsize=20)
plt.subplot(121)
plt.plot(t_list[:-diff], traj[4])
plt.ylabel('ddtheta [rad/sec^2]')
plt.xlabel('time [sec]')
plt.grid()
plt.title('Shoulder Angular Acceleration')

plt.subplot(122)
plt.plot(t_list[:-diff], traj[5])
plt.ylabel('ddtheta [rad/sec^2]')
plt.xlabel('time [sec]')
plt.grid()
plt.title('Elbow Angular Acceleration')
plt.show()

plt.figure(figsize=(15,5))
plt.suptitle('Torque Vs. Time', fontsize=20)
plt.subplot(121)
plt.plot(t_list, tau1_list)
plt.ylabel('tau [Nm]')
plt.xlabel('time [sec]')
plt.grid()
plt.title('Shoulder Torque')

plt.subplot(122)
plt.plot(t_list, tau2_list)
plt.ylabel('tau [Nm]')
plt.xlabel('time [sec]')
plt.grid()
plt.title('Elbow Torque')
plt.show()

plt.figure(figsize=(15,5))
plt.suptitle('Power Vs. Time', fontsize=20)
plt.subplot(121)
plt.plot(t_list, power1_list)
plt.ylabel('power [W]')
plt.xlabel('time [sec]')
plt.grid()
plt.title('Shoulder Power')

plt.subplot(122)
plt.plot(t_list, power2_list)
plt.ylabel('power [W]')
plt.xlabel('time [sec]')
plt.grid()
plt.title('Elbow Power')
plt.show()

plt.figure(figsize=(15,5))
plt.suptitle('Speed Vs. Torque', fontsize=20)
plt.subplot(121)
plt.plot(tau1_list[:-diff], traj[2])
plt.ylabel('dtheta [rad/sec]')
plt.xlabel('tau [Nm]')
plt.grid()
plt.title('Shoulder Speed-Torque')

plt.subplot(122)
plt.plot(tau2_list[:-diff], traj[3])
plt.ylabel('dtheta [rad/sec]')
plt.xlabel('tau [Nm]')
plt.grid()
plt.title('Elbow Speed-Torque')
plt.show()
```

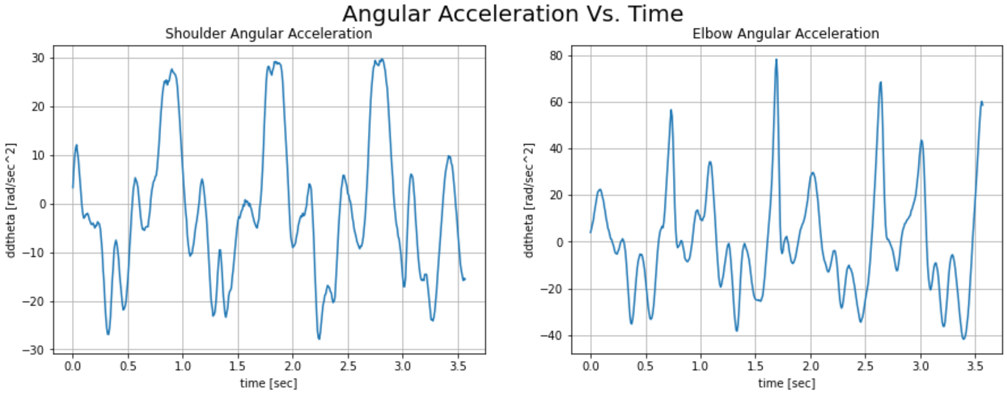
Angles Vs. Time



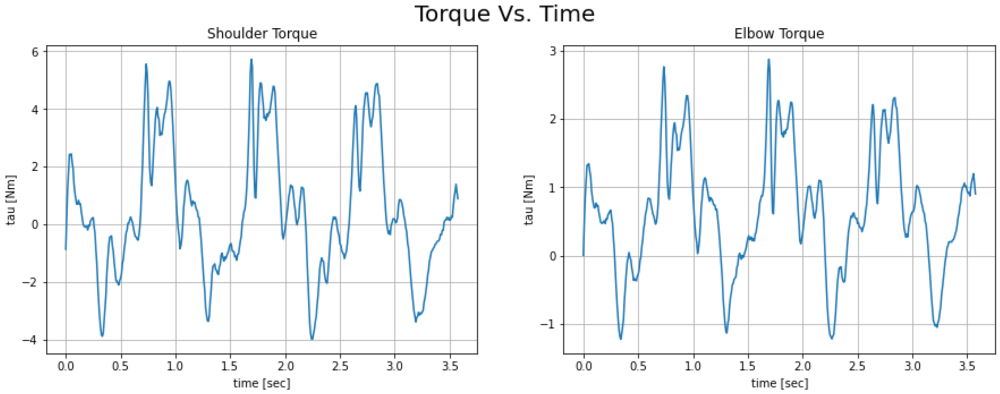
Angular Velocity Vs. Time



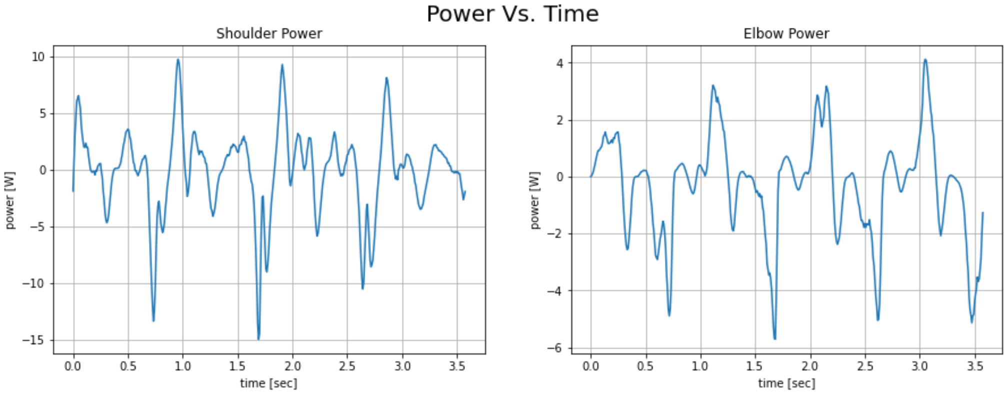
Angular Acceleration Vs. Time

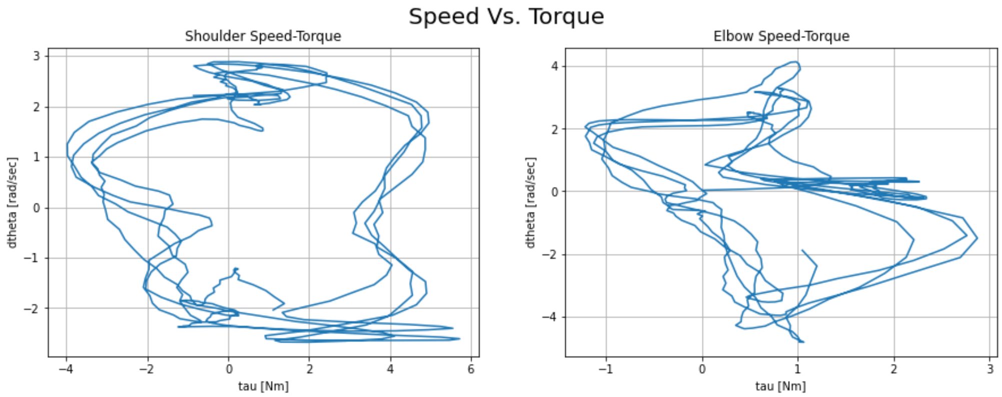


Torque Vs. Time



Power Vs. Time



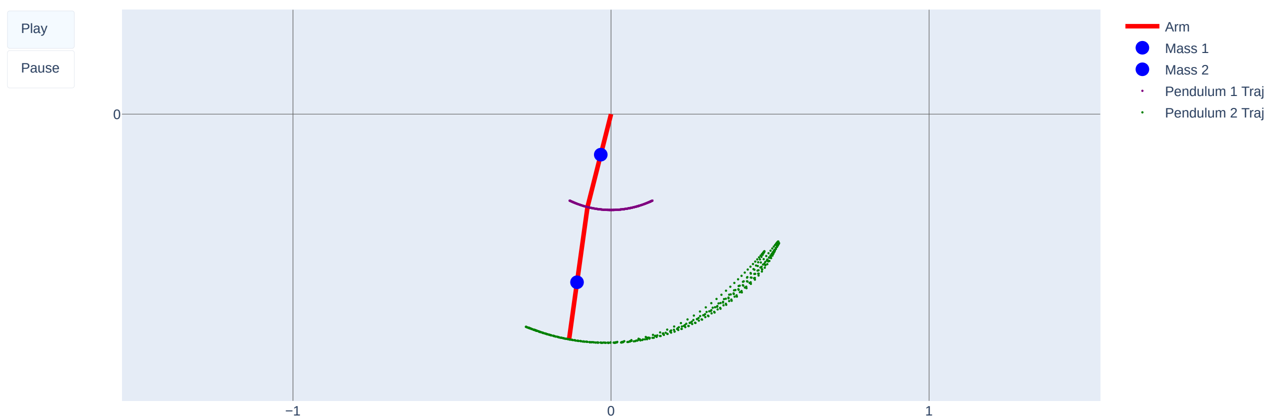


Animating the simulation:

In [10]:
```
def animate_double_pend(traj, L1, L2, L1_COM, L2_COM, T=5):
    """
    Function to generate web-based animation of double-pendulum system

    Parameters:
        traj:        trajectory of theta1 and theta2
        L1:          length of the upper arm
        L2:          length of the lower arm
        L1_COM:      length of the center of mass of the upper arm from the shoulder
        L2_COM:      length of the center of mass of the lower arm from the elbow
        T:           length/seconds of animation duration

    Returns: None
    """

    # Browser configuration
    def configure_plotly_browser_state():
        import IPython
        display(IPython.core.display.HTML('''
            <script src="/static/components/requirejs/require.js"></script>
            <script>
              requirejs.config({
                paths: {
                  base: '/static/base',
                  plotly: 'https://cdn.plot.ly/plotly-1.5.1.min.js?noext',
                },
              });
            </script>
            '''))
    configure_plotly_browser_state()
    init_notebook_mode(connected=False)

    # Getting data from pendulum angle trajectories
    xx1 = L1 * np.sin(traj[0])
    yy1 = -L1 * np.cos(traj[0])
    xx1_COM = L1_COM * np.sin(traj[0])
    yy1_COM = -L1_COM * np.cos(traj[0])
    xx2 = xx1 + L2 * np.sin(traj[0] + traj[1])
    yy2 = yy1 - L2 * np.cos(traj[0] + traj[1])
    xx2_COM = xx1 + L2_COM * np.sin(traj[0] + traj[1])
    yy2_COM = yy1 - L2_COM * np.cos(traj[0] + traj[1])
    N = len(traj[0])

    # Using these to specify axis limits
    xm = np.min(xx1)
    xM = np.max(xx1)
    ym = np.min(yy1) - 0.6
    yM = np.max(yy1) + 0.6

    # Defining data dictionary
    data = [dict(x=xx1, y=yy1,
                 mode='lines', name='Arm',
                 line=dict(width=2, color='blue')
                ),
            dict(x=xx1_COM, y=yy1_COM,
                 mode='lines', name='Mass 1',
                 line=dict(width=2, color='purple')
                ),
            dict(x=xx2_COM, y=yy2_COM,
                 mode='lines', name='Mass 2',
                 line=dict(width=2, color='green')
                ),
            dict(x=xx1, y=yy1,
                 mode='markers', name='Pendulum 1 Traj',
                 marker=dict(color="purple", size=2)
                ),
            dict(x=xx2, y=yy2,
                 mode='markers', name='Pendulum 2 Traj',
                 marker=dict(color="green", size=2)
                )
          ]

    # Preparing simulation layout
    layout = dict(xaxis=dict(range=[xm, xM], autorange=False, zeroline=False,dtick=1),
                  yaxis=dict(range=[ym, yM], autorange=False, zeroline=False,scaleanchor = "x",dtick=1),
                  title='Arm Modeled as a Double Pendulum Simulation',
                  hovermode='closest',
                  updatemenus= [{'type': 'buttons',
                                 'buttons': [{'label': 'Play','method': 'animate',
                                              'args': [None, {'frame': {'duration': T, 'redraw': False}}]},
                                             {'args': [[None], {'frame': {'duration': T, 'redraw': False}, 'mode': 'immediate',
                                              'transition': {'duration': 0}}],'label': 'Pause','method': 'animate'}
                                            ]
                                }]
                 )

    # Defining the frames of the simulation
```

```
    frames = [dict(data=[dict(x=[0,xx1[k],xx2[k]],
                              y=[0,yy1[k],yy2[k]],
                              mode='lines',
                              line=dict(color='red', width=4)),
                         go.Scatter(
                              x=[xx1_COM[k]],
                              y=[yy1_COM[k]],
                              mode="markers",
                              marker=dict(color="blue", size=12)),
                         go.Scatter(
                              x=[xx2_COM[k]],
                              y=[yy2_COM[k]],
                              mode="markers",
                              marker=dict(color="blue", size=12)),
                        ]) for k in range(N)]

    # Putting it all together and plotting
    figure = dict(data=data, layout=layout, frames=frames)
    iplot(figure)

# Animate the system
animate_double_pend(traj, L1=L_upper_arm_dict[participants_list[index]], L2=L_lower_arm, L1_COM=L_upper_arm_COM_dict[participants_list[index]], L2_COM=L_lower_arm_COM, T=
```

Arm Modeled as a Double Pendulum Simulation



Calculation summary:

In [11]:
```
# print(f"Shoulder max angular velocity:\t{format(max(dtheta1_list), '.3f')} [rad/sec]\t\t Shoulder average angular velocity:\t{format(sum(dtheta1_list) / float(len(dthet
# print(f"Shoulder max angular velocity:\t{format(max(dtheta1_list)*60/(2*pi), '.3f')} [rpm]\t\t Shoulder average angular velocity:\t{format((sum(dtheta1_list) / float(le
# print(f"Elbow max angular velocity:\t{format(max(dtheta2_list), '.3f')} [rad/sec]\t\t Elbow average angular velocity:\t{format(sum(dtheta2_list) / float(len(dtheta2_lis
# print(f"Elbow max angular velocity:\t{format(max(dtheta2_list)*60/(2*pi), '.3f')} [rpm]\t\t Elbow average angular velocity:\t{format((sum(dtheta2_list) / float(len(dthe
# print(f"Shoulder max torque:\t\t{format(max(tau1_list), '.3f')} [Nm]\t\t Shoulder average torque:\t\t{format(sum(tau1_list) / float(len(tau1_list)), '.3f')} [Nm]")
# print(f"Elbow max torque:\t\t{format(max(tau2_list), '.3f')} [Nm]\t\t Elbow average torque:\t\t\t{format(sum(tau2_list) / float(len(tau2_list)), '.3f')} [Nm]\n")
# print(f"Shoulder max power:\t\t{format(max(power1_list), '.3f')} [W]\t\t Shoulder average power:\t\t{format(sum(power1_list) / float(len(power1_list)), '.3f')} [W]")
# print(f"Elbow max power:\t\t{format(max(power2_list), '.3f')} [W]\t\t Elbow average power:\t\t\t{format(sum(power2_list) / float(len(power2_list)), '.3f')} [W]\n")

max_Elbow_tau, max_Elbow_power, max_Elbow_Vel = 0, 0, 0
max_Elbow_tau_index, max_Elbow_power_index, max_Elbow_Vel_index = 0, 0, 0

for i in range(len(Elbow_tau_list)):
    if max_Elbow_Vel < max(Elbow_Vel_list[i]):
        max_Elbow_Vel = max(Elbow_Vel_list[i])
        max_Elbow_Vel_index = i

    if max_Elbow_tau < max(Elbow_tau_list[i]):
        max_Elbow_tau = max(Elbow_tau_list[i])
        max_Elbow_tau_index = i

    if max_Elbow_power < max(Elbow_power_list[i]):
        max_Elbow_power = max(Elbow_power_list[i])
        max_Elbow_power_index = i

print(f"maximum elbow angular velocity is {format(max_Elbow_Vel, '.3f')} [rad/sec] ({format(max_Elbow_Vel*60/(2*pi), '.3f')} [rpm]), in trial {max_Elbow_Vel_index}")
print(f"maximum elbow torque is {format(max_Elbow_tau, '.3f')} [Nm], in trial {max_Elbow_tau_index}")
print(f"maximum elbow power is {format(max_Elbow_power, '.3f')} [W], in trial {max_Elbow_power_index}")
```

```
maximum elbow angular velocity is 4.143 [rad/sec] (39.560 [rpm]), in trial 69
maximum elbow torque is 4.763 [Nm], in trial 125
maximum elbow power is 7.720 [W], in trial 125
```

Motor selection

In [ ]:
```
```