COST-EFFECTIVE OPTIMIZATION OF AN UPPER LIMB REHABILITATION MECHANISM

Dulat Aimukhanov, B. Eng and Technology

Submitted in fulfilment of the requirements

for the degree of Masters of Science

in Mechanical Engineering

School of Engineering

Department of Mechanical & Aerospace Engineering

Nazarbayev University

53 Kabanbay Batyr Avenue,

Astana, Kazakhstan, 010000

Supervisor: Konstantinos Kostas

Co-supervisor: Evagoras Xydas

December 2018

DECLARATION

I hereby, declare that this manuscript, entitled “Cost-effective optimization of an upper limb rehabilitation mechanism”, is the result of my own work except for quotations and citations which have been duly acknowledged.

I also declare that, to the best of my knowledge and belief, it has not been previously or concurrently submitted, in whole or in part, for any other degree or diploma at Nazarbayev University or any other national or international institution.

Name: Dulat Aimukhanov

Date: 13 December 2018

Abstract

In recent years, a vast variety of mechanisms for upper limb rehabilitation have been designed by researchers. The majority of these designs are based on multi degree of freedom and open kinematic chain assemblies. The application of such mechanisms can offer significant aid in successful treatment. Their disadvantages, however, include complexity and costliness. As an alternative to these, other types of mechanisms, such as four and six bar linkages, can be employed in rehabilitation of patients with arm-motion disabilities. These alternative mechanisms are simpler and cheaper, but still have the capacity to offer complex kinematic characteristics.

This thesis is based on the analysis of Hoeken's four bar linkage for straight-line and curved-line motion. In the design of our mechanism, neurophysiological models such as the Minimum Jerk Model, Fitts's Law and the Just Noticeable Difference concept are taken into account. The scope of this work covers the kinematic and kinetostatic analysis, inertial optimization and use of passive control elements.

The objective of this work is to decrease the cost of the mechanism by minimizing the required input torque for generating a Minimum Jerk Model compliant motion. The position of each link's center of gravity and the addition of linear translational springs, as passive control elements, are employed in mechanism's design optimization.

Acknowledgements

Firstly, I would like to express the deepest gratitude to my supervisor Konstantinos Kostas for his helping and mentoring. Secondly, I would like to express my appreciation to my co-supervisor Evagoras Xydas for his responsiveness and aiding in implementing this thesis. This work has completed thanks to their advices and discussion hours that they have devoted.

Finally, I would like to say thanks to my parents and friends, who have been supporting me all this time.

Table of Contents

List of Abbreviations and Symbols

α	Angular acceleration
θ	Angular position
ω	Angular velocity
CG	Center of Gravity
DOF	Degree of Freedom
a	Linear acceleration
V	Linear velocity
MJM	Minimum Jerk Model
I_G	Moment of Inertia
R	Position vector
k	Stiffness of spring
T	Torque
t_f	Total time of movement

List of Tables

List of Figures

Chapter 1 – Introduction

1.1 Upper limb rehabilitation

According to the World Health Organization stroke is the third leading cause of disability and 15 million people suffer from stroke worldwide each year. Generally, about 80% of strokes occur in low and middle income countries [1]. Many people surviving a stroke have upper and lower limb disabilities. The recovery of post-stroke patients is more effective, when patient gets regular treatment including practice of upper and lower limb motion [2]. These types of therapy are also applied in the context of rehabilitation for people with cerebral palsy, multiple sclerosis, traumatic brain injury, spinal cord injury and other neuro-disabilities. Patients with such disabilities need to undergo a long-term rehabilitation process to strengthen their neural system. The rehabilitation's aim is to enable restoration of lost abilities, which ultimately allow patients to carry out daily life tasks and activities. Recovery is a laborious process and requires individual interaction with therapist. Therefore, any lack of specialized therapists and nurses helping such patients can lead to serious problems. Further to this, the need for such intensive care increases significantly expenses for medical treatment and rehabilitation. As a consequence, the development of mechanisms for rehabilitation of upper and lower limbs has seen a significant increase in recent decades.

In the last decades [3], robotic systems for upper limb rehabilitation are supplementing established therapeutical procedures, as they can ensure high-intensity training and accelerate rehabilitation while keeping the treatment cost relatively low. Several mechanisms have been developed aiming upper limb motion recovery. These are either Exoskeleton or Operational machines. Exoskeleton machines comprise wearable mechanisms that act in parallel to patient's movements. They are used as means to reinforce and/or restore human motion performance. Their structure is generally very complicated mainly due to numerous sensors and actuators. Operational machines, or planar robotic systems, are considerably simpler and guide the users via their end-effector. The motion of the end-effector is pre-programmed in Cartesian space and the mechanism assists the patient in following the trajectory of the end-effector. Operational machines are further divided into two subtypes. The first type includes generally expensive, low friction and high back-driveability mechanisms that allow measurement of human-arm resistance [3]. These are mainly based on open kinematic chains with multiple degrees of freedom (DOF), which guarantee a wide range of motions, but also have shortcomings. Rehabilitation systems with multiple DOFs are generally expensive as multiple actuators and sensors are required, which can also lead to very large installations. Their complex electronics require qualified personnel to maintain properly. Moreover, sometimes rehabilitation procedures require simple trajectories for upper limb, so it makes complex mechanisms overdesigned for particular exercises. Further to the above, the large number of

DOFs, can lead to unwanted oscillations and unsteadiness. The second subtype of operational machines are generally simple mechanisms featuring active compensation of friction and low cost. As mentioned in the beginning of this chapter, upper limb rehabilitation systems are mainly required in developing countries and therefore, simple and inexpensive solutions become a prerequisite.

One of the simplest mechanisms that can be employed is the four bar linkage with one DOF. A four bar linkage consists of four rigid rods connected by pin joints and allows the convenient generation of a rather large number of motions, including straight-line and curved motion trajectories. Motor power requirements and rehabilitation efficiency are key elements when employing such mechanisms as rehabilitation aids.

1.2 Aim and Objectives

One of the major aims in this work is the design of a mechanism with low motor-power requirements, which will obviously reduce the overall cost. Specifically, we target the reduction of the required input torque on the driving link and this can be accomplished by carefully selecting the driving link, optimizing the position of each link's center of gravity and employing linear translational springs to assist and/or replace the motor itself.

Obviously, at the same time we need to be able to maintain the correct trajectory and/or velocity and acceleration profiles and therefore a complete kinematic and kinetostatic analysis is required.

1.3 Methodologies and techniques

The synthesis of the four bar linkage mechanism investigated in this work is performed in the basis of neurophysiological models such as the Minimum Jerk Model (MJM), Fitts's law and the Just Noticeable difference (detailed explanations of models are given in Chapter 2.3). Straight-line and curved motion are examined in terms of MJM. Initial, intermediate and final positions of coupler point are defined by using GIM software. The kinematic analysis of the mechanism performed by using vector loop representation and classical Freudenstein approach (detailed explanation of approach is given in Chapter 3.2). Force analysis or inverse dynamic problem are solved by using Newton's laws of motion in order to define external forces and torque that required to provide desired motion of the mechanism. Inertial optimization of the mechanism was realized by finding the optimal positions of Center of Gravities (CG) of each link by regulating the length of position vectors. This procedure is implemented with help of fmincon function tool in MATLAB. The position, length and stiffness of springs were obtained by using genetic algorithm method in MATLAB.

1.4 Thesis structure

The thesis comprises the following chapters: a) literature review on existing approaches of four bar linkage application to upper limb rehabilitation; b) the design and analysis of the mechanism, which includes kinematic analysis; kinetostatic analysis; inertial optimization and inclusion of springs as passive control elements. Finally, results and concluding observations complete this work.

Chapter 2 – Literature review

2.1 Application of robotic systems in upper limb rehabilitation

Planar robotic systems are widely used in upper limb rehabilitation. They can be employed to perform straightforward and teachable recovery exercises. Availability of graphical user interface (GUI) in these systems make them more understandable for the disabled patients. In addition, the usage of Cartesian based coordinate system is convenient for evaluate patient's progress in recovery. The main shortcoming of such mechanisms is the limited training of patient's shoulder when compared to Exoskeleton machines. This disadvantage can be partially compensated by inclining the workspace of mechanism. Planar robotic systems can be divided into three main types: Cartesian robots, Selective Compliance Articulated Robot Arm (SCARA) robots and cable driven robots.

Most current robotic rehabilitation systems have either two or three DOFs. An early type of Cartesian robot used in rehabilitation was named "Mechatronic system for motor recovery after stroke (MEMOS)" [4]. MEMOS, despite its simple structure, has been demonstrated to decrease the level of disability in an effective way. Zollo et al. [5] proposed design criteria for modifying Campus Bio-Medico-Motus robot. The distinctive feature of this robot is low and isotropic inertia. Zollo et al. presented the results of acceleration and inertia characteristics taking at the end-effector.

The most well known SCARA robot is the MIT/Manus [6, 7]. The MIT/Manus robot has two DOFs, implemented via a direct-drive five bar linkage. In this case, recovery exercises are presented via a LCD panel that is located in the front of the patient. It mainly targets shoulder and elbow rehabilitation. Commercial versions of this robot are InMotion and Braccio di Ferro [8]. Their differences to MIT/Manus include the capability to incline the planar workspace, motor power and human machine interface. Li et al. [9] have developed a SCARA rehabilitation system based on the surface electromyographic (sEMG) signal. This robot uses the sEMG sensor to evaluate the impairment of motion function and then sets optimal exercise trajectories.

The third type of planar robotic systems generates a force at the end-effector via a series of stretched cables activated by an appropriate motorized mechanism. Examples of such robots include NeRebot [10], Maribot [11], FeRiBa3 [12], Sophia-3 and Sophia-4 [3]. An extended list of robotic systems for upper limb rehabilitation is presented in review done by BioMed Central workers [13].

All above mentioned robots are complex systems with open kinematic chains and multi-DOFs. These robots are expensive and cannot be easily customized. Therefore, small rehabilitation centers in developing countries cannot afford them. One possible alternative is to use less expensive, but still useful mechanisms. There are several research works focusing on simpler mechanisms with a closed kinematic chain and one DOF. Xydas [14] investigated Chebyshev's mechanism for generating straight-line motion having a natural velocity profile.

This work includes the kinematic and kinetostatic analysis of the mechanism, which is used to calculate the required torque for generating the desired position and velocity profiles. The presented results indicate that Chebyshev's mechanism can be used as a rehabilitation system and impaired patients may improve their coordination skills with this mechanism.

Xydas et al. in [15] described a method that uses passive control elements, such as linear springs, to develop a cost-effective Chebyshev's mechanism. The proposed method applies two linear springs that generate the required torque and motion. This paper describes the optimization procedure that determines springs' optimal positions on mechanism. Results of their optimizations are tested and verified with help of Multibody Dynamics analysis software ALASKA.

Xydas and Mueller in [16] compared Watt's, Chebyshev's and Hoeken's mechanisms and analyzed required torque for varying input and output links. In addition, they modified the mechanisms by adding a flywheel with different masses. Changing the mass of the flywheel for Watt's mechanism had a relatively small impact on the required input torque in comparison with other two mechanisms. They also describe the differences of required input torque and power of motor that is installed on output link when linear springs are attached to input link. The results of their work show that application of linear springs on input link leads to a required motor power decrease for all three mechanisms.

Kurmashev et al. in [17] designed and manufactured upper limb rehabilitation apparatus. Their work describes synthesis of Hoeken's mechanism

that performs straight-line trajectory. They performed kinematic and kinetostatic analysis of the mechanism. They considered various configurations of the mechanism to generate different curved trajectories. They carried out inertial optimization of the mechanism by adjustment of Center of Gravities of each link in order to minimize the required input torque. They also investigated the case, when two linear springs can be used instead of driving motor. Their work contains detailed description of the manufacturing process of the mechanism. Moreover, choosing of motor and programming of controller were considered. Kurmashev et al. in [18] simulated the mechanism designed in [17] with help of Multibody Dynamics environment using ALASKA. Simulation results show that spring and motor characteristics found for straight-line motion can be applied for curved motion.

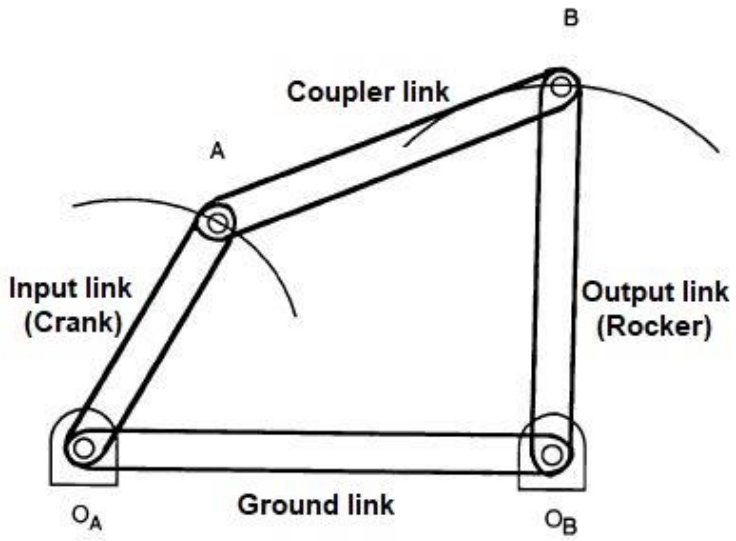
The last five above mentioned works provided the motivation and basis for this thesis.

2.2 Four bar linkage mechanism

Many research works are focused on four bar linkage mechanisms. The four bar linkage (see Figure 2.1) is a flexible mechanism that is applied to transfer motion or ensure mechanical advantage. It is more preferable in comparison with other mechanisms due to its simplicity in manufacturing, stability of performance and low friction. Four bar linkage consist of four links: input link (crank), output link (rocker), ground link and coupler link. One end of input and output links are connected to a fixed revolute joint on the ground and other ends are linked with a

coupler link by revolute pair joints. A point on the coupler link is called coupler point, while in robotics this is considered as an end-effector. When the input link is rotating the coupler point follows a trajectory called the coupler curve. Various coupler curves can be obtained by modification of links' lengths [19].

Figure 2.1: Four bar mechanism [19]

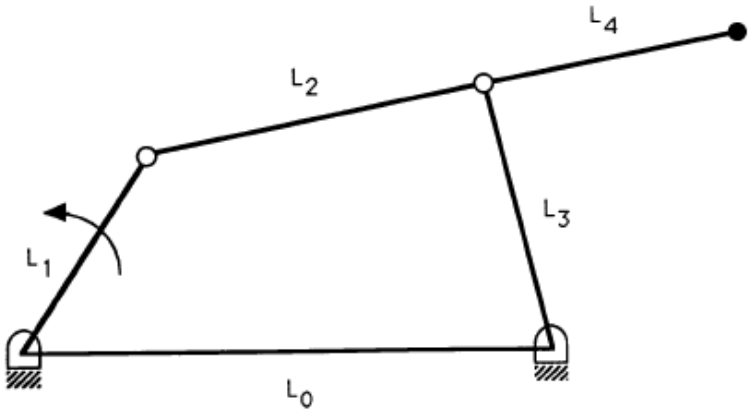


Four bar mechanisms are classified into two main groups based on the rotational ability of their links. In 1883, Grashof presented a simple rule that determines the links' rotational ability: consider a four bar linkage where l_1 is the shortest link, l_4 is the longest link, and the remaining two links' lengths are l_2 and l_3. Grashof claimed that if $l_1 + l_4 \leq l_2 + l_3$, then at least one of links can fully rotate with respect to the other three link, otherwise none of links can perform a full rotation [20]. Hence, four bar mechanisms are divided into Grashof and Non-Grashof mechanisms.

Further to this categorization, there are many types of four bar mechanisms such as Hoeken's, Watt's, Roberts's, Chebyshev's and Peaucellier's. In this work,

Hoeken's four bar mechanism (see Figure 2.2) is used due to its ability to generate both straight-line and curved motion.

Figure 2.2: Hoeken's four bar mechanism [19]



2.3 Minimum Jerk Model

Flash and Hogan in [21] formulated a mathematical model that describes the coordination of human arm movements. This is accomplished by the minimization of the following function:

$$C = \frac{1}{2}\int_0^{t_f}\left(\left(\frac{d^3x}{d^3t}\right)^2 + \left(\frac{d^3y}{d^3t}\right)^2\right)dt \tag{2.1}$$

where, x and y are time-varying coordinates of the hand position in a Cartesian coordinate system, third-order derivatives of coordinates correspond to jerk and t_f is the total movement time. Mathematical expression of $x(t)$ and $y(t)$ are derived that minimizes objective function. For the straight-line motion these expressions are as follows:

$$x_{MJM}(t) = x_0 + (x_0 - x_f)(15\tau^4 - 6\tau^5 - 10\tau^3) \tag{2.2}$$

$$y_{MJM}(t) = y_0 + (y_0 - y_f)(15\tau^4 - 6\tau^5 - 10\tau^3) \quad (2.3)$$

where, $\tau = t/t_f$, x_0, y_0 and x_f, y_f are initial and final positions respectively.

To find the position expressions for curved motion, it was assumed that in order to transfer hand from initial position to the final position in defined time it should pass through third specified positions x_1 and y_1 at undetermined time t_1. Defining of time t_1 is come from optimization technique that corresponds to problem with interior point equality constraints [21]. Results of optimization indicate the position expressions:

$$x(t \le t_1) = \frac{{t_f}^5}{720} * [\pi_1 * ({\tau_1}^4 * (15 * \tau^4 - 30 * \tau^3) + {\tau_1}^3 * (80 * \tau^3 - 30 * \tau^4) - 60 * \tau^3 * {\tau_1}^2 + 30 * \tau^4 * \tau_1 - 6 * \tau^5) + c_1 * (15 * \tau^4 - 10 * \tau^3 - 6 * \tau^5)] + x_0 \quad (2.4)$$

$$y(t \le t_1) = \frac{{t_f}^5}{720} * [\pi_2 * ({\tau_1}^4 * (15 * \tau^4 - 30 * \tau^3) + {\tau_1}^3 * (80 * \tau^3 - 30 * \tau^4) - 60 * \tau^3 * {\tau_1}^2 + 30 * \tau^4 * \tau_1 - 6 * \tau^5) + c_2 * (15 * \tau^4 - 10 * \tau^3 - 6 * \tau^5)] + y_0 \quad (2.5)$$

$$x(t > t_1) = \frac{{t_f}^5}{720} * \left[\pi_1 * \left({\tau_1}^4 * (15 * \tau^4 - 30 * \tau^3 + 30 * \tau - 15) + {\tau_1}^3 * (80 * \tau^3 - 30 * \tau^4 - 60 * \tau^2 + 10)\right) + c_1 * (15 * \tau^4 - 10 * \tau^3 - 6 * \tau^5 + 1)\right] + x_f \quad (2.6)$$

$$y(t > t_1) = \frac{{t_f}^5}{720} * \left[\pi_2 * \left({\tau_1}^4 * (15 * \tau^4 - 30 * \tau^3 + 30 * \tau - 15) + {\tau_1}^3 * (80 * \tau^3 - 30 * \tau^4 - 60 * \tau^2 + 10)\right) + c_2 * (15 * \tau^4 - 10 * \tau^3 - 6 * \tau^5 + 1)\right] + y_f \quad (2.7)$$

where, $\tau_1 = t_1/t_f$. The constant coefficients c_1, c_2, π_1 and π_2 are as following:

$$c_1 = \frac{1}{{t_f}^5 * {\tau_1}^2 * (1-\tau_1)^5} * \left[\left(x_f - x_0\right) * (300 * {\tau_1}^5 - 1200 * {\tau_1}^4 + 1600 * {\tau_1}^3) + {\tau_1}^2 * \left(-720 * x_f + 120 * x_1 + 600 * x_0\right) + (x_0 - x_1) * (300 * \tau_1 - 200)\right] \quad (2.8)$$

$$c_2 = \frac{1}{{t_f}^5 * {\tau_1}^2 * (1-\tau_1)^5} * \left[\left(y_f - y_0\right) * (300 * {\tau_1}^5 - 1200 * {\tau_1}^4 + 1600 * {\tau_1}^3) + {\tau_1}^2 * \left(-720 * y_f + 120 * y_1 + 600 * y_0\right) + (y_0 - y_1) * (300 * \tau_1 - 200)\right] \quad (2.9)$$

$$\pi_1 = \frac{1}{{t_f}^5 * {\tau_1}^5 * (1-\tau_1)^5} * \left[\left(x_f - x_0\right) * (120 * {\tau_1}^5 - 300 * {\tau_1}^4 + 200 * {\tau_1}^3) - 20 * (x_1 - x_0)\right] \quad (2.10)$$

$$\pi_2 = \frac{1}{{t_f}^5 * {\tau_1}^5 * (1-\tau_1)^5} * \left[\left(y_f - y_0\right) * (120 * {\tau_1}^5 - 300 * {\tau_1}^4 + 200 * {\tau_1}^3) - 20 * (y_1 - y_0)\right] \quad (2.11)$$

Thus, coupler point path of four bar linkage should match to the MJM trajectory.

In order to define the total time of movement Fitts's law are applied. Fitts's law states that the time required to move hand to specified target is a function of

the distance to the target divided by the size of the target. Fitts derived the following equation that compute movement time to hit the target:

$$t_f = a + b * log_2(\frac{2*A}{W}) \quad (2.12)$$

where, A is the target distance, W is the width of the target, a and b are intercept and slope respectively that are calculated experimentally. Experiment that computes total movement time conducted in [17] and it is known that total time is equal to 0.5 seconds.

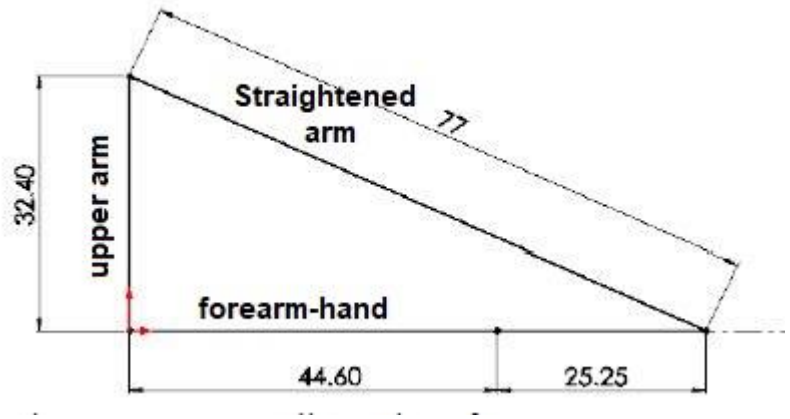
Moreover, coupler point conforming to MJM has to abide a straight-line motion with allowable fluctuations. These instabilities should be within the framework of defined range called Just Noticeable Difference (JND). These differences can be measured by three method: constant stimuli method, limit method and adjustment method. Constant stimuli method consist of set of trials. Standard and comparison stimulus are presented at each trial. The comparison stimulus is selected randomly from a set of 5-9 variations. Percentage of trials are calculated when selected stimulus is larger than standard one. The JND is defined as the difference between the two comparison stimulus which were evaluated larger in 75% (or 25%) cases. Limit method has the same approach, but comparison stimulus starts from much lower values than standard one and rises incrementally. In adjustment method, comparison stimulus can be changed until it is concerned as equal to standard one. The JND is estimated as a standard deviation of the comparison stimulus from standard one [17].

Chapter 3 – Mechanism's design & Analysis

3.1 Dimensional synthesis

The dimensions of four bar linkage are obtained from defining the length of straight-line trajectory. To define the length of trajectory the anthropometric data from NASA is used [22]. According to this data, length of upper arm is 32.4 cm and length of forearm-hand is 44.6 cm. The length of trajectory can be taken from difference of initial position when arm is bent and final position when arm straightened. Graphical representation of arm position is presented in Figure 3.1.

Figure 3.1: Dimensions of arm [17]

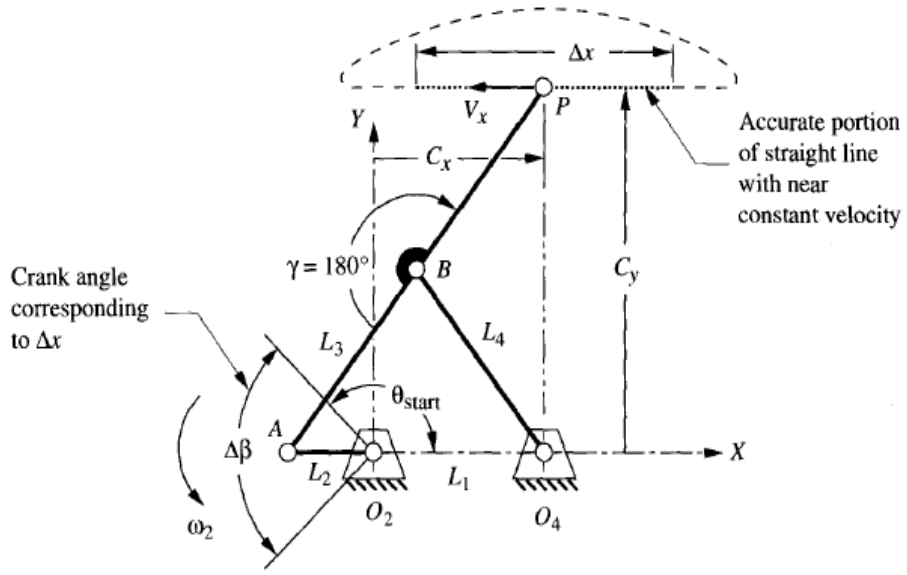


According to the Figure 3.1 the length of straight line trajectory is 25.25 cm. According to the experiments, conducted in [17] arm movement diverges

from the straight-line trajectory no more than 1 degree to up or to down. Thus, the deviation of straight-line trajectory is equal to $\Delta = 0.2525 * \tan(1) = 0.0044$ m.

The next step is to define links' lengths of the mechanism. Hoeken four bar linkage can give suitable straightness of coupler point and it is crank-rocker, which means drive motor can be employed. Hoeken mechanism's dimensions and coupler point path is presented in Figure 3.2.

Figure 3.2: Hoeken's four bar mechanism with dimensions [17]



Since it is symmetrical four bar linkage and γ=180, so $L_3 = L_4 = BP$. In this case, two link ratios are enough, suppose they are L_1/L_2 and L_3/L_2. A research was implemented to define the errors in straightness over different values of $\Delta\beta$ of the 2nd link cycle as a function of the link ratios. Errors were calculated separately for each angle range $\Delta\beta$ from 20 to 180 degrees [23]. The results of errors is shown in Table 3.1.

In order to follow compactness and low cost, it is decided to select last configuration when $\Delta\beta$ is equal to 180 degrees. According to the Table 3.1, the lengths of links are need to be:

$$L_2 = \Delta x/4.181 = 0.2525/4.181 = 0.0604 \text{ m} \quad (3.1)$$

$$L_3 = 2.8 * L_2 = 0.169 \text{ m} \quad (3.2)$$

$$L_1 = 2.2 * L_2 = 0.133 \text{ m} \quad (3.3)$$

Table 3.1: Link ratios for smallest obtainable error in straightness [17]

Range of Motion			Optimized for Straightness					
$\Delta\beta$ (deg)	θ_{start} (deg)	% of cycle	Minimum ΔC_y %	ΔV %	$V_x/(l_2\omega_2)$	Link Ratios		
						l_1/l_2	l_3/l_2	$\Delta x/l_2$
20	170	5.6%	0.00001%	0.38%	1.436	2.975	3.963	0.601
40	160	11.1%	0.00004%	1.53%	1.504	2.950	3.925	1.193
60	150	16.7%	0.00027%	3.48%	1.565	2.900	3.850	1.763
80	140	22.2%	0.001%	6.27%	1.611	2.825	3.738	2.299
100	130	27.8%	0.004%	9.90%	1.646	2.725	3.588	2.790
120	120	33.3%	0.010%	14.68%	1.679	2.625	3.438	3.238
140	110	38.9%	0.023%	20.48%	1.702	2.500	3.250	3.623
160	100	44.4%	0.047%	27.15%	1.717	2.350	3.025	3.933
180	90	50.0%	0.096%	35.31%	1.725	2.200	2.800	4.181

3.2 Kinematic analysis

Kinematic analysis of the mechanism performed in this section in order to determine angles, positions, velocities and accelerations of the links. First, the mechanism model with known dimensions was assembled in GIM software (see Figure 3.3) to get initial, intermediate and final positions of coupler point for

straight-line and curved motion. Then, position analysis is done. Coupler curve should correspond to MJM, so specific angles θ_2, θ_3 and θ_4 are calculated for the known coupler point position. Vector loop equation and Freudenstein approach used to compute these angles. The links are represented as a position vectors. The choice of vector direction that depicted in Figure 3.4 lead to this vector loop equation:

$$R_2 + 2 * R_3 - R_p = 0 \tag{3.4}$$

Equation 3.4 can be written in Polar form:

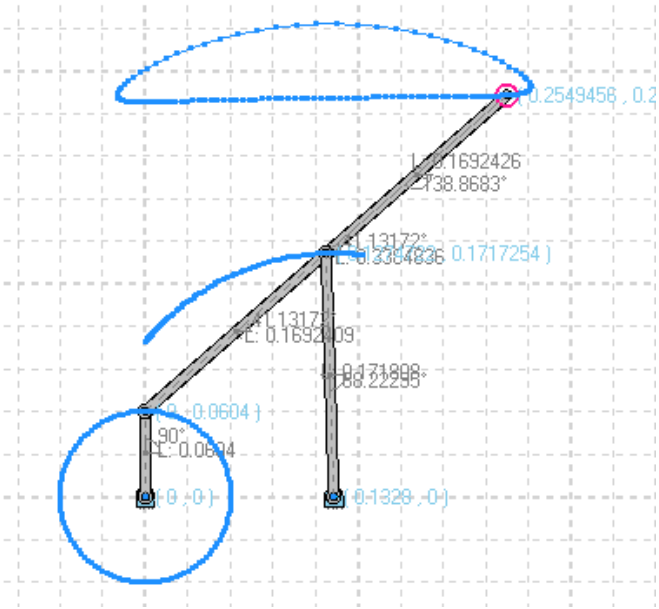
$$L_2 e^{j\theta 2} + 2 * L_3 e^{j\theta 3} - (RP_x(t) + iRP_y(t)) = 0 \tag{3.5}$$

By using Euler identity and dividing into real and imaginary part equation 3.5 can be written in Cartesian form:

$$2 * L_3 cos\theta_3 = RP_x(t) - L_2 cos\theta_2 \tag{3.6}$$

$$2 * L_3 sin\theta_3 = RP_y(t) - L_2 sin\theta_2 \tag{3.7}$$

Figure 3.3: Coupler path of mechanism [GIM software]



By squaring both sides of equations 3.6 and 3.7, adding them and using half angle identities $\sin\theta_2 = \frac{2\tan\theta_2/2}{1+tan^2(\theta_2/2)}$ and $\cos\theta_2 = \frac{1-tan^2(\theta_2/2)}{1+tan^2(\theta_2/2)}$, following quadratic equation is obtained:

$$A_2 tan^2(\theta_2/2) + B_2 tan(\theta_2/2) + C_2 = 0 \quad (3.8)$$

where, $A_2 = {RP_x}^2 + 2 * RP_x + {L_2}^2 + {RP_y}^2 - 4 * {L_3}^2$ (3.9)

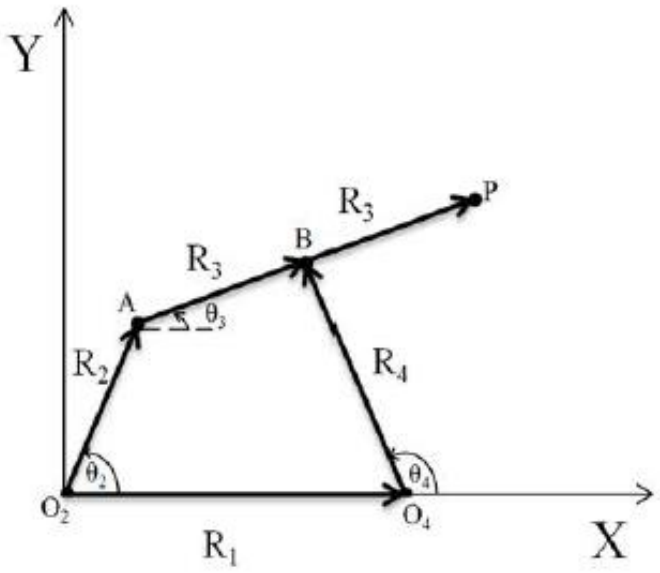
$$B_2 = -4 * RP_y * L_2 \quad (3.10)$$

$$C_2 = {RP_x}^2 - 2 * RP_x L_2 + {L_2}^2 + {RP_y}^2 - 4 * {L_3}^2 \quad (3.11)$$

Then angle θ_2 expression is equal to:

$$\theta_2(t) = 2 * atan(\frac{-B_2 \pm sqrt({B_2}^2 - 4A_2C_2)}{2A_2}) \quad (3.12)$$

Figure 3.4: Vector representation of mechanism [17]



Equation (3.12) can have different solutions. First case, if the discriminant under the radical is negative, solution is complex conjugate. It means that selected links' lengths are not able to connect with each other at given angle θ_2. When

solutions is real and unequal, these are treated as for open and crossed configurations of the mechanism. Negative solution is used for open configuration, positive solution is referred to crossed configuration. Four bar linkage is considered as a crossed if two links that adjoining to the shortest link are crossed, otherwise it has open configuration. In this work, open configuration refers to straight-line motion, crossed configuration for curved motion.

Equations for angles θ_3 and θ_4 is found in the same manner as for angle θ_2. After that, velocity analysis should be done in order to define angular velocities of each link. Velocity expressions were derived by differentiating equation 3.5 respect to time and obtaining velocity difference equations. As a results, velocity matrix is as follows:

$$\begin{bmatrix} -L_2\sin\theta_2 & -L_3\sin\theta_3 & L_4\sin\theta_4 & 0 & 0 & 0 & 0 & 0 \\ L_2\cos\theta_2 & L_3\cos\theta_3 & -L_4\cos\theta_4 & 0 & 0 & 0 & 0 & 0 \\ L_2\sin\theta_2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ -L_2\cos\theta_2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 2L_3 sin\theta 3 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -2L_3\cos\theta_3 & 0 & 0 & 0 & 0 & 1 & 0 \\ -L_2\sin\theta_2 & -2L3\sin\theta_3 & 0 & 0 & 0 & 0 & 0 & 0 \\ -L_2\cos\theta_2 & -2L3\cos\theta_3 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_2 \\ \omega_3 \\ \omega_4 \\ VAx \\ VAy \\ VPAx \\ VPAy \\ VPy \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ VPx \\ 0 \end{bmatrix} \quad (3.13)$$

where, ω_n is the angular velocity of n[th] link. VPx is the linear velocity of coupler point, which is derived by differentiating equation 2.2 respect to time and corresponds to MJM. Above matrix is solved numerically in MATLAB and unknown velocities are computed. To calculate angular accelerations of each link the same method is used as in velocity analysis, apart from twice differentiating respect to time. As a result, acceleration matrix is as follows:

$$\begin{bmatrix} -L_2\sin\theta_2 & -L_3\sin\theta_3 & L_4\sin\theta_4 & 0 & 0 & 0 & 0 & 0 \\ L_2\cos\theta_2 & L_3\cos\theta_3 & -L_4\cos\theta_4 & 0 & 0 & 0 & 0 & 0 \\ L_2\sin\theta_2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ -L_2\cos\theta_2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 2L_3 sin\theta_3 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -2L_3\cos\theta_3 & 0 & 0 & 0 & 0 & 1 & 0 \\ -L_2\sin\theta_2 & -2L_3\sin\theta_3 & 0 & 0 & 0 & 0 & 0 & 0 \\ -L_2\cos\theta_2 & -2L_3\cos\theta_3 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ aAx \\ aAy \\ APAx \\ APAy \\ APy \end{bmatrix} =$$

$$\begin{bmatrix} L_2\omega_2^2 cos\theta_2 + L_3\omega_3^2 cos\theta_3 - L_4\omega_4^2 cos\theta \\ L_2\omega_2^2 sin\theta_2 + L_3\omega_3^2 sin\theta_3 - L_4\omega_4^2 sin\theta \\ -L_2\omega_2^2 cos\theta_2 \\ -L_2\omega_2^2 sin\theta_2 \\ -2L_3\omega_3^2 cos\theta_3 \\ -2L_3\omega_3^2 sin\theta_3 \\ APx + L_2\omega_2^2 cos\theta_2 + 2L_3\omega_3^2 cos\theta_3 \\ -L_2\omega_2^2 sin\theta_2 + 2L_3\omega_3^2 sin\theta_3 \end{bmatrix} \quad (3.14)$$

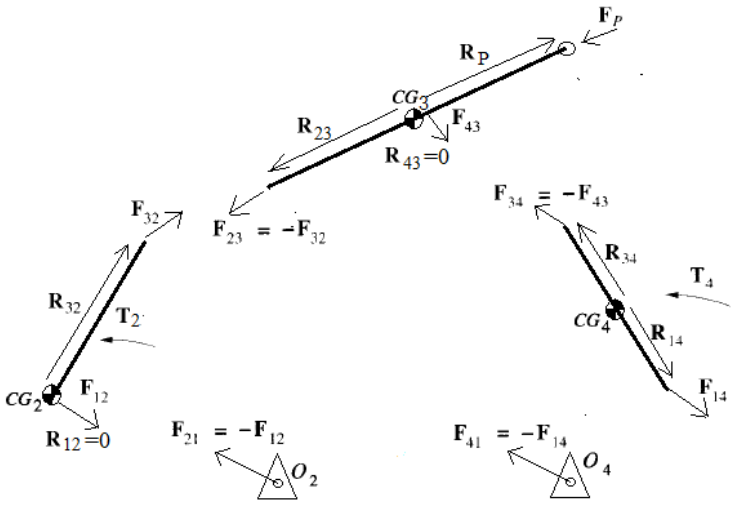
where, α_n is the angular acceleration of n[th] link and APx is linear acceleration of coupler point, which is derived by differentiating equation 2.2 twice respect to time and corresponds to MJM. This matrix is also solved numerically in MATLAB and unknown accelerations are computed. Detailed procedure of position, velocity and acceleration analysis of four bar linkage are illustrated in [23].

3.3 Kinetostatic analysis

After defining link positions, its lengths, angular velocities and angular accelerations of each link it is required to implement force analysis. These analysis include calculation of all internal forces acting on pin joints, external forces and required torque to perform motion corresponding to MJM. In order to compute masses and moments of inertia of links 20x20mm bars made of

aluminum are used [17]. Free body diagrams (see Figure 3.5) of each link are drawn to identify all forces and torque acting on it. The position vectors R_{nn} of each link are deduced depending on center of mass. As a first guess, center of mass will be at the center of each link, apart from input link. Center of mass of input link will be at pivot O_2 due to using flywheel with mass of 2.5 kg.

Figure 3.5: Free body diagram of links



Expressions for forces and torques can be derived by using Newton's laws. These laws can be written in a form of summation of all forces and torque in the system:

$$\sum F = ma \qquad \sum T = I_G \alpha \tag{3.15}$$

where, I_G is the moment of inertia.

According to equations 3.15, following expressions are written for link 2:

$$F_{12x} + F_{32x} = m_2 a_{G2x} \tag{3.16}$$

$$F_{12y} + F_{32y} = m_2 a_{G2y} \tag{3.17}$$

$$T_2 + (R_{12x}F_{12y} - R_{12y}F_{12x}) + (R_{32x}F_{32y} - R_{32y}F_{32x}) = I_{G2}\alpha_2 \tag{3.18}$$

Where T_2 is torque that is needed to generate coupler curve corresponding to MJM. Important to note that for that particular case, torque T_2 is considered as the required input torque since drive motor is installed on input link. According to Figure 3.5, external torque T_4 is assumed to be zero. Then relevant expressions are deduced for other links and force matrix is as follows:

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -R12y & R12x & -R32y & R32x & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & R23y & -R23x & -R43y & R43x & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & R34y & R34x & -R14y & R14x & 0 \end{bmatrix} \begin{bmatrix} F_{12x} \\ F_{12y} \\ F_{32x} \\ F_{32y} \\ F_{43x} \\ F_{43y} \\ F_{14x} \\ F_{14y} \\ T_2 \end{bmatrix} =$$

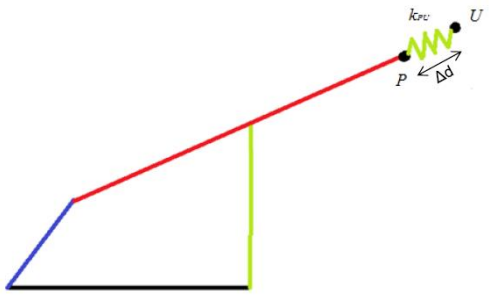
$$\begin{bmatrix} m_2 a_{G2x} \\ m_2 a_{G2y} \\ I_{G2}\alpha_2 \\ m_3 a_{G3x} - FP_x \\ m_3 a_{G3y} - FP_y \\ I_{G3}\alpha_3 - RP_X * FP_y + RP_y * FP_x \\ m_4 a_{G4x} \\ m_4 a_{G4y} \\ I_{G4}\alpha_4 \end{bmatrix} \quad (3.19)$$

where, FP_x is the external force acting on coupler point and RP_x is its position vector. T_2 is the required input torque on input link that performs desired motion. Patient's force can be considered as an external force. During rehabilitation, movement trajectory of patient's arm differs from MJM due to his disability. External force can be calculated by the application of imaginary spring between patient's hand and coupler point (see Figure 3.6). The imaginary spring stiffness is as follows:

$$k_{PU} = \frac{F_{PUmax}}{\Delta d_{max}} \tag{3.20}$$

where, Δd_{max} is maximum difference between patient's hand and coupler point positions. Detailed approach of this application are described in [24]. The maximum patient's force F_{PUmax} is equal to 15 N, which experimentally measured in [24]. External force are calculated using resulting spring stiffness and deviation of MJM trajectory from patient's hand trajectory. Profile of patient's hand trajectory are experimentally found in [25].

Figure 3.6: An imaginary spring between user's hand and coupler point [17]



Matrix 3.19 is solved numerically in MATLAB and required torque T_2 is defined.

3.4 Inertial optimization

The main aim of this work is to design such mechanism that will be less expensive than complex rehabilitation robotic systems, but as effective as possible. This can be achieved by decreasing the power of drive motor. One way of minimizing required input torque found in force analysis section is change the position of link's Center of Gravities (CG). The optimization of CG's position

based on adjustment of the position vectors of each link. Masses of links and moment of inertia were remained the same. The position vectors were altered within following range: $R_{CG} \in [0{,}1 * L_n;\ 0{,}9 * L_n]$, where L_n is the length of input and output link.

3.5 Spring forces analysis

Linear springs can be employed as a passive control elements in order to generate required torque according to MJM model. For the Hoeken's mechanism linear springs can be set at output link. First, it is necessary to find required input torque T_4 acting on output link considering that external T_2 is equal to zero. Calculation of torque T_4 is performed in a same manner as torque T_2, but force matrix 3.19 transforms as following:

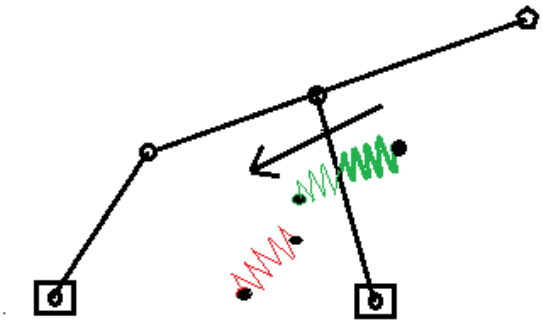
$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -R12y & R12x & -R32y & R32x & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & R23y & -R23x & -R43y & R43x & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & R34y & R34x & -R14y & R14x & 1 \end{bmatrix} \begin{bmatrix} F_{12x} \\ F_{12y} \\ F_{32x} \\ F_{32y} \\ F_{43x} \\ F_{43y} \\ F_{14x} \\ F_{14y} \\ T_4 \end{bmatrix} =$$

$$\begin{bmatrix} m_2 a_{G2x} \\ m_2 a_{G2y} \\ I_{G2}\alpha_2 \\ m_3 a_{G3x} - FP_x \\ m_3 a_{G3y} - FP_y \\ I_{G3}\alpha_3 - RP_X * FP_y + RP_y * FP_x \\ m_4 a_{G4x} \\ m_4 a_{G4y} \\ I_{G4}\alpha_4 \end{bmatrix} \tag{3.20}$$

One accelerating and one decelerating springs can be utilized to move the mechanism under the torque T_4. The principle is that accelerating spring speed up

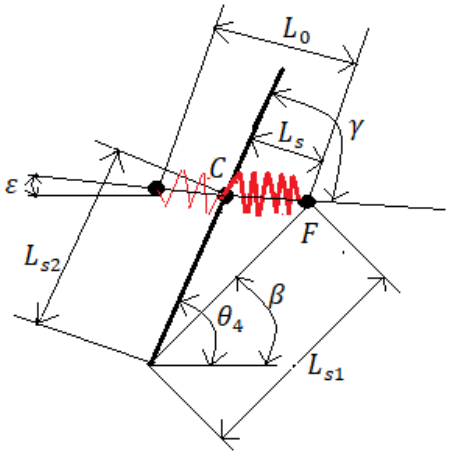
the mechanism until output link faces with decelerating spring, after that accelerating spring stops acting on output link. Then decelerating spring slows down the mechanism until it comes to the resting position and parallel to output link. When springs contacts with output link they can rotate around their fixed end. The visual presentation of two spring indicated in Figure 3.7. The green one is accelerating spring, the red one is decelerating spring.

Figure 3.7: Hoeken's mechanism with springs



To calculate spring force acting on output link only derivation of accelerating spring are considered, since calculation for second spring alike. Figure 3.8 depicts the position of accelerating spring with respect to output link.

Figure 3.8: Accelerating spring



Notations in Figure 3.8 are following: F is fixed point of spring to the ground, C is a contact point of spring with output link, L_{s1} is a distance from pivot point to fixed point, L_{s2} is a distance from pivot point to contact point, β is angle between fixed point and X abscissa, γ is angle between spring force and output link, L_0 is a free length of spring, L_s is the length of spring at a certain angular position θ_4 of output link, ε is the direction angle of spring force.

The length of spring is found from the cosine theorem:

$$L_s = \sqrt{L_{s1}^2 + L_{s2}^2 - 2L_{s1}L_{s2}\cos(\theta_4 - \beta)} \tag{3.21}$$

The angle γ between spring force and output can be derived by drawing right triangular and using trigonometry property:

$$\gamma = atan(\frac{L_{s1}\sin(\theta_4-\beta)}{L_{s1}\cos(\theta_4-\beta)-L_{s2}} \tag{3.22}$$

The spring force F_s can be obtained using Hooke's law:

$$F_s = k_s(L_s - L_0) \tag{3.23}$$

where, k_s is the spring stiffness.

In order to find torque T_s generating by spring, it is necessary to define the positon vector $\overrightarrow{R_C}$ of contact point and force vector $\overrightarrow{F_s}$ of spring force:

$$\overrightarrow{R_C} = L_{s2}\cos(\theta_4)\,\vec{\imath} + L_{s2}\sin(\theta_4)\,\vec{\jmath} \tag{3.24}$$

$$\overrightarrow{F_s} = -F_s\cos(\varepsilon)\,\vec{\imath} + F_s\sin(\varepsilon)\,\vec{\jmath} \tag{3.25}$$

By multiplying equations 3.24 and 3.25, torque T_s is derived:

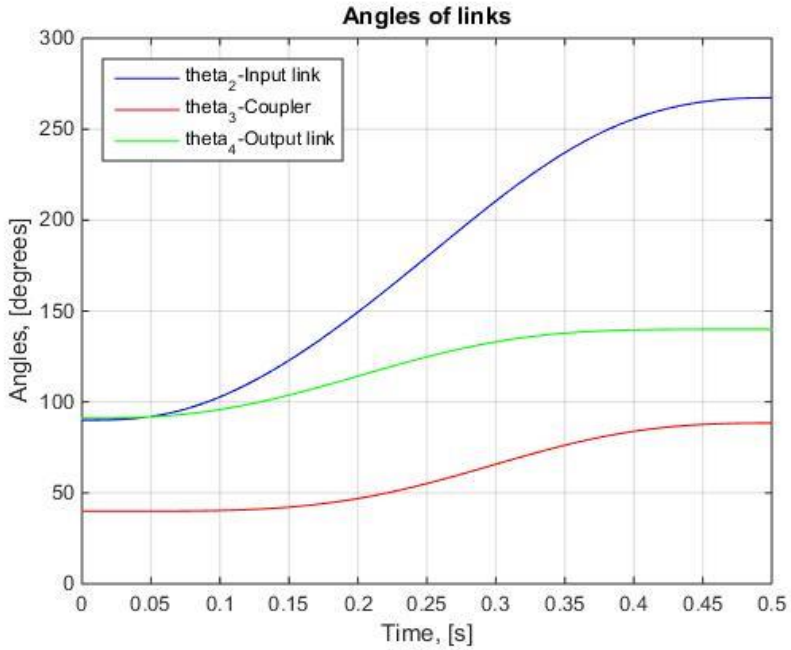
$$T_s = L_{s2}k_s(L_s - L_0)\ \sin(\gamma) \tag{3.26}$$

Chapter 4 – Implementation & Results

4.1 Straight-line motion

After defining MJM profile for straight-line motion, corresponding angles of links were determined. The results of position analysis are depicted in Figure 4.1.

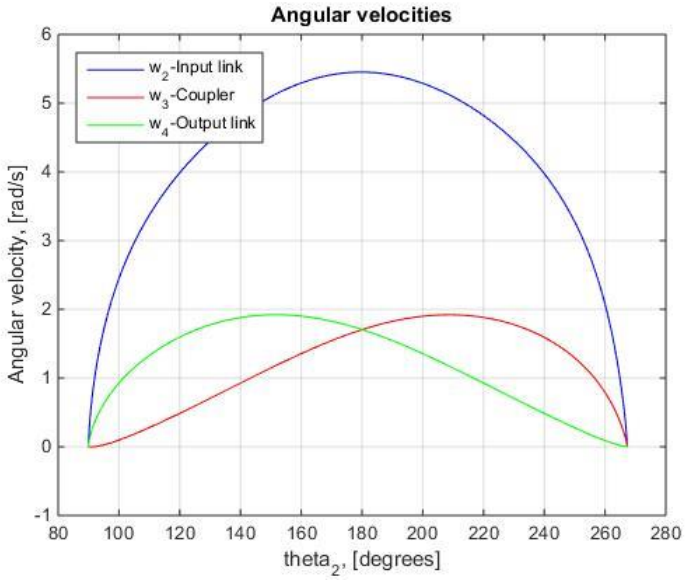
Figure 4.1: Angles of links



The results of velocity analysis are obtained by solving matrix 3.13. Angular velocities of each link are shown in Figure 4.2. The results of acceleration analysis are obtained by solving matrix 3.14. Angular accelerations of each link are shown in Figure 4.3.

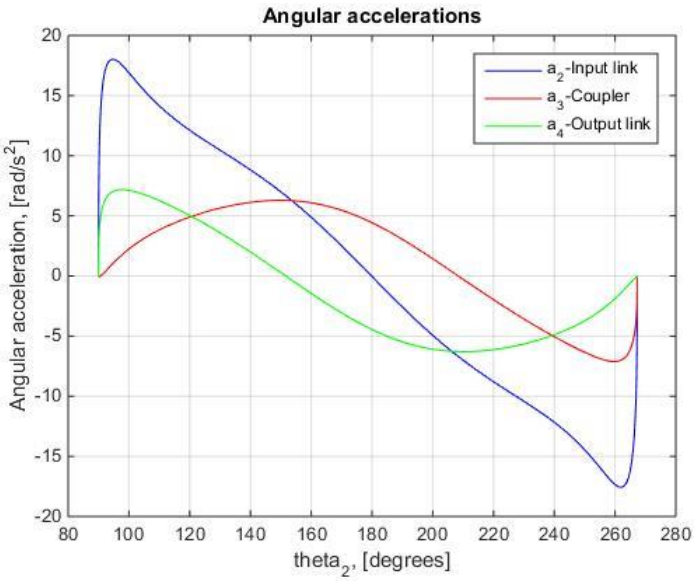
The impaired user's profile as mentioned in Chapter 3.3 was obtained from experiments conducted in [25]. The impaired user positon and MJM position on horizontal plane are shown in Figure 4.4.

Figure 4.2: Angular velocities of links



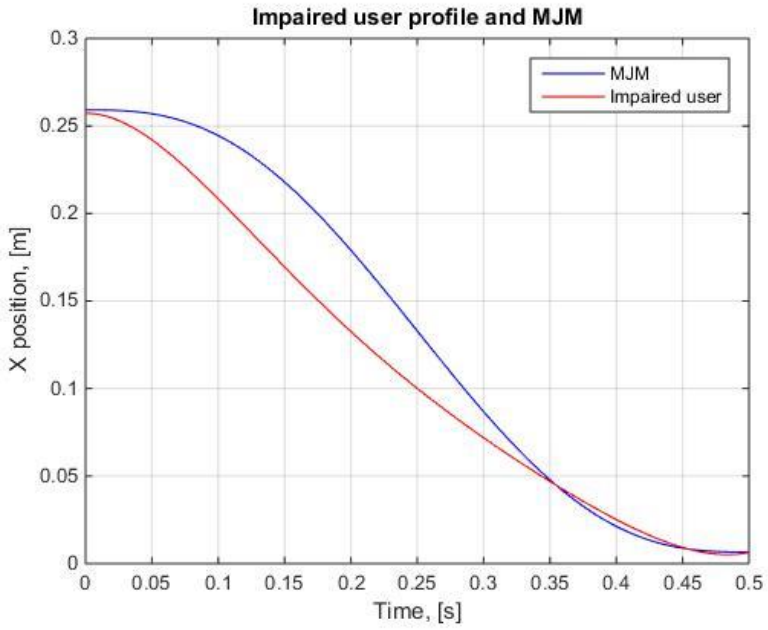
The difference between impaired user's profile and MJM profile was used to determine external force that acts on coupler point.

Figure 4.3: Angular accelerations of links



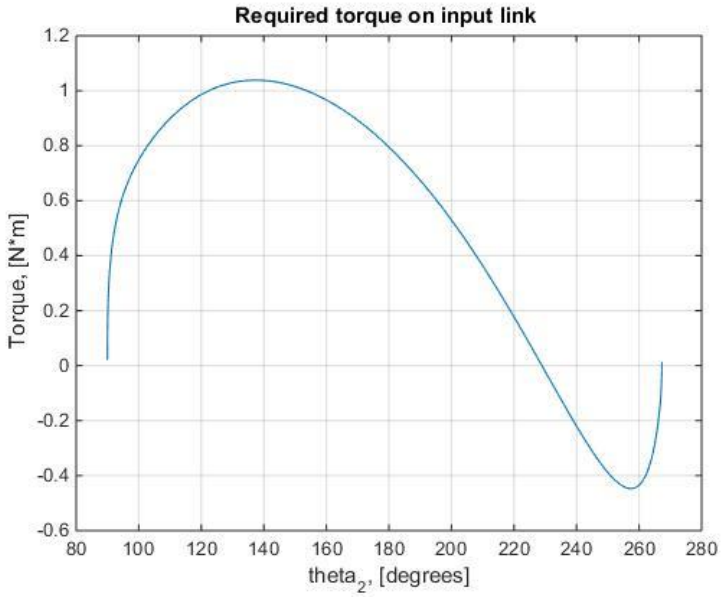
The maximum difference is equal to 0.0494 m. Thereby stiffness of imaginary spring is equal to 303.5170 N/m.

Figure 4.4: Impaired user's profile and MJM profile on horizontal plane



By solving force matrix 3.19 required torque on input link are calculated. Required torque on input link are shown in Figure 4.5.

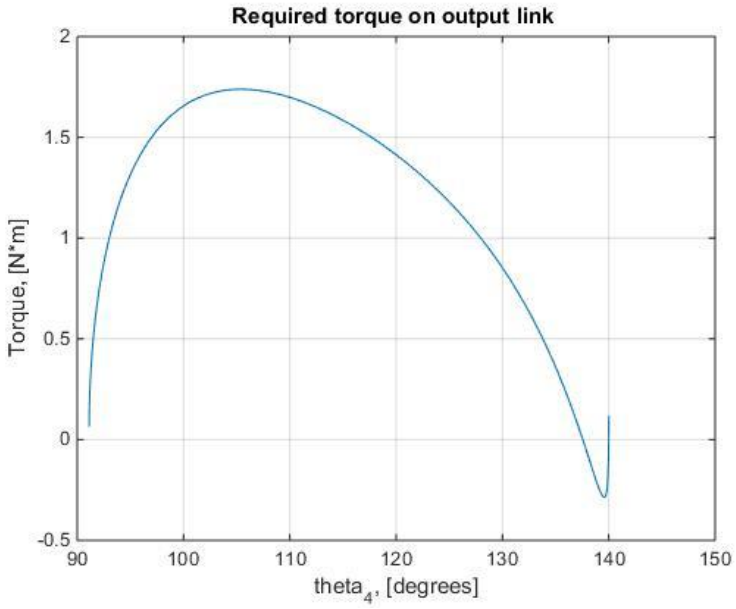
Figure 4.5: Required torque on input link



The maximum torque of 1.038 N*m on input link is required when angle of input link is about 136.5 degrees.

Required torque on output link are shown in Figure 4.6.

Figure 4.6: Required torque on output link



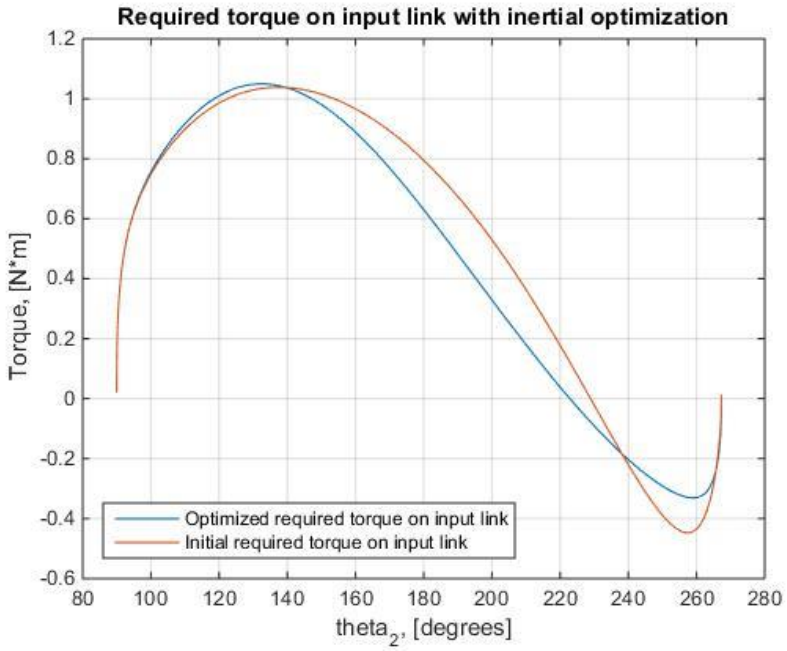
The maximum torque of 1.7493 N*m on output link is required when angle of output link is about 105.1 degrees.

Inertial optimization is performed in order to minimize required torque on input link. The CG positions of input and output links were changed. To find optimal position vectors fmincon function in MATLAB was applied. The results of inertial optimization depicted in Figure 4.7. According to solution, optimal input and output link's CG positions are 0.9*L_2 and 0.1*L_4 respectively. Results of inertial optimization show that there is no significant decreasing of the required input torque.

As mentioned in Chapter 3.5 two linear springs can be employed as a passive control elements. Due to dimension restrictions of second link, springs will be applied only on output link. The aim is to use these springs instead of

driving motor. The objective is to generate the torque by springs that will correspond to the required torque. This can be done by minimizing the difference between this torques. It was decided to minimize the sum of square of differences.

Figure 4.7: Required torque on input link with inertial optimization



Then optimization problem states as following:

$$min_{L_{s11},L_{s12},\beta_1,L_{s21},L_{s22},\beta_2,L_{01},L_{02},k_1,k_2} \sum(T_4 - T_s)^2 \qquad (4.1)$$

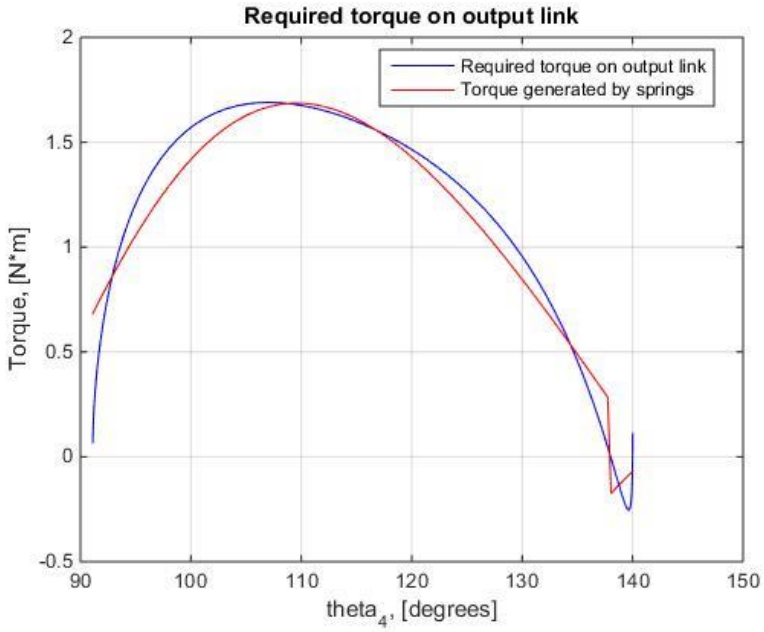
where, T_4 is the required torque on output link. There are 10 design variables; its notations are specified in Chapter 3.5. The results of optimal design variables are listed in Table 4.1. The optimization procedure is implemented with help of genetic algorithm function tool in MATLAB software package. The distances from fixed joint of output link to ends of decelerating spring attachment are limited between 0.03 m and 0.10 m due to dimension restrictions. Free length of springs are limited between 0.02 m and 0.10 m. Stiffness of springs varies from 0.005 to 3000.

Table 4.1: The optimal configurations of springs

Accelerating spring	Decelerating spring
L_{s11}=0.0458 m	L_{s21}=0.0789 m
L_{s12}=0.0845 m	L_{s22}=0.0618 m
β_1=85.04 deg	β_2=154.09 deg
L_{01}=0.0704 m	L_{02}=0.0229 m
k_1=2085.39	k_2=1143.08

The results of torque generated by springs are shown in Figure 4.8.

Figure 4.8: Torque generated by springs on output link



The results of optimization showed that torques generated by springs almost fits the required torque. However, it has sharp changes when the torque should be equal to zero and decelerating spring starts to slow down the mechanism. The high slope of accelerating at the beginning of motion is explained

by high force from compressed spring at the initial position. Due to the initial high force of spring, link meets decelerating spring earlier and at the end of motion the torque generated by decelerating spring is lower than the required torque.

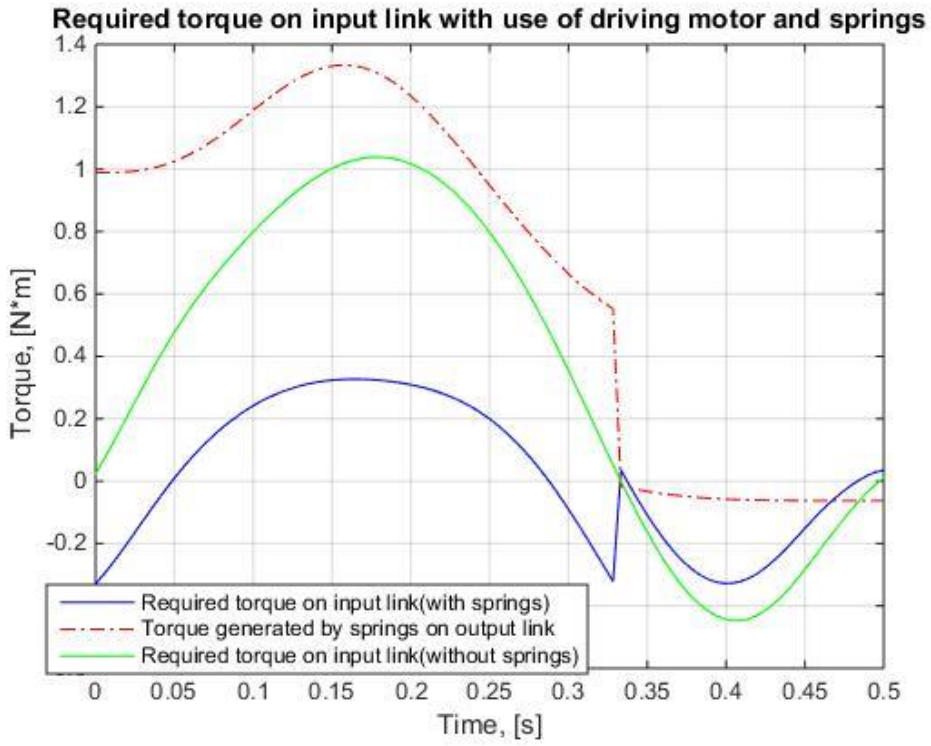
Another way of minimizing required torque is to apply springs as assistant control elements for driving motor. Springs can be installed on output link, while driving motor installed on input link. The results of optimization shown in Figure 4.9. The results of design variables are listed in Table 4.2.

Table 4.2: The optimal configurations of springs (with driving motor)

Accelerating spring	Decelerating spring
L_{s11}=0.0530 m	L_{s21}=0.0631 m
L_{s12}=0.0917 m	L_{s22}=0.0791 m
β_1=77.72 deg	β_2=217.29 deg
L_{01}=0.0928 m	L_{02}=0.0939 m
k_1=723.61	k_2=267.51

Using assistant springs significantly decreased the required torque for driving motor. As it can be seen from Figure below, mainly motion is performed by torque which generated by springs. At the beginning of motion driving motor's torque repeats the profile of required torque but with the lower amplitude. However, when accelerating spring "switches" to decelerating spring there is a sharp "jump" in torque. Then it has the similar profile with the required profile.

Figure 4.9: Required torque in case of use springs and driving motor



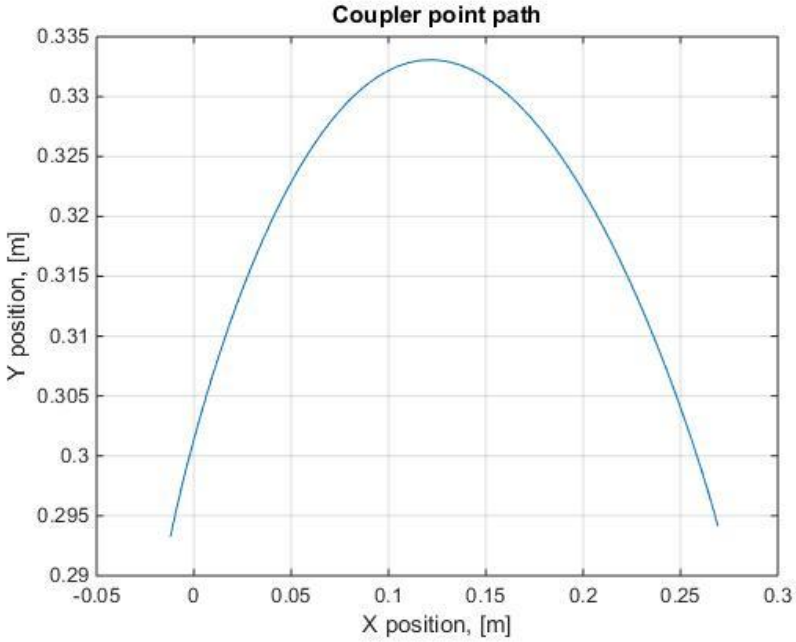
The MATLAB code for straight-line motion is shown in Appendix A.

4.2 Curved motion

Hoeken's mechanism can be used for curved motion. Curved trajectory exercises are more effectively than straight-line motion in upper limb rehabilitation. The position, velocity, acceleration and force analysis for curved motion has performed similarly as in straight-line motion, but with exception, that MJM has different equations.

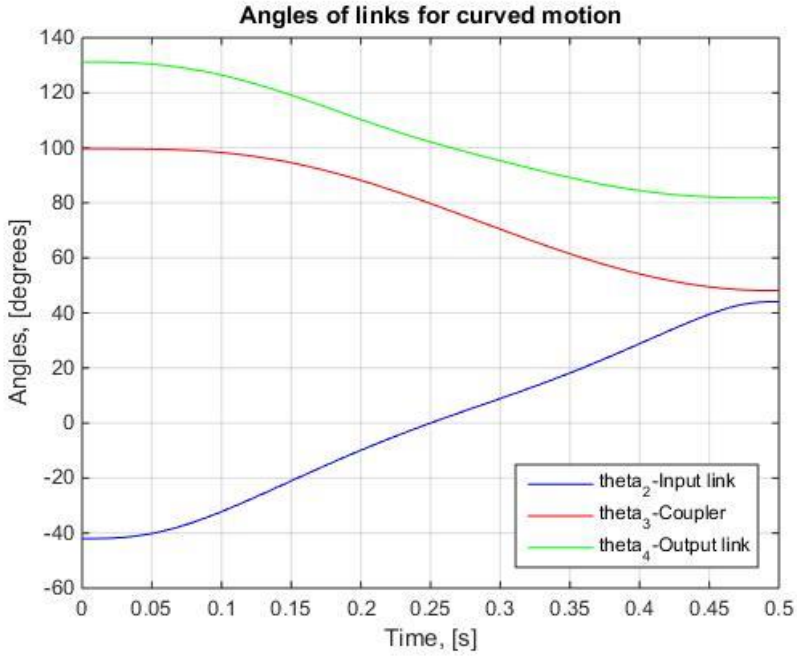
According to MJM of curved motion, the hand is required to pass via point at time t_1. This time is obtained from optimization technique [21] and equal to 0.243349. It was rounded off to 0.25. Coupler point path for curved motion is shown in Figure 4.10.

Figure 4.10: Curved motion path



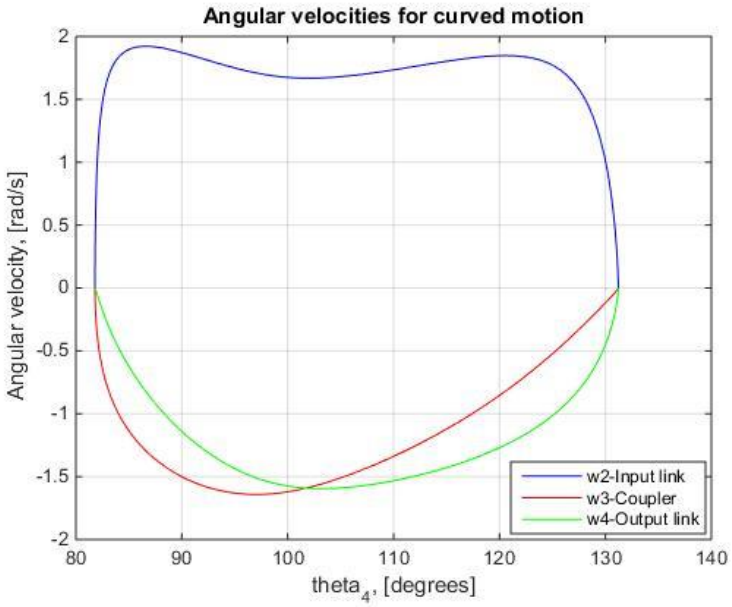
The results of position analysis for curved motion are shown in Figure 4.11.

Figure 4.11: Angles of links for curved motion



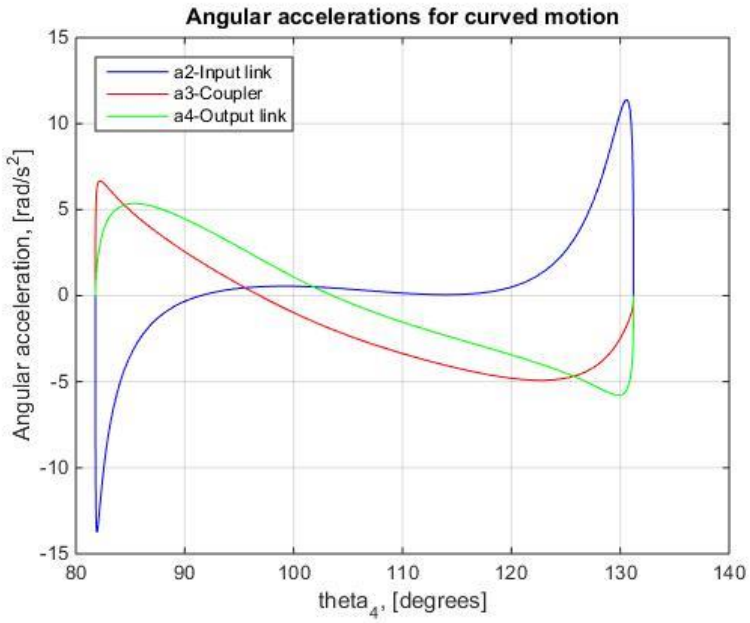
In case of curved motion, coupler and output links rotates in opposite direction compared to straight-line motion. Angular velocities and angular accelerations are depicted in Figures 4.12 and 4.13, respectively.

Figure 4.12: Angular velocities for curved motion



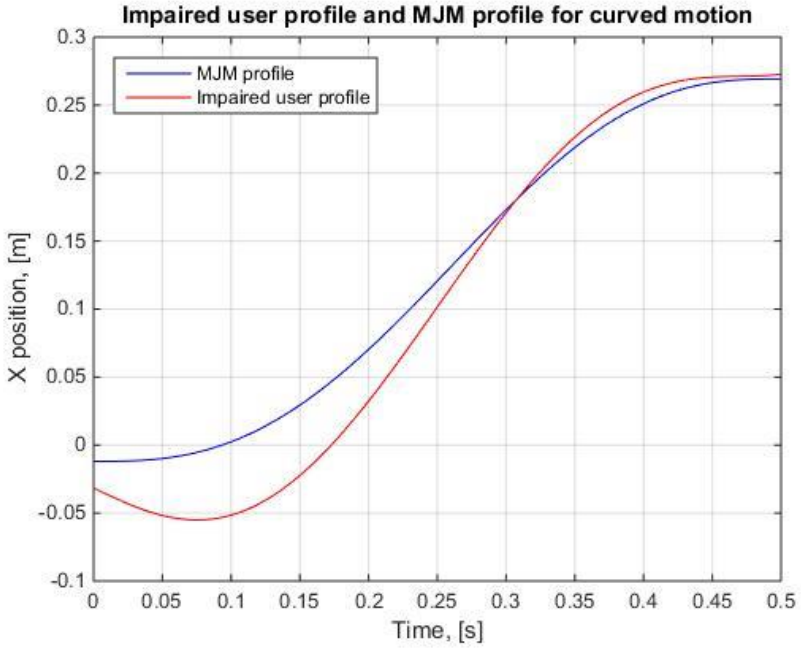
For the curved motion, it is assumed that only horizontal deviation of impaired user trajectory from MJM profile are considered.

Figure 4.13: Angular accelerations for curved motion



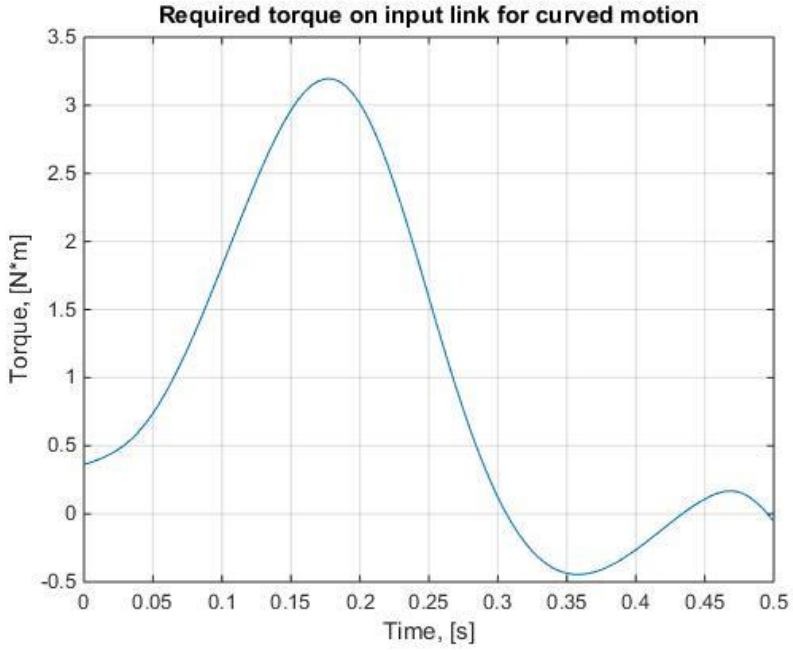
Thus, external force acts on coupler point only on X-direction. The impaired user position and MJM position for curved motion on horizontal plane is shown in Figure 4.14.

Figure 4.14: Impaired user profile and MJM profile for curved motion



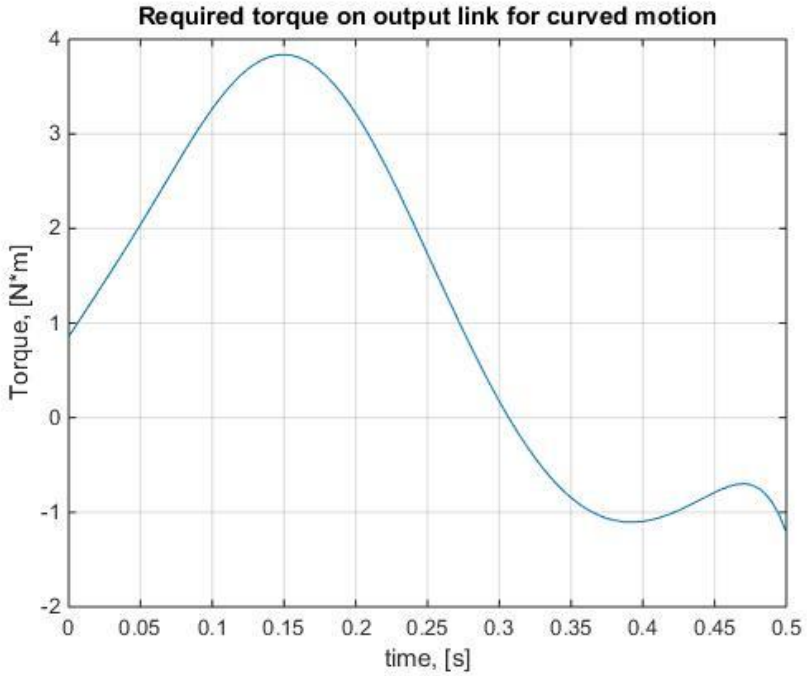
Required torque on input and output link for curved motion is depicted on Figures 4.15 and 4.16.

Figure 4.15: Required torque on input link for curved motion



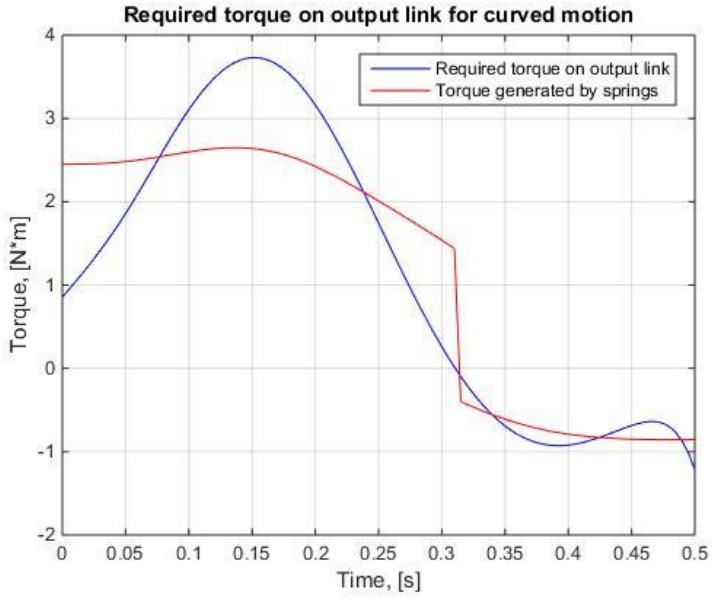
The maximum torque of 3.195 N*m on input link is required at the 0.175 seconds of motion.

Figure 4.16: Required torque on output link for curved motion



The maximum torque of 3.838 N*m on output link is required at the 0.15 seconds of motion. The same approach as for straight-line motion is used to optimize the torque generated by springs for curved motion. Torque profile of springs is shown in Figure 4.17.

Figure 4.17: Torque generated by springs on output link for curved motion



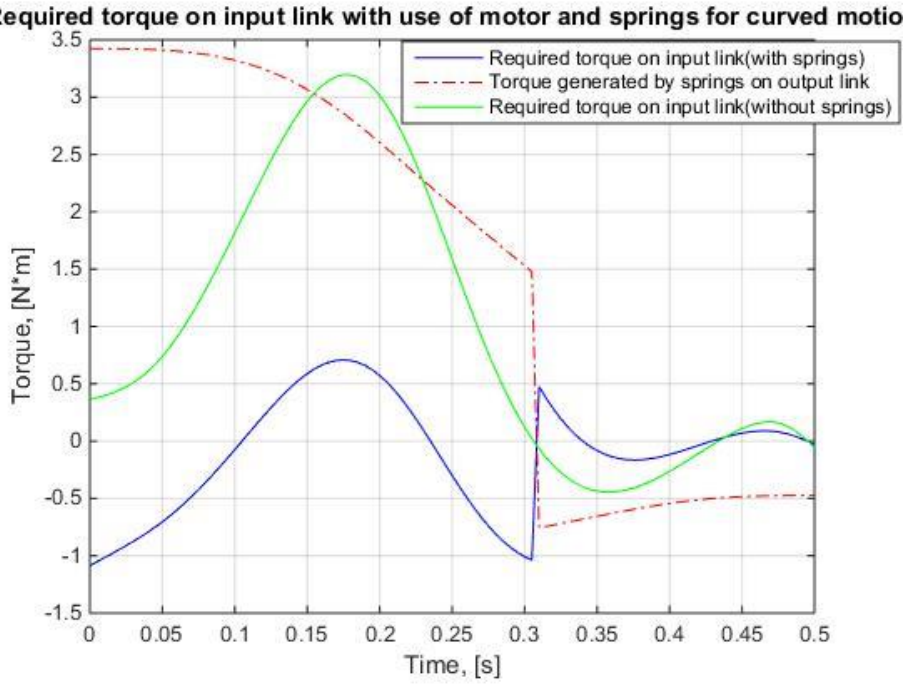
In case of curved motion, output link rotates in clockwise direction, so acceleration spring will located on the left side of output link. Due to dimension restrictions, the distance from fixed joint of output link to fixed end of accelerating spring is limited between 0.03 m and 0.08 m. The distance from fixed joint of output link to connection point with accelerating spring is limited between 0.03 m and 0.10 m. According to Figure 4.17, torque generated by accelerating spring does not ideally fit to required torque because of dimension restrictions. Results of optimum design variables are listed in Table 4.3.

Table 4.3: The optimal configurations of springs for curved motion

Accelerating spring	Decelerating spring
L_{s11}=0.0795 m	L_{s21}=0.08 m
L_{s12}=0.10 m	L_{s22}=0.0541 m
β_1=259.54 deg	β_2=61.30 deg
L_{01}=0.1399 m	L_{02}=0.0510 m
k_1=2999.99	k_2=1230.91

The use of spring with driving motor is also considered for curved motion. The required torques for driving motor and springs are depicted in Figure 4.18. As in straight-line motion, most of torque generated by spring and required torque on driving motor is decreased. But there is also sharp changes, when accelerating spring stops acting on output link.

Figure 4.18: Required torque in case of use springs and motor for curved motion



The optimal design variables are listed in Table 4.4.

Table 4.4: The optimal configurations of springs (with motor) for curved motion

Accelerating spring	Decelerating spring
L_{s11}=0.0599 m	L_{s21}=0.1393 m
L_{s12}=0.0699 m	L_{s22}=0.1393 m
β_1=257.80 deg	β_2=10.14 deg
L_{01}=0.0767 m	L_{02}=0.1317 m
k_1=2995.86	k_2=133.60

The MATLAB code for curved motion is shown in Appendix B. Above results showed that assistive springs can be applied in order to decrease the required torque for driving motor.

Chapter 5 – Conclusion

Mechanisms with one Degree of Freedom can be applied in upper limb rehabilitation in developing countries due to their simplicity and low cost. Despite their rather limited range of motions, they can be effectively used in treatment of patients suffered from stroke, multiple sclerosis and other neuro-disabilities.

In this thesis we analyzed Hoeken's four bar mechanism which can generate both straight-line and curved motion. The generated trajectories of our mechanism correspond to neurophysiological model such as the Minimum Jerk Model. Dimensional synthesis was performed to obtain the optimal length of links. Position angles, angular velocities and angular accelerations of each link were determined via kinematic analysis. Kinetostatic analysis was carried out to compute all forces acting on the mechanism and acquire the required torque that actuates mechanism according to MJM. Required torque minimization has been investigated via links' Center of Gravities modification and/or introduction of two appropriate linear springs. Change of link's Center of Gravities has no significant impact on decreasing of the required torque. The results of the springs' optimization procedures showed that the use of springs, as passive control elements, leads to reduction in the required torque. Various cases of aiding spring's installation either replacing or assisting the driving motor were considered. The main contribution of this thesis is the design of the Hoeken's mechanism and its spring's optimization for curved motion. Some questions

appears from the analysis in this work. Mechanism with dimensions considered in this work can perform only one trajectory. It is required to design adjustable mechanism that generates various profiles, which can enhance the rehabilitation procedure. When springs are used instead of actuator, it is important to define how to set the mechanism to “initial position” after performing one cycle of motion and how to connect springs with links. According to the results, there is a sharp shifting of torque when accelerating spring stops acting on the mechanism and the mechanism “meets” the decelerating spring. Further research will investigate the installation of springs, so that accelerating part of motion smoothly “switches” to decelerating part. Also kinetostatic analysis performed in this work should be validated by solving forward dynamic problem.

To conclude, the results of this thesis represent that proposed methods can be used as the basis in development of cost-effective mechanism, which can be employed in upper limb rehabilitation.

References

[1] "Stroke: a global response is needed", Bulletin of the World Health Organization, Retrieved from: http://www.who.int/bulletin/volumes/94/9/16-181636/en/.

[2] Butefisch, C., Hummelsheim, H., Denzler, P., and Mauritz, K., May 1995. "Repetitive training of isolated movements improves the outcome of motor rehabilitation of the centrally paretic hand". Journal of the Neurological Sciences, 130(10), pp. 59–68.

[3] Rosati, G., Secoli, R., Zanotto, D., Rossi, A., and Boschetti, G., 2008, "Planar robotic systems for upper-limb post-stroke rehabilitation," in ASME2008 International Mechanical Engineering Congress and Exposition, pp. 115–124.

[4] Micera, S., et al., 2005. "A simple robotic system for neurorehabilitation". Autonomous Robots, 19(3), pp. 271–284.

[5] Zollo, L., Accoto, D., Torchiani, F., Formica, D., and Guglielmelli, E., 2008. "Design of a planar robotic machine for neuro-rehabilitation". In Proc. of the IEEE International Conference on Robotics and Automation, pp.2031-2036.

[6] Hogan, N., Krebs, H., Charnnarong, J., Srikrishna, P., and Sharon, A., 1992. "Mit-manus: A workstation for manual therapy and training". In IEEE International Workshop on Robot and Human Communication, pp. 161–165.

[7] Masia, L., Krebs, H., Cappa, P., and Hogan, N., 2007. "Design and characterization of hand module for whole-arm rehabilitation following stroke", IEEE/ASME Transaction on Mechatronics, 12(4), pp. 399–407.

[8] Casadio, M., Sanguineti, V., Morasso, P., and Arrichiello, V., 2006. "Braccio di ferro: a new haptic workstation for neuromotor rehabilitation", Technology Health Care, 14(3), pp. 123–142.

[9] Q. Li, D. Wang, Z. Du, and L. Sun, 2005. "A novel rehabilitation system for upper limbs," Annual International Conference of the IEEE Engineering in Medicine and Biology Society, vol. 7, pp. 6840–6843.

[10] Rosati, G., Gallina, P., and Masiero, S., 2007. "Design, implementation and clinical tests of a wire-based robot for neurorehabilitation". IEEE Transactions on Neural Systems and Rehabilitation Engineering, 15(4), pp.560-569.

[11] Rosati, G., Gallina, P., Rossi, A., and Masiero, S., 2006. "Wire-based robots for upper-limb rehabilitation". International Journal of Assistive Robotics and Mechatronics, 7(2), pp. 3–10.

[12] Gallina, P., Rosati, G., and Rossi, A., 2001. "3-d.o.f. wire driven planar haptic interface", Journal of Int. and Robotic Systems: Theory and Applications, 32(1), pp. 23–36.

[13] Maciejasz, P., Eschweiler, J., Gerlach-Hahn, K., Jansen-Troy, A. and Leonhardt S., 2014, "A survey on robotic devices for upper limb rehabilitation", Journal of NeuroEngineering and Rehabilitation, 11(3).

[14] Xydas G. E., 2014, "Synthesis and Analysis of a Chebyschev's Straight Line Four-Bar Linkage for Generating a Minimum-Jerk Velocity Profile", 38th Mechanisms and Robotics Conference (MR), ASME IDETC/CIE, Buffalo, New York, USA.

[15] Xydas G. E., Louca S. L. and Mueller A., 2015, "Analysis and Passive Control of a Four-bar Linkage for the Rehabilitation of Upper-limb Motion", ASME 2015 Dynamic Systems and Control Conference, DSCC 2015, Columbus, OH, USA.

[16] Xydas G. E. and Mueller A., 2015, "Minimally Actuated 4-bar Linkages for Upper Limb Rehabilitation: Performance Analysis with Forward Dynamics",

39th Mechanisms and Robotics Conference (MR), ASME IDETC/CIE, Boston, MA, USA.

[17] Kurmashev, S., Malik, A., Ospanov, S., 2017, "Synthesis, Analysis, Design and Manufacturing of an UpperLimb Rehabilitation Apparatus based on a Hoeken's Four bar Linkage", Final Capstone Work, Nazarbayev University.

[18] Kurmashev, S., Malik, A., Ospanov, S., Xydas, E., Mueller, A., 2018, “Flexibility in upper limb rehabilitation with the use of 1-DOF fourbar linkages”, 42nd Mechanisms and Robotics Conference (MR), ASME IDETC/CIE, Quebec, Canada.

[19] Natesan, Arun K., 1994, “Kinematic analysis and synthesis of four-bar mechanisms for straight line coupler curves”, Thesis, Rochester Institute of Technology.

[20] Shariatfar, M., 2011, “Geometric Design of Planar Four-Bar Mechanisms”, Thesis, McGill University.

[21] Flash, T., Hogan, N., 1985, “The Coordination of Arm Movements: An Experimentally Confirmed Mathematical Model”, The Journal of Neuroscience, 7(5), pp.1688-1703.

[22] "Anthropometry and Biomechaniscs”, NASA, Retrieved from: https://msis.jsc.nasa.gov/sections/section03.htm#_3.2_GENERAL_ANTHROPOMETRICS.

[23] Norton, Robert L., 2008, “Design of machinery: an introduction to the synthesis and analysis of mechanisms and machines”, 4thedition, McGraw-Hill Professional, ISBN: 978-0-07-312158-1.

[24] Xydas G. E. and Mueller A., 2016, “Minimally Actuated 4-bar Linkages for Upper Limb Rehabilitation, MESROB 2016, Graz, Austria.

[25] Xydas G. E. and Loucas S. L., 2018, “Planar conformity of movements in 3D reaching tasks for persons with Multiple Sclerosis”, Human Movement Science, 62, pp.221-234.

Appendices

Appendix A

MATLAB code computing required torque on input link for straight-line motion:

```
%Link dimensions
L1 = 0.1328711;
L2 = 0.06039597;
L3 = 0.1691087;
L4 = L3;

%Link and flywheel masses
m2 = 0.03;
m3 = 0.1;
m4 = 0.06;
mf = 2.5;
%Flywheel radius
Rf = 0.160/2;
%Moment of Inertia
IG2 = m2 * (L2^2 + 0.02^2) / 12 + mf * Rf^2;
IG3 = m3 * (L3^2 + 0.02^2) / 12;
IG4 = m4 * (L4^2 + 0.02^2) / 12;

% Position vectors
v=[1 0 0.5];
R12 = (1-v(1))*L2;
R32 = v(1)*L2;
R23 = (1-v(2))*L3;
R43 = v(2)*L3;
R34 = (1-v(3))*L4;
R14 = v(3)*L4;

%Initial and final coupler point positions
x0 = 0.2591242;
xf = 0.006618034;
y0 = 0.2777567;
yf = 0.2777567;

tf = 0.5; %total movement time
t = linspace(0,tf,100);

%Minimum Jerk Model
Rx = x0 + (x0 - xf) * (-10 .* (t./tf).^3 + 15 .* (t./tf).^4 -
6 .* (t./tf).^5);
Ry = y0 + (y0 - yf) * (-10 .* (t./tf).^3 + 15 .* (t./tf).^4 -
6 .* (t./tf).^5);
VPx = (x0 - xf) * (-30 .* (t./tf).^2 + 60 .* (t./tf).^3 - 30
.* (t./tf).^4);
```

```
APx = (x0 - xf) * (-60 .* (t./tf) + 180 .* (t./tf).^2 - 120 .*
(t./tf).^3);

%Impaired model
imp_t = [0 0.05 0.1 0.15 0.2 0.25 0.3 0.35 0.4 0.45 0.5];
imp_pos = [0.2591 0.235 0.216 0.17 0.13 0.095 0.076 0.05 0.022
0.0093 0.0066];
imp_poly = polyfit(imp_t,imp_pos,5);
imp_poly = poly2sym(imp_poly);
imp_prof = subs(imp_poly, 'x', t);
max_diff = vpa(max(Rx-imp_prof));
img_spr = 15/max_diff;
imp_diff = Rx - imp_prof;

% Position analysis
% Calculation of theta_2
A2 = Rx.^2 + 2 .* Rx .* L2 + L2.^2 + Ry.^2 - 4 .* L3.^2;
B2 = -4 .* Ry .* L2;
C2 = Rx.^2 + Ry.^2 + L2.^2 - 4 .* L3.^2 - 2 .* Rx .* L2;
theta_2 = 2 .* atan( (- B2 + sqrt(B2.^2 - 4 .* A2 .* C2) ) ./
( 2 .* A2 ) );
radtodeg(theta_2);
theta_2 = unwrap(theta_2); % Correct phase angles to produce
smoother phase plots

% Calculation of theta_3
A3 = - Rx.^2 - 4 .* Rx .* L3 + L2.^2 - Ry.^2 - 4 .* L3.^2;
B3 = 8 .* Ry .* L3;
C3 = - Rx.^2 - Ry.^2 + L2.^2 - 4 .* L3.^2 + 4 .* Rx .* L3;
theta_3 = 2 .* atan( (- B3 + sqrt(B3.^2 - 4 .* A3 .* C3) ) ./
( 2 .* A3 ) );
radtodeg(theta_3);

% Calculation of theta_4
A4 = - Rx.^2 + 2 .* Rx .* L1 - L4.^2 - Ry.^2 + L3.^2 - L1.^2 -
2 .* Rx .* L4 + 2 .* L4 .* L1;
B4 = 4 .* Ry .* L4;
C4 = - Rx.^2 - Ry.^2 + L3.^2 - L4.^2 + 2 .* Rx .* L1 + 2 .* Rx
.* L4 - 2 .* L4 .* L1 - L1.^2;
theta_4 = 2 .* atan( (- B4 - sqrt(B4.^2 - 4 .* A4 .* C4) ) ./
( 2 .* A4 ) );
radtodeg(theta_4);

% Velocity and Acceleration Analysis
for i = 1:1:100
% Velocity matrix
M1 = [-L2 .* sin(theta_2(i))     -L3 .* sin(theta_3(i))
L4 .* sin(theta_4(i))    0 0 0 0 0;
       L2 .* cos(theta_2(i))      L3 .* cos(theta_3(i))      -
L4 .* cos(theta_4(i))    0 0 0 0 0;
```

```
       L2 .* sin(theta_2(i))       0
0                        1 0 0 0 0;
      -L2 .* cos(theta_2(i))       0
0                        0 1 0 0 0;
       0                            2*L3 .* sin(theta_3(i))
0                        0 0 1 0 0;
       0                           -2*L3 .* cos(theta_3(i))
0                        0 0 0 1 0;
      -L2 .* sin(theta_2(i))      -2 .* L3 .* sin(theta_3(i))
0                        0 0 0 0 0;
      -L2 .* cos(theta_2(i))      -2 .* L3 .* cos(theta_3(i))
0                        0 0 0 0 1];

Y1 = [0; 0; 0; 0; 0; 0; VPx(i); 0];

X1 = M1\Y1;

w2(i) = X1(1,1);
w3(i) = X1(2,1);
w4(i) = X1(3,1);
VAx(i) = X1(4,1);
VAy(i) = X1(5,1);
VBAx(i) = X1(6,1);
VBAy(i) = X1(7,1);
VPy(i) = X1(8,1);

%Acceleration matrix
M2 = [-L2 .* sin(theta_2(i))      -L3 .* sin(theta_3(i))
L4 .* sin(theta_4(i))    0 0 0 0 0;
       L2 .* cos(theta_2(i))       L3 .* cos(theta_3(i))      -
L4 .* cos(theta_4(i))    0 0 0 0 0;
       L2 .* sin(theta_2(i))       0
0                        1 0 0 0 0;
      -L2 .* cos(theta_2(i))       0
0                        0 1 0 0 0;
       0                            2*L3 .* sin(theta_3(i))
0                        0 0 1 0 0;
       0                           -2*L3 .* cos(theta_3(i))
0                        0 0 0 1 0;
      -L2 .* sin(theta_2(i))      -2 .* L3 .* sin(theta_3(i))
0                        0 0 0 0 0;
      -L2 .* cos(theta_2(i))      -2 .* L3 .* cos(theta_3(i))
0                        0 0 0 0 1];

Y2 = [ L2 .* w2(i)^2 .* cos(theta_2(i)) + L3 .* w3(i)^2 .*
cos(theta_3(i)) - L4 .* w4(i)^2 .* cos(theta_4(i));
       L2 .* w2(i)^2 .* sin(theta_2(i)) + L3 .* w3(i)^2 .*
sin(theta_3(i)) - L4 .* w4(i)^2 .* sin(theta_4(i));
      -L2 .* w2(i)^2 .* cos(theta_2(i));
      -L2 .* w2(i)^2 .* sin(theta_2(i));
      -2*L3 .* w3(i)^2 .* cos(theta_3(i));
```

```
      -2*L3 .* w3(i)^2 .* sin(theta_3(i));
       APx(i) + L2 .* w2(i)^2 .* cos(theta_2(i)) + 2 .* L3 .*
w3(i)^2 .* cos(theta_3(i));
      -L2 .* w2(i)^2 .* sin(theta_2(i)) - 2 .* L3 .* w3(i)^2
.* sin(theta_3(i))];

X2 = M2\Y2;

a2(i) = X2(1,1);
a3(i) = X2(2,1);
a4(i) = X2(3,1);
AAx(i) = X2(4,1);
AAy(i) = X2(5,1);
ABAx(i) = X2(6,1);
ABAy(i) = X2(7,1);
APy(i) = X2(8,1);

% Accelerations of center of gravities
ACG2x(i) = -(1-v(1))*L2 * a2(i) * sin (theta_2(i)) - (1-
v(1))*L2 * (w2(i))^2 * cos(theta_2(i));
ACG2y(i) =  (1-v(1))*L2 * a2(i) * cos (theta_2(i)) - (1-
v(1))*L2 * (w2(i))^2 * sin(theta_2(i));
ACG3x(i) = AAx(i) + ABAx(i);
ACG3y(i) = AAy(i) + ABAy(i);
ACG4x(i) = -v(3)*L4 * a4(i) * sin (theta_4(i)) - v(3)*L4 *
(w4(i))^2 * cos(theta_4(i));
ACG4y(i) = v(3)*L4 * a4(i) * cos (theta_4(i)) - v(3)*L4 *
(w4(i))^2 * sin(theta_4(i));

% External impaired force
imp_force(i) = img_spr .* imp_diff(i);

% x and y components of the position vectors
R12x = -R12 * cos(theta_2(i));
R12y = -R12 * sin(theta_2(i));
R32x = R32 * cos(theta_2(i));
R32y = R32 * sin(theta_2(i));
R23x = -R23 * cos(theta_3(i));
R23y = -R23 * sin(theta_3(i));
R43x = R43 * cos(theta_3(i));
R43y = -R43 * sin(theta_3(i));
R34x = R34 * cos(theta_4(i));
R34y = R34 * sin(theta_4(i));
R14x = R14 * cos(theta_4(i));
R14y = -R14 * sin(theta_4(i));
RPx = (1-v(2))*L3 * cos(theta_3(i));
RPy = (1-v(2))*L3* sin(theta_3(i));
FPx = imp_force(i)/sqrt(2);
FPy = imp_force(i)/sqrt(2);

%Force matrix
```

```
M3 = [  1      0       1        0        0         0        0       0
0;
        0      1       0        1        0         0        0       0
0;
      -R12y   R12x   -R32y     R32x      0         0        0       0
1;
        0      0      -1        0        1         0        0       0
0;
        0      0       0       -1        0         1        0       0
0;
        0      0      R23y    -R23x    -R43y      R43x      0       0
0;
        0      0       0        0       -1         0        1       0
0;
        0      0       0        0        0        -1        0       1
0;
        0      0       0        0       R34y     -R34x    -R14y
R14x      0];

Y3 = [m2 * ACG2x(i);
      m2 * ACG2y(i);
      IG2 * a2(i);
      m3 * ACG3x(i) - FPx;
      m3 * ACG3y(i) - FPy;
      IG3 * a3(i) - RPx * FPy + RPy * FPx;
      m4 * ACG4x(i);
      m4 * ACG4y(i);
      IG4 * a4(i)];

X3 = M3\Y3;

F12x(i) = X3(1);
F12y(i) = X3(2);
F32x(i) = X3(3);
F32y(i) = X3(4);
F43x(i) = X3(5);
F43y(i) = X3(6);
F14x(i) = X3(7);
F14y(i) = X3(8);
T2(i) = X3(9);
end

 figure
 plot(radtodeg(theta_2),T2)
 title('Required torque on input link ')
 ylabel('Torque, [N*m]')
 xlabel('theta_2, [degrees]')
 grid on
```

Appendix B

MATLAB code computing required torque on input link for curved motion:

```
%Link dimensions
L1 = 0.1328;
L2 = 0.0604;
L3 = 0.1692409;
L4 = L3;

%Link and flywheel masses
m2 = 0.03;
m3 = 0.1;
m4 = 0.06;
mf = 2.5;
Rf = 0.160/2;
IG2 = m2 * (L2^2 + 0.02^2) / 12+ mf * Rf^2;
IG3 = m3 * (L3^2 + 0.02^2) / 12;
IG4 = m4 * (L4^2 + 0.02^2) / 12;

% position vectors
v=[1 0 0.5];
R12 = (1-v(1))*L2;
R32 = v(1)*L2;
R23 = (1-v(2))*L3;
R43 = v(2)*L3;
R34 = (1-v(3))*L4;
R14 = v(3)*L4;

% initial, intermediate and final positions of coupler point
xf = 0.2692255;
x0 = -0.01197358;
yf = 0.2941459;
y0 = 0.2932582;
tf = 0.5;
x1 = 0.1207077;
y1 = 0.3330677;
t1 = 0.25;

tau1 = t1 / tf;
c1 = 1 / ( tf^5 * tau1^2 * ( 1 - tau1 )^5 ) * ( ( xf - x0 ) *
( 300 * tau1^5 - 1200 * tau1^4 + 1600 * tau1^3 )...
     + tau1^2 * ( -720 * xf + 120 * x1 + 600 * x0 )...
      + ( x0 - x1 ) * ( 300 * tau1 - 200 ) );
pi1 = 1 / ( tf^5 * tau1^5 * ( 1 - tau1 )^5 ) * ( ( xf - x0 ) *
( 120 * tau1^5 ...
     - 300 * tau1^4 + 200 * tau1^3 ) - 20 * ( x1 - x0 ) );

c2 = 1 / ( tf^5 * tau1^2 * ( 1 - tau1 )^5 ) * ( ( yf - y0 ) *
( 300 * tau1^5 - 1200 * tau1^4 + 1600 * tau1^3 )...
     + tau1^2 * ( -720 * yf + 120 * y1 + 600 * y0 )...
```

```
      + ( y0 - y1 ) * ( 300 * tau1 - 200 ) );
pi2 = 1 / ( tf^5 * tau1^5 * ( 1 - tau1 )^5 ) * ( ( yf - y0 ) *
( 120 * tau1^5 ...
     - 300 * tau1^4 + 200 * tau1^3 ) - 20 * ( y1 - y0 ) );

i = 1;

% MJM position, velocity and acceleration
for t = 0:0.005:t1

    tau = t / tf;

    Rx ( 1, i ) = tf^5 / 720 * ( pi1 * ( tau1^4 * ( 15 * tau^4
- 30 * tau^3 )...
         + tau1^3 * ( 80 * tau^3 - 30 * tau^4 ) - 60 * tau^3 *
tau1^2 + 30 * tau^4 * tau1 - 6 * tau^5 )...
          + c1 * ( 15 * tau^4 - 10 * tau^3 - 6 *tau^5 ) ) +
x0;
    Ry ( 1, i ) = tf^5 / 720 * ( pi2 * ( tau1^4 * ( 15 * tau^4
- 30 * tau^3 )...
         + tau1^3 * ( 80 * tau^3 - 30 * tau^4 ) - 60 * tau^3 *
tau1^2 + 30 * tau^4 * tau1 - 6 * tau^5 )...
          + c2 * ( 15 * tau^4 - 10 * tau^3 - 6 *tau^5 ) ) +
y0;

    VPx ( 1, i ) = tf^5 / 720 * ( pi1 * ( tau1^4 * ( 60*tau^3-
90*tau^2)...
         + tau1^3 * ( 240*tau^2 - 120*tau^3) - 180
*tau^2*tau1^2 + 120 * tau^3 * tau1 - 30*tau^4)...
          + c1 * ( 60*tau^3 - 30*tau^2 - 30*tau^4) );
    VPy ( 1, i ) = tf^5 / 720 * ( pi2 * ( tau1^4 * ( 60*tau^3-
90*tau^2)...
         + tau1^3 * ( 240*tau^2 - 120*tau^3) - 180
*tau^2*tau1^2 + 120 * tau^3 * tau1 - 30*tau^4)...
          + c2 * ( 60*tau^3 - 30*tau^2 - 30*tau^4) );

     APx ( 1, i ) = tf^5 / 720 * ( pi1 * ( tau1^4 * (
180*tau^2-180*tau)...
         + tau1^3 * ( 480*tau - 360*tau^2) - 360 *tau*tau1^2 +
360 * tau^2 * tau1 - 120*tau^3)...
          + c1 * ( 180*tau^2 - 60*tau - 120*tau^3) );

    i = i + 1;

end

for t = t1+0.005:0.005:tf

    tau = t / tf;
```

```
    Rx ( 1, i ) = tf^5 / 720 * ( pi1 * ( tau1^4 * ( 15 * tau^4
- 30 * tau^3 +30 * tau - 15)...
         + tau1^3 * (  - 30 * tau^4 + 80 * tau^3 - 60 * tau^2
+ 10) )...
          + c1 * ( - 6 *tau^5 + 15 * tau^4 - 10 * tau^3  + 1 )
) + xf;
    Ry ( 1, i ) = tf^5 / 720 * ( pi2 * ( tau1^4 * ( 15 * tau^4
- 30 * tau^3 +30 * tau - 15)...
         + tau1^3 * (  - 30 * tau^4 + 80 * tau^3 - 60 * tau^2
+ 10) )...
          + c2 * ( - 6 *tau^5 + 15 * tau^4 - 10 * tau^3  + 1 )
) + yf;
    VPx ( 1, i ) = tf^5 / 720 * ( pi1 * ( tau1^4 * ( 60*tau^3-
90*tau^2+30)...
         + tau1^3 * (-120*tau^3+240*tau^2-120*tau))...
          + c1 * ( -30*tau^4+60*tau^3-30*tau^2) );
      VPy ( 1, i ) = tf^5 / 720 * ( pi2 * ( tau1^4 * (
60*tau^3-90*tau^2+30 )...
         + tau1^3 * (-120*tau^3+240*tau^2-120*tau ))...
          + c2 * ( -30*tau^4+60*tau^3-30*tau^2 ) );
      APx ( 1, i ) = tf^5 / 720 * ( pi1 * ( tau1^4 * (
180*tau^2-180*tau)...
         + tau1^3 * (-360*tau^2+480*tau-120 ))...
          + c1 * ( -120*tau^3+180*tau^2-60*tau) );
      i = i + 1;
end

t = linspace(0,tf,101);

%Impaired model
imp_t = [0 0.025 0.05 0.1 0.15 0.2 0.25 0.3 0.35 0.4 0.425
0.45 0.475 0.5];
imp_pos =[ -0.0120   -0.0930   -0.0186   -0.0600   -0.0050
0.0300    0.0750 0.1820    0.2380    0.2700    0.2462
0.2726    0.2726    0.2746];
imp_poly = polyfit(imp_t,imp_pos,5);
imp_poly = poly2sym(imp_poly);
imp_prof = subs(imp_poly, 'x', t);

max_diff = vpa(max(abs(Rx-imp_prof)));
img_spr = 15/max_diff;
imp_diff = Rx - imp_prof;

% Calculation of theta_2
A2 = Rx.^2 + 2 .* Rx .* L2 + L2.^2 + Ry.^2 - 4 .* L3.^2;
B2 = -4 .* Ry .* L2;
C2 = Rx.^2 + Ry.^2 + L2.^2 - 4 .* L3.^2 - 2 .* Rx .* L2;
theta_2 = 2 .* atan( (- B2 - sqrt(B2.^2 - 4 .* A2 .* C2) ) ./
( 2 .* A2 ) );
theta_2_deg=radtodeg(theta_2);
```

```
% Calculation of theta_3
A3 = - Rx.^2 - 4 .* Rx .* L3 + L2.^2 - Ry.^2 - 4 .* L3.^2;
B3 = 8 .* Ry .* L3;
C3 = - Rx.^2 - Ry.^2 + L2.^2 - 4 .* L3.^2 + 4 .* Rx .* L3;
theta_3 = 2 .* atan( (- B3 - sqrt(B3.^2 - 4 .* A3 .* C3) ) ./
( 2 .* A3 ) );
radtodeg(theta_3);


% Calculation of theta_4
A4 = - Rx.^2 + 2 .* Rx .* L1 - L4.^2 - Ry.^2 + L3.^2 - L1.^2 -
2 .* Rx .* L4 + 2 .* L4 .* L1;
B4 = 4 .* Ry .* L4;
C4 = - Rx.^2 - Ry.^2 + L3.^2 - L4.^2 + 2 .* Rx .* L1 + 2 .* Rx
.* L4 - 2 .* L4 .* L1 - L1.^2;
theta_4 = 2 .* atan( (- B4 - sqrt(B4.^2 - 4 .* A4 .* C4) ) ./
( 2 .* A4 ) );
radtodeg(theta_4);

for i = 1:1:101
% Velocity matrix
M1 = [-L2 .* sin(theta_2(i))     -L3 .* sin(theta_3(i))
L4 .* sin(theta_4(i))    0 0 0 0 ;
       2*L2 .* cos(theta_2(i))    3*L3 .* cos(theta_3(i))    -
L4 .* cos(theta_4(i))    0 0 0 0 ;
       L2 .* sin(theta_2(i))      0
0                        1 0 0 0 ;
      -L2 .* cos(theta_2(i))      0
0                        0 1 0 0 ;
       0                          L3 .* sin(theta_3(i))     0
0 0 1 0 ;
       0                         -L3 .* cos(theta_3(i))     0
0 0 0 1 ;
      -L2 .* sin(theta_2(i))     -2 .* L3 .* sin(theta_3(i))
0                        0 0 0 0] ;

Y1 = [0; VPy(i); 0; 0; 0; 0; VPx(i)];

X1 = M1\Y1;

w2(i) = X1(1,1);
w3(i) = -X1(2,1);
w4(i) =  -X1(3,1);
VAx(i) = X1(4,1);
VAy(i) = X1(5,1);
VBAx(i) = X1(6,1);
VBAy(i) = X1(7,1);

% Acceleration matrix
```

```
M2 = [-L2 .* sin(theta_2(i))     -L3 .* sin(theta_3(i))
L4 .* sin(theta_4(i))    0 0 0 0 0;
       L2 .* cos(theta_2(i))      L3 .* cos(theta_3(i))      -
L4 .* cos(theta_4(i))    0 0 0 0 0;
       L2 .* sin(theta_2(i))      0
0                        1 0 0 0 0;
      -L2 .* cos(theta_2(i))      0
0                        0 1 0 0 0;
       0                          L3 .* sin(theta_3(i))     0
0 0 1 0 0;
       0                         -L3 .* cos(theta_3(i))     0
0 0 0 1 0;
      -L2 .* sin(theta_2(i))     -2 .* L3 .* sin(theta_3(i))
0                        0 0 0 0 0;
      -L2 .* cos(theta_2(i))     -2 .* L3 .* cos(theta_3(i))
0                        0 0 0 0 1];

Y2 = [ L2 .* w2(i)^2 .* cos(theta_2(i)) + L3 .* w3(i)^2 .*
cos(theta_3(i)) - L4 .* w4(i)^2 .* cos(theta_4(i));
       L2 .* w2(i)^2 .* sin(theta_2(i)) + L3 .* w3(i)^2 .*
sin(theta_3(i)) - L4 .* w4(i)^2 .* sin(theta_4(i));
      -L2 .* w2(i)^2 .* cos(theta_2(i));
      -L2 .* w2(i)^2 .* sin(theta_2(i));
      -L3 .* w3(i)^2 .* cos(theta_3(i));
      -L3 .* w3(i)^2 .* sin(theta_3(i));
       APx(i) + L2 .* w2(i)^2 .* cos(theta_2(i)) + 2 .* L3 .*
w3(i)^2 .* cos(theta_3(i));
      -L2 .* w2(i)^2 .* sin(theta_2(i)) - 2 .* L3 .* w3(i)^2
.* sin(theta_3(i))];

X2 = M2\Y2;

a2(i) = X2(1,1);
a3(i) = -X2(2,1);
a4(i) = -X2(3,1);
AAx(i) = X2(4,1);
AAy(i) = X2(5,1);
ABAx(i) = X2(6,1);
ABAy(i) = X2(7,1);
APy(i) = X2(8,1);

% % Accelerations of center of gravities
ACG2x(i) = -(1-v(1))*L2 * a2(i) * sin (theta_2(i)) - (1-
v(1))*L2 * (w2(i))^2 * cos(theta_2(i));
ACG2y(i) =  (1-v(1))*L2 * a2(i) * cos (theta_2(i)) - (1-
v(1))*L2 * (w2(i))^2 * sin(theta_2(i));
ACG3x(i) = AAx(i) + ABAx(i);
ACG3y(i) = AAy(i) + ABAy(i);
ACG4x(i) = -v(3)*L4 * a4(i) * sin (theta_4(i)) - v(3)*L4 *
(w4(i))^2 * cos(theta_4(i));
```

```
ACG4y(i) = v(3)*L4 * a4(i) * cos (theta_4(i)) - v(3)*L4 *
(w4(i))^2 * sin(theta_4(i));

% User force
imp_force(i) = img_spr .* imp_diff(i);

%x and y components of the center of gravity vectors
R12x = -R12 * cos(theta_2(i));
R12y = -R12 * sin(theta_2(i));
R32x = R32 * cos(theta_2(i));
R32y = R32 * sin(theta_2(i));
R23x = -R23 * cos(theta_3(i));
R23y = -R23 * sin(theta_3(i));
R43x = R43 * cos(theta_3(i));
R43y = -R43 * sin(theta_3(i));
R34x = R34 * cos(theta_4(i));
R34y = R34 * sin(theta_4(i));
R14x = R14 * cos(theta_4(i));
R14y = -R14 * sin(theta_4(i));
RPx = L3 * cos(theta_3(i));
RPy = L3* sin(theta_3(i));
FPx = imp_force(i);
FPy = 0;

%Force matrix
M3 = [  1     0      1       0       0        0       0      0
0;
        0     1      0       1       0        0       0      0
0;
      -R12y  R12x  -R32y    R32x     0        0       0      0
1;
        0     0     -1       0       1        0       0      0
0;
        0     0      0      -1       0        1       0      0
0;
        0     0     R23y   -R23x   -R43y     R43x     0      0
0;
        0     0      0       0       -1       0       1      0
0;
        0     0      0       0       0       -1       0      1
0;
        0     0      0       0      R34y    -R34x   -R14y
R14x     0];

Y3 = [m2 * ACG2x(i);
      m2 * ACG2y(i);
      IG2 * a2(i);
      m3 * ACG3x(i) - FPx;
      m3 * ACG3y(i) - FPy;
      IG3 * a3(i) - RPx * FPy + RPy * FPx;
      m4 * ACG4x(i);
```

```
      m4 * ACG4y(i);
      IG4 * a4(i)];

X3 = M3\Y3;

F12x(i) = X3(1);
F12y(i) = X3(2);
F32x(i) = X3(3);
F32y(i) = X3(4);
F43x(i) = X3(5);
F43y(i) = X3(6);
F14x(i) = X3(7);
F14y(i) = X3(8);
T2(i) = X3(9);
end

figure
plot(t,T2)
title('Required torque on input link for curved motion')
xlabel('Time, [s]')
ylabel('Torque, [N*m]')
grid on
```